PLOVER - Preliminary List Of Vulnerability Examples for Researchers

[*] Author: Steve Christey (coley@mitre.org)
[*] Date: Match 15, 2006
[*] Document Version: 0.24

Disclaimer: This is a DRAFT document that does not represent an official position of The MITRE Corporation. This document has been created to spur progress in vulnerability classification and vulnerability research.

SECTION.1. [INTRO] Introduction

Recently, there has been renewed interest in the classification and categorization of vulnerabilities, attacks, faults, and other concepts. Past efforts have largely focused on high-level theories, taxonomies, or schemes that do not sufficiently cover the wide variety of security issues that are found in today's products.

PLOVER - the Preliminary List Of Vulnerability Examples for Researchers - is a working document that lists over 1400 diverse, real-world examples of vulnerabilities, identified by their CVE number. The vulnerabilities are organized within a novel, detailed conceptual framework. The framework does not solve the entire classification problem, but it provides useful discussion points and an effective vocabulary for describing vulnerabilities at a low level of detail.

PLOVER defines a set of terms and concepts that could help in communicating about vulnerabilities at an abstract level.

PLOVER is intended for use by parties who are interested in vulnerability research and classification, including academic researchers, code auditing tool developers, secure programming researchers, and others. It is a resource for knowledgeable and skilled vulnerability analysts and may be of less use to the general public.

PLOVER includes:

 [*] Vulnerability Theory: a conceptual framework for describing and

discussing several aspects of vulnerabilities at a low level

[*] High-level and low-level vulnerability types and definitions, the relevant attributes, and the inter-relationships between those types, for a current total of 290 types

[*] Over 1400 real-world examples of vulnerabilities, identified by their CVE name

[*] Discussion of current terminology and its limitations

[*] Research gaps

PLOVER is an extension and improvement of the "Vulnerability Auditing Checklist," which was posted to various security mailing lists between 2002 and 2004. That checklist has been retired, although PLOVER can still be used as a checklist.

After review and revision, it is hoped that PLOVER can become a concise, well-defined, commonly used set of terms and concepts that will improve communications regarding vulnerabilities, support the development and evaluation of code analysis tools, and provide a rich environment for academic research.

PLOVER will be used by the CVE project to (1) define terms used in CVE descriptions, (2) provide clarity in distinguishing between different bug types when applying CVE content decisions, and (3) perform more precise vulnerability trend analysis.

It must be emphasized that PLOVER, while extensive, is a working document that may contain errors or omissions. It has not been closely validated or compared with past efforts. However, due to the increased interest in vulnerability classification, the author believes that PLOVER can be a useful resource for advancing vulnerability theory within the security community.

SECTION.2. [TOC] Table of Contents

SECTION.3. [DEFS] Terms and Definitions

This section identifies the most critical terms, definitions, and concepts that are used in PLOVER. They are used throughout the rest of the document.

Use of these terms and concepts may improve communication about vulnerability theory.

==========

DEFS.CDEFS. Core Definitions

PROPERTY: A characteristic of data or an action (step) that is relevant to the security of a product. Examples: Is the data well-formed? Is the step allowed given the previous step?

ATTACKER: A person or independently executing program that intends to compromise the confidentiality, integrity, or availability of a product.

MANIPULATION: A modification by an ATTACKER of a data element, group of elements, action, or group of actions based on one or more PROPERTIES. Examples: modify the input by removing a required argument; perform steps out of order.

WIFF: Weakness, Idiosyncrasy, Flaw, or Fault. An algorithm, sequence of code, or a configuration in the product, whether it arises from implementation, design, or other processes, that can cross data or object boundaries that could not be crossed during normal operation of the product.

CONSEQUENCE: An action performed by the product after a data or object boundary has been crossed, which could not have occurred otherwise.

CHANNEL: A communications channel, or an interface, between two entities.

ATTACK VECTOR: The minimal set of MANIPULATIONS, CHANNELs, and operational constraints, by the attacker or the product, that are required to cause the product to reach a WIFF through one or more CHANNELs.

ATTACK CHANNEL: A CHANNEL in an ATTACK VECTOR that must be controlled or influenced by an ATTACKER for the attack to succeed.

VULNERABILITY: A WIFF in a specific product, or a design intended for a class of products that provide the same functionality, that has at least one ATTACK VECTOR.

ATTACK: The set of actions by which an ATTACKER follows an ATTACK VECTOR to exploit a VULNERABILITY to achieve a desired CONSEQUENCE.

==========

DEFS.ODEFS. Other definitions

RESULTANT: Only existing as a result of another WIFF, VULNERABILITY, or CONSEQUENCE.

PRIMARY: Existing independently of another WIFF, VULNERABILITY, or CONSEQUENCE.

MULTI-FACTOR VULNERABILITY (MFV): A vulnerability that contains two or more WIFFs, two or more manipulations, or two or more attack channels.

MULTI-CHANNEL VULNERABILITY: A vulnerability whose attack vector contains two or more attack channels that must be controlled by the attacker.

MULTI-CHANNEL ATTACK: An ATTACK on a multi-channel vulnerability.

MULTI-MANIPULATION ATTACK: An ATTACK that requires two or more "trigger" manipulations.

ATOMIC CONSEQUENCE: The first low-level product action that crosses data or object boundaries. Examples: read or write data past boundary, perform operation on wrong object.

FUNCTIONAL CONSEQUENCE: A higher-level action whose security implications can only be described at the functional level of the product. Examples: source code disclosure, authentication bypass, code execution.

DIAGNOSIS: The process by which a person analyzes the product in order to identify the underlying WIFFs, CONSEQUENCES, MANIPULATIONS, or ATTACK VECTORS of a VULNERABILITY.

==========

DEFS.DPROP. Data Properties

There are several properties of data that are relevant to vulnerabilities.

STRUCTURE: The Data is either well-formed or malformed.

VALIDITY: Data is either valid or invalid. Invalid data includes (1) data of the wrong type (e.g. an alphabetic string when a number is expected), (2) an out-of-range numeric value, or (3) an undefined value (e.g. "Maybe" when the expected answers are either "Yes" or "No").

CONSISTENCY: The relationships between data elements or steps are either consistent or inconsistent. Examples: when the boundary string specified in a multipart MIME header is used in the body (consistent), or when the length field for an input does not match the actual length of the input.

EQUIVALENCE: Equivalence determines whether multiple identifiers or references can exist for the same entity within a particular context. The data can be "equivalent" or "exclusive." An example of exclusive data is a primary record key in a database. In a data entry application, "F" and "Female" and "f" might all be treated as equivalent when entered by the user.

ENCODING: There may be multiple encodings or representations that are supported for the data. For example, a web application might accept straight ASCII text, URL encoding sequences such as "%20", or Unicode.

MUTABILITY: This specifies whether the data is expected to vary as the product executes. For example, a search query, a subject line in a forum post, or the name of a new user in a registration form might all be mutable; an internal buffer for storing the administrator's e-mail address might be immutable.

Note that vulnerabilities can arise from violations of expected properties of data. Frequently, data can be manipulated in ways that violate the developer's assumptions. Each of the above properties might be an assumed property by the programmer, which could lead to WIFFs. It might be useful to discuss certain data elements in these terms, e.g. "assumed-immutable", "assumed-consistent", or "assumed-exclusive."

For example, a PHP file include vulnerability might allow direct

requests to support scripts (assumed-valid access) that can facilitate modification of global variables (assumed-immutable).

==========

DEFS.ALT. Alternate Elements

Alternate elements are elements that have more than one identifier, reference, object, or method of access. They are important factors in many vulnerabilities, so they are briefly described here. More specific examples are provided in other sections.

ALTERNATE CHANNEL: A specific action or data in a product is accessible through one channel, but another channel exists. Example: a web server opens up another listening port on TCP/8080.

ALTERNATE NAME: ("alias"). An entity has a name or identifier that is typically used, but there are other names/identifiers that identify the same entity. Example: "abc/def.txt" and "ghi/../abc/def.txt" are alternate names for the same file.

ALTERNATE PATH: Within a single channel, the product has one typical "path" of steps that the user must follow to reach a certain functionality, but there are other paths that reach the same functionality. Example: "admin.php" is assumed to be reachable only from links within "index.php", but the attacker can directly access "admin.php".

****

DEFS.ALT.NAMES. Examples of Alternate Names

Some alternate names include:

[*] symbolic links
[*] hard links
[*] ".." in a path
[*] absolute path
[*] relative path
[*] "C:" drive letter (Windows)
[*] 8.3 filenames
[*] CLSID

NOTE: the current list includes mostly filenames, but there are other examples.

```
================================================================
DEFS.MANIPS.  Manipulations

There are two main classes of manipulations:

  Data manipulation: data is modified

  Step manipulation: steps are modified

****

Data manipulations may include, but are not limited to:

   [*] providing more or less input than expected
   [*] inject special character
   [*] use invalid syntax
   [*] using an alternate encoding
   [*] omitting a required value
   [*] providing data of the wrong type
   [*] modifying one item so that it is inconsistent with another item

Step manipulations may include, but are not limited to:

   [*] skip first step
   [*] skip a required step
   [*] perform steps out of order
   [*] perform repeated steps
   [*] do not finish step
   [*] interrupt step

NOTE: more specific examples are provided in another section.

****

Some examples of manipulations at the product level include:

   [*] well-formed data with an invalid value, e.g. a web command:
       GETTT / HTTP/1.0
       http://www.example.com/

   [*] malformed data with valid value
     - GET /
     - "GET" is a valid command and "/" is a valid URI, but there's
       no version specifier, so the input is malformed
```

```
   [*] well-formed data with valid value:
     - ABCDEF~1.DAT   (equivalent filename for "ABCDEFGHIJ.DAT")

   [*] well-formed data with inconsistent value
     - $String = "Hello World!"; $StringLength = 2;

   [*] log into FTP server and send PASS command before USER

   [*] connect to telnet server but don't send any data

   [*] exit connection while server is still sending data

   [*] press "Escape" key instead of entering screensaver password


****

The same manipulation may have different data properties depending on
the context.  For example, the string "O'Neill" is valid in a text
file but not a SQL query.  "i < 3" is a well-formed expression in
Javascript, but it is syntactically incorrect in HTML.  This is a
strong argument for performing canonicalization ONLY at data borders -
as soon as it comes in, and just before it goes out.

****

Manipulations serve different roles to an attacker.

TRIGGER: specifically intended to exploit a WIFF.

ESSENTIAL: must be performed to properly interact with the product.
  Examples: a parameter in a CGI script must be base64-encoded; or,
  the attacker must log in and navigate to a specific menu.  These
  manipulations are, by definition, valid and well-formed.

FACILITATOR: must be performed to work within the constraints of
  product execution.  Examples: shellcode for a buffer overflow
  exploit must be less than 100 bytes and cannot contain any null
  characters; an XSS issue requires a ">" before the malicious string
  in order to terminate an open HTML tag being generated by the
  product.

Note that PLOVER only covers Trigger manipulations.
```

==========

DEFS.CON. Consequences

As defined above, there are two types of consequences, atomic and functional.

Note that functional consequences can be primary or resultant. For example, a SQL injection issue might have a primary functional consequence of modifying a database; if the database is used for authentication, then the resultant consequence is modification or theft of authentication credentials. A primary disk consumption could result in resultant CPU consumption as the processor does more bookkeeping work than normally needed.

Note that a "bypass" Consequence can occur as a result of manipulations of different alternate entity properties such as alternate name and alternate channel.

****

DEFS.CON.ATOM. Atomic Consequences

These are informal categories that may partially overlap. They are intended to demonstrate the concept rather than precisely define it.

- [*] out-of-bounds write (buffer overflow or underflow)
- [*] out-of-bounds read
- [*] execute code
- [*] operation on wrong entity
- [*] wrong operation on entity
- [*] numeric overflow
- [*] undefined mathematical operation (e.g. divide-by-zero)
- [*] invalid pointer dereference, including null dereference
- [*] infinite loop
- [*] long loop
- [*] infinite recursion
- [*] deep recursion
- [*] deadlock
- [*] access of stale identifier
- [*] access of previously freed memory, including double-free
- [*] access of uninitialized memory

****

DEFS.CON.FUNC. Functional Consequences

These are informal categories that may partially overlap. They are intended to demonstrate the concept rather than precisely define it.

- [*] path traversal
- [*] code execution
- [*] command execution
- [*] path disclosure
- [*] information leak
- [*] username enumeration
- [*] source code disclosure
- [*] authentication bypass
- [*] filter bypass
- [*] detection evasion / information hiding
- [*] wrong operation on object
- [*] operation on wrong object
- [*] hang or freeze
- [*] corrupt memory
- [*] refuse new connections
- [*] drop existing connections
- [*] memory consumption or exhaustion
- [*] CPU consumption or exhaustion
- [*] disk consumption or exhaustion
- [*] resource consumption or exhaustion
- [*] inability to restart
- [*] lockout
- [*] network amplification (e.g. storm)
- [*] data amplification
- [*] authentication credentials disclosure
- [*] obtain meta-data
- [*] decrypt data
- [*] determine filename existence
- [*] hide activities
- [*] hide attack source
- [*] disabled or weakened security feature
- [*] gain additional privileges, rights, roles, etc.
- [*] modify permissions or ACLs

Each of these operations can be controlled (attacker has full control over the operation on the object), partially controlled, or uncontrolled. For an uncontrolled consequence, the attacker has no role except to take advantage of the consequence whenever it occurs.

==========

DEFS.CHANNELS. Channels

Here are some examples of channels. Note that any channel can be an attack channel for some vulnerability.

DEFS.CHANNELS.REM. Remote Channels

Remote channels include:

- [*] user-to-server
- [*] server-to-consultant - e.g. RADIUS, DNS server lookups
- [*] user-to-intermediary

DEFS.CHANNELS.LOCAL. Local Channels

Local Channels include:

- [*] command line
- [*] process invocation
- [*] data file or object
- [*] file or directory name
- [*] file descriptor
- [*] profile (e.g. user name, GECOS field)
- [*] environment variable
- [*] signal or semaphore
- [*] registry
- [*] configuration file
- [*] keyboard device
- [*] mouse device
- [*] GUI API
- [*] alternate data stream
- [*] shared memory
- [*] mapped memory
- [*] Windows named pipe

DEFS.CHANNELS.PHYS. Physical Channels

Physical channels include:

- [*] serial port
- [*] keyboard
- [*] mouse
- [*] floppy disk
- [*] CD drive
- [*] USB device

==========
DEFS.ENDPOINTS. Endpoints

Channels exist between two entities, which are called endpoints.

The attacker must perform actions as one (or more) of these endpoints in order to exploit a vulnerability.

USER: user of the product, possibly an administrator

SERVICE: (or server). A networked or local service.

OUTSIDER: an entity that may perform actions outside of the context of the product. For example, an attacker who sends a malicious URL via e-mail to exploit a web application vulnerability, is acting as an outsider. An attack that requires social engineering may involve an outsider.

CONSULTANT: a separate entity that is used by a product to provide information that affects how the product operates. For example, a product uses a DNS server as a consultant in order to look up the IP address of a given hostname; a product that performs authentication might use a RADIUS or LDAP server as a consultant to verify that the provided credentials are correct.

INTERMEDIARY: an entity that controls the channels between endpoints, possibly limiting the kinds of interactions that are allowed within accepted channels. Examples include a firewall, anti-virus product, proxy. Effectively, an intermediary splits a single channel between A and B into two channels - A to the intermediary, and the intermediary to B.

MONITOR: a monitor observes the data or actions that are used within the channel, but it is a passive observer. Examples include a sniffer, log file monitor, or intrusion detection system.

==========
SECTION.4. [VC] Additional Vulnerability Concepts
==========

****

VC.DIRLOC. Direction and Location of Channels

Note: this is a new concept that is still being refined.

Vulnerabilities that require complicated attack chains, especially those that involve more than two endpoints, can be further described in terms of the "direction" and "location" of the channels that are involved.

LOCATION: The LOCATION of a channel is relative to a particular endpoint and to the nature of the interaction when a vulnerable condition is being entered.

The channel's location can be:

EXTERNAL: out of the control of the endpoint, but involving data or steps that are relevant to the endpoint

INTERNAL: involving the endpoint

DIRECTION: The DIRECTION of a channel is also relative to a particular endpoint and to the nature of the interaction when a vulnerable condition is being entered.

The channel's direction can be:

INCOMING: at the particular time, the interaction is being driven by the other end of the channel

OUTGOING: at the particular time, the interaction is being driven by the endpoint itself.

TRANSIENT: the endpoint is a MONITOR and the interaction is occurring between two other endpoints.

Note that the direction and locality changes with respect to the endpoint.

Consider an attack that involves a client exploiting a WIFF on a server.

For the attacker, the channel would be OUTGOING and INTERNAL.

For the server, the channel would be INCOMING and INTERNAL.

Consider an attack in which an FTP server exploits a buffer overflow by sending a long response to a request. The channel would be OUTGOING/INTERNAL for the server and INCOMING/INTERNAL for the victim.

Consider another case in which the attacker manipulates network traffic in a way that exploits a vulnerability in a sniffer. For the sniffer, the channel would be EXTERNAL/TRANSIENT as Monitor; for the attacker, the channel would be INTERNAL/OUTGOING as Outsider.

==========

VC.MULTCHAN. Multi-Channel Attacks

Vulnerabilities can be viewed in terms of the channels and endpoints that are involved.

Most vulnerabilities involve one channel - user-to-server over a network connection in remote cases, or user-to-user via a program execution in local cases.

Other vulnerabilities, or their associated attacks, are multi-channel.

Consider a buffer overflow involving reverse DNS. The attacker connects to a target web server from a particular IP address, then has a DNS server send a long response when the target performs reverse resolution to get the domain name for the IP address.

Step 1, Channel 1: attacker-as-user to service; INTERNAL/OUTGOING.

Step 2, Channel 2: service to attacker-as-consultant: INTERNAL/OUTGOING.

Step 3, Channel 2: attacker-as-consultant to service: INTERNAL/OUTGOING.

Consider a more complicated example involving cross-site scripting. XSS can involve two or three channels, with 3 endpoints.

Suppose there is a WIFF in which the attacker uses a web service to inject HTML onto a page that is then viewed by all users of that application.

The channels are:

  [*] (1) attacker-as-user to service
  [*] (2) service-to-user

When analyzing the service, the channels and attack steps are:

  [*] 1. Channel 1: attacker-as-user to service: INCOMING/INTERNAL
  [*] 2. Channel 2: service to user: OUTGOING/INTERNAL

When analyzing the user, the channels and attack steps are:

  [*] 1. Channel 2: service to user: INCOMING/INTERNAL

When analyzing the attacker, the channels and attack steps are:

  [*] 1. Channel 1: attacker-as-user to service: OUTGOING/INTERNAL

No other channels or steps are needed for the attacker before the WIFF is exploited.

Using the direction/locale model, one can see one reason why XSS is common: it is easy for the attacker to exploit, being single step and single channel, but on the server side, there are two separate channels involved.

Now, consider the classic cross-site scripting issue in which the attacker must force a user to click on a link while the user is interacting with the product. There are still two channels:

  (1) attacker-as-outsider to user

  (2) user-to-server

However, the number of steps, and the directionality or location, differ.

When analyzing the service, the channels and attack steps are:

  [*] 1. Channel 2: user to service: INCOMING/INTERNAL
  [*] 2. Channel 2: service to user: OUTGOING/INTERNAL

When analyzing the user, the channels and attack steps are:

  [*] 1. Channel 1: attacker-as-outsider to user: INCOMING/EXTERNAL
  [*] 2. Channel 2: user to service: OUTGOING/INTERNAL
  [*] 3. Channel 2: service to user: INCOMING/INTERNAL

When analyzing the attacker, the channels and attack steps are:

[*] 1. Channel 1: attacker-as-outsider to user: OUTGOING/EXTERNAL

There are some interesting observations here. First, the XSS attack is effectively launched by the user, in a trusted channel between the user and the server. This makes it more understandable why a programmer might allow an XSS problem in this context - users are not expected to attack themselves. Secondly, the attacker requires less interaction than any endpoint, and the attacker doesn't even need to use an internal channel with the product. This is another explanation for why XSS appears so frequently.

Note that the above scenario also describes Cross-Site Request Forgery (CSRF) attacks. The user is performing an action which, from the server perspective, is coming directly from the user.

Further study is needed to determine whether this concept is useful in identifying more complex vulnerabilities and attack scenarios.

==========

VC.MFV. Multi-Factor Vulnerabilities (MFV)

Vulnerabilities are often thought of as atomic entities. It is believed that there is a single fault in one place in the code (or its design), which opens one or more vectors for attack.

However, many vulnerabilities are really combinations of multiple factors or problems, which include WIFFs, attack channels, and manipulations. Removal of just one of the factors usually results in the elimination of the vulnerability, or at least a reduction in the attack surface.

Multi-Factor Vulnerabilities (MFVs) can be more complicated to prevent, find, and exploit than atomic vulnerabilities. They are also difficult to classify effectively, since currently available schemes treat vulnerabilities as if they are atomic. Understanding the role of multi-factor vulnerabilities is important in making improvements to existing terminology and classification.

The classic MFV is symbolic link following. Factors may include:

[*] permissions (the attacker must have access to a directory that the victim operates in; the product doesn't check the ownership of a file being written to)

[*] filename predictability (the attacker knows, or can predict, the
    name of the file that will be accessed)
[*] race condition
[*] design factor: lack of built-in support for safe temporary file
    creation in most programming languages, lack of atomic
    operations for effectively creating symlinks

Another common MFV is encoded path traversal. Consider the
application that protects itself against "../" strings, but not
against "%2e%2e%2fTARGET" strings.

PHP remote file include vulnerabilities are also multi-factor. The
product allows global variables to be modified, and the attacker must
interact over two separate channels (one to interact directly with the
product, and another to provide a malicious file). Unlike XSS,
however, the attacker must control a Consultant endpoint in order to
provide the malicious PHP file.

==========
SECTION.5. [TERMPROB] Problems with Existing Terminology
==========

Here are some of the problems with existing terminology in the
vulnerability world. Note: The heavily-discussed subjects like
"attack" versus "threat" versus "vulnerability" are avoided here.

1) The same term can be used to describe a WIFF, a manipulation, a
   vulnerability, or a consequence. "Buffer overflow" is the most
   obvious example. There are many WIFFs that can result in buffer
   overflows, such as format strings, off-by-one errors, integer
   signedness errors, and array index problems, not to mention the
   "classic" variants; however, they are all referred to as buffer
   overflows. At the same time, an attacker manipulation by crafting
   an extremely large input is not necessarily exploiting an overflow,
   although it may be called such. From an operational defensive
   standpoint, this distinction is usually immaterial; but for
   understanding vulnerabilities in terms of WIFFs, it is essential.

   Other multiple-use terms include "directory traversal,"
   "authentication bypass," "path disclosure," and many others.

2) Due to their nature, multi-factor vulnerabilities and multi-channel
   attacks do not usually have a single term. In addition, often a
   single term will be used for a multi-factor vulnerability, which
   obscures the true nature of the issue.

3) The same manipulation could be useful in attacks on multiple WIFFs, which can cause people to label it as if it is one WIFF, when it could be another. The "buffer overflow" by long input is one example; a product may treat a long input as if it is an invalid value, but poor error handling could trigger a crash. Another example occurs when an attacker provides a "-1" argument that causes a crash, it could be due to an integer overflow, a signedness error, or other factors. It is likely that overflows and signedness errors are frequently reported as the wrong bug type.

4) The same consequence can result from a broad range of WIFFs, but some problems are only described in terms of their consequence. The most egregious example, by far, is "denial of service," which can be triggered by a wide variety of WIFFs, but the term also covers a variety of consequences, some of which may be unimportant or irrelevant to a system administrator. For a less obvious example, consider a null dereference, which could be the result of a parsing error (due to a missing argument), a failed memory allocation (due to an integer signedness error), or a state machine violation (due to an inability to detect out-of-order steps).

5) Some terms refer to manipulations, but terms do not exist for the associated vulnerabilities, WIFFs, or consequences.

6) Some terms are used in different ways for multiple WIFFs. For example, the "leak" term can refer to the disclosure of information, an error in reclaiming used resources ("memory leak"), or inadvertently providing a trusted resource to an untrusted entity ("file descriptor leak").

SECTION.6. [DIAG] Diagnostic Errors and Challenges

Some diagnostic errors and challenges are covered in specific WIFF entries. Additional comments are below.

** DRAFT DRAFT DRAFT ** NOTE: this section has not been refined yet.

****

DIAG.PRES. Lack of distinction between primary and resultant errors

A large percentage of researcher reports focus only on the resultant errors form particular manipulations. The researcher does not perform sufficient diagnosis to identify the primary factors or to ensure that all elements of a multi-factor vulnerability are known and understood.

****

DIAG.MANIP. Role of manipulations in diagnostic errors

The same manipulation could be used in multiple WIFFs. The same WIFF could have multiple manipulations.

Diagnostic errors are likely to occur with manipulations that can trigger different faults. For example, a "long input" could trigger a buffer overflow, a null dereference due to an invalid value, an unhandled error condition, or other factors.

There is a diagnostic difficulty in distinguishing between integer errors. e.g. a "-1" input could lead to a signedness error or an integer overflow, but you can't label it as a signedness error just because a -1 was provided as the input. To make matters worse, sometimes a signedness error enables an integer overflow.

Also, manipulations in one data context (e.g. a "<" special character for XSS) could produce unrelated, resultant errors in another context which, if not diagnosed, do not detect a more serious underlying WIFF. For example: a SQL syntax error that's generated on a "<" XSS character injection could be an indicator of SQL injection.

****

DIAG.DOS. Insufficient diagnosis in "DoS" vulnerabilities

Most DoS vulnerabilities are not diagnosed to determine the associated WIFFs. The manipulations are often less structured, too. Thus, there is not much understanding of the underlying causes of DoS (i.e. the WIFFs).

Many vulnerabilities are described as crashes, which could be the result of infinite loops that cause memory allocation that eventually lead to an unhandled error condition, a null dereference, etc.

Some vulnerabilities that involve flooding attacks of large numbers

of connections are reported to cause a crash, but the crash could be resultant from overflows in arrays that are used to manage the connections.

The use of fuzzers and fault injection, while powerful technologies, make it easier for researchers cause product instability without needing to know what manipulations caused it, or why.

****

DIAG.OBSC. Surface-level diagnosis obscuring the real problem

Product-external error message infoleaks can be the source of many diagnostic errors. They are often simply reported as infoleaks when the underlying WIFF is more serious. In addition, a particular data manipulation could be directly inserted into the resulting message, which leads to a resultant WIFF. For example, an XSS manipulation might trigger a SQL syntax error that is product-external, and reflected directly back to the user with the XSS intact. This might be reported as XSS when it's really an indicator of some SQL problem.

Or, an XSS manipulation might cause a fault because the product cannot handle *any* invalid values, not just XSS, but the invalid value is used in an external uncontrolled error message and thus appears to be primary XSS.

It is highly suspected by the author that many "XSS" vulnerabilities in SQL-friendly PHP applications are actually resultant XSS, from resultant infoleaks of SQL injection vulns, where the SQL syntax error message is reflected back to the user.

XSS/SQL is being used as an example here, but there are other similar problems.

These diagnostic errors happen quite frequently, but sometimes researchers do not publish enough relevant details to know whether the vulnerability is resultant or not.

==========
SECTION.7. [HOT] Hypotheses, Observations, and Theories
==========

This is a very free-form collection of hypotheses, observations, and theories about vulnerabilities.

** DRAFT DRAFT DRAFT **

****

HOT.DOS. On "denial of service"

A key note: the phrase "denial of service" is often treated like it is a vulnerability. However, it is a CONSEQUENCE of the exploitation (or attempted exploitation) of a vulnerability. A variety of WIFFs can lead to a denial of service, and there are also many types of "denial of service". There is little research that tries to identify the underlying causes for "DoS."

****

HOT.IMM. On immutable vs. mutable

Some immutable data isn't critical (text color); but some critical data IS mutable (username upon login).

In some cases, an attacker can make something immutable and have an impact; e.g. changing perm's on a shared file so that others can't read it. This is not (currently) well-covered in PLOVER.

****

HOT.CLASS. On classification and taxonomies

Multi-factor vulnerabilities, by their nature, could fit into two or more separate categories, especially when they are multi-WIFF. Thus MFVs are good stress testers for any classification scheme.

****

HOT.COMPLEX. On vulnerability complexity

Theory: can vulnerability complexity - and/or attack complexity - be measured in terms of PLOVER concepts?

[*] number of attack channels the attacker has to control
[*] number of manipulations needing to be performed
[*] "popularity" of those manipulations (i.e. often are those manipulations publicly reported?)
[*] number of WIFFs necessary for exploit

[*] minimum number of inputs required as part of the attack
[*] environmental / operational constraints

NOTE that the attack complexity is different from the vuln complexity. For example, symlinks and PHP file inclusion are both multi-factor vulns, but the attacks are usually very simple. Buffer overflow involving an off-by-one error requiring a long hostname returned by DNS resolution is multi-channel but single WIFF.

Might want to cover multi-input attacks somehow, e.g. a path traversal where you have to provide 2 parameters, one for directory and one for filename. This is at least a little more complex.

****

HOT.VULNS. Thoughts on vulnerabilities

Notice how argument injection in process invocation is similar to attribute injection in XSS variants. Both involve whitespace and "arguments." Some SQL injection exploits require whitespace as well.

The exploitation of a vulnerability can involve the introduction of invalid, malformed, or inconsistent input in one context, which is valid, well-formed, and consistent in another context. One example is SQL injection. A counter-example is a vuln that results in DoS. Maybe this ONLY APPLIES to discuss vulns that cross security boundaries? DoS does NOT cross boundaries... or more precisely, it crosses different boundaries than code execution, privilege escalation, etc.

More vulnerabilities are multi-factor than you'd expect.

A product that's vulnerable to a single-factor issue is likely to be simultaneously vulnerable to multi-factor variants of that issue. For example, an application that blindly accepts "../" is likely to accept "%2e2e%2f" and so on. However, once a product begins to perform cleansing, the new manipulations could be invalid for an older version of the product.

Web browser vulnerabilities are often multi-factor, multi-input, multi-step, and/or multi-channel.

Some kinds of "intentional" infoleaks don't require any manipulations by the attacker, and the attacker only needs to have a Monitor role in a particular channel. This is not yet well-covered

in PLOVER.

The same vector or line of code can have multiple manipulations and WIFFs for completely different fault types. Consider the statement "open($filename)" in Perl.

Note how vulnerability "variants" often require different manipulations to exploit.

The role of context switching should be examined more closely; it is present in many vulns.

****

HOT.DESIMP. Design vs. Implementation

Thoughts on whether particular vulns are design vs. implementation - sometimes you can't tell without knowing developer's intentions! Also, some non-traditional implementation bugs are the result of failure to implement security mechanisms as required by the designs, e.g. "basic constraints" certificates.

Design decisions play a role in many vulnerabilities, if not all. This is especially the case with MFVs.

Programming language design plays a major role in WIFFs.

Theory: every implementation bug is multi-factor - at least one fault, which is effectively enabled by at least one design flaw or weakness. (hmmm need to rephrase this, but I know what I mean)

****

HOT.POP. Popularity of Some Vulnerabilities

Why are buffer overflows still so common today? New faults are discovered... multi-step and multi-input attacks are being found... new manipulations are being discovered.

Why is XSS so common? See the Alternate Channels sections for more specific details, but... It's multi-channel, so developers don't think of the attack. The exploit is single-path, so it's easy for researchers to find. There are many different manipulations that can bypass the more obvious protections, and at least some of the XSS that's reported is really resultant XSS instead of primary XSS, e.g. when an XSS manipulation triggers an SQL error due to invalid

syntax.

****

HOT.RESPRI. Resultant and Primary Vulns

A resultant vuln in one context could be primary in another. For example, suppose a researcher finds 2 issues. Issue 1 allows the attacker to gain extra privileges, but not administrator privileges. Then, in Issue 2, the attacker can then use those extra privileges to gain administrator privileges. Issue 2 is resultant from Issue 1, but it is also independent of it; if Issue 1 did not exist, then Issue 2 would still be a problem.

It would be very useful to identify the relationships between primary and resultant WIFFs. E.g. buffer overflow can be a resultant vuln of format string, signedness error, etc.; XSS is a resultant vuln of SQL injection if the manipulation contains XSS and the SQL engine generates an error.

****

HOT.STD. Standards vs. Non-standards

The lack of standards compliance is a MAJOR FACTOR in interaction errors, especially multiple interpretation errors. This makes the job of monitors and intermediaries extremely difficult to do correctly.

****

HOT.EVOL. Evolution of Security of a Product

This is based on observations.

Initial vulnerability reports for a product involve the most obvious entry points and the most obvious WIFFs, e.g. buffer overflows in username/pass, subject lines, etc., or basic "../" path traversal.

As the product matures, more complex manipulations, multiple manipulations, or alternate channels may be required.

Less obvious entry points are found. e.g. all the commands of a product have been tested; what about file format manipulations of the files that it processes?

The most mature, well-tested product is only subject to rare kinds of WIFFs, or new entirely classes of WIFFs.

****

HOT.CODE. On the State of Code Analysis

Code analysis technologies have different focuses.

- fuzzers seem focused on data manipulations, but not step manipulations

- code auditing tools are fault focused

****

HOT.BUFF. Buffer Overfows, Today and Yesterday

Most of yesterday's "classic" buffer overflows are single-input and single-channel. The attacker fills a single field with long input of any set of characters, the program blindly accepts the input, and it crashes or executes code.

Many of today's "classic" buffer overflows are multi-factor.

While classic "blind unbounded copy" buffer overflows still exist today, there are many multi-factor vulnerabilities today that are also referred to as "buffer overflows." One common MFV overflow involves the input field and a length field, in which the attacker modifies the length field and provides an input field whose actual length is inconsistent with the length field. Integer overflows can be one factor of an MFV in this scenario.

Another MFV overflow example is an off-by-one error that overwrites the terminating null character of a string, which effectively causes the string to be larger than expected, even when the programmer has otherwise kept very close track of string lengths. The factors involved here include the off-by-one error itself (possibly made easier by the design factor of 0-based vs. 1-based array indexing), the design factor in C of using terminator characters for strings, and the fault during execution, i.e. that a large input is copied into a small buffer.

Other MFV overflows can include "expansion-based" buffer overflows, in which the attacker provides special inputs that are translated into larger strings (think "&" to "&" in web applications), or

overflows that involve long sequences of special characters that cause the parser to lose track of where it is in the buffer that it is writing to.

Note that none of these MFV overflows are easily detectable using brute force black-box techniques. Each requires inputs that are more well-crafted than a long string of "A" characters followed by shellcode. This demonstrates how MFVs can have more complicated exploit scenarios, and it might explain why most MFV overflows are only found by the top researchers.

==========
SECTION.8. [GENESIS] Genesis of Vulnerabilities
==========

This section identifies specific phases of the software life cycle. Contrary to popular opinion, most vulnerabilities can be introduced during any of several phases. However, some vulnerabilities do tend to appear in one phase or another.

The phases include:

- [*] design
- [*] implementation
- [*] bundling
- [*] distribution
- [*] installation
- [*] configuration
- [*] documentation
- [*] patch
- [*] removal

****

GENESIS.DESIGN. Design

Note: this seems under-studied, especially with respect to classification of design flaws. Most "design limitations" or "design errors" are probably covered by other vulnerability categories. It is the author's belief that many implementation bugs are enabled by design flaws.

Common problems in this phase include:

- [*] introduction of many WIFFs

[*] failure to introduce design elements or patterns that minimize
the likelihood and risk of classes of implementation errors
(e.g. "use lookup table for valid values" to avoid special
character, MAID, and overflow errors)
[*] Incomplete specification, leading to interpretation errors,
[*] Vague specification, leading to multiple interpretation errors
[*] Lack of support for security-relevant options
[*] Required adherence to an insecure standard. For example, the
DOCSIS standard has certain design flaws, as does
IP/TCP/UDP/ICMP.

****

GENESIS.IMPLEMENTATION. Implementation

WIFFs in this phase are well-covered by PLOVER.

****

GENESIS.TESTING. Testing

Common problems in this phase include practices that make testing
more efficient:

[*] introducing back doors to facilitate testing. CVE-2002-1272 -
back door intended for development accidentally left enabled
in production
[*] leaving in debugging code. CVE-2001-0528 - debugging version
of DLL logs plaintext password. CVE-1999-0095 - debug command
in product left enabled
[*] using insecure configuration. CAN-2003-0983 - default
settings should have been disabled by the vendor, include a
user account and open TCP port

****

GENESIS.BUNDLING. Bundling Phase.

A product may have dependencies on third-party products or libraries
that need to be bundled or made available on the end system for
proper functioning.

Common problems in this phase include:

[*] The bundled product itself may have vulnerabilities.
Exploitation might require proxied channels through the main

product, or direct channels with the bundled product.
[*] There may be interaction errors between the main product and the bundled product, such as behavioral changes.

Examples:

CAN-2005-2385 - AV product uses a third-party library that contains directory traversal and buffer overflow issues

****

GENESIS.DISTRIB. Distribution

Common problems in this phase include:

[*] not undoing modifications from the testing phase (debugging code, back doors, insecure configuration)
[*] not providing a mechanism for integrity checking of the software. This is especially problematic for automatic download or update.
- CVE-2002-0671, CVE-2002-0676, CAN-2001-1125, CAN-2003-0237 - product downloads executables from a web site but does not verify integrity of the executables, allowing malicious injection using DNS spoofing
[*] introduction of embedded malicious code at the distribution point
- CAN-2002-1840 - backdoor in the configuration file of an IRC client downloaded from compromised site
- CAN-2002-2049 - configure compilation script modified at distribution point

****

GENESIS.INSTALL. Installation Phase

Common problems in this phase include:

[*] insecure permissions
[*] undeleted temporary files containing cleartext sensitive information
[*] WIFFs in the installation scripts themselves, e.g. symlink following in shell scripts

****

GENESIS.PATCH. Patch Error

Common problems in this phase include:

[*] regression error: an old vulnerability is introduced into new code
- CAN-2005-2158, CAN-2005-1937
- CAN-2002-1233 - regression error enables symlink
- CAN-2005-1649 - regression error of "Land" vulnerability (spoofed packet, self-referencing manipulation, infinite loop)

[*] overwrite of security patch with older patch
- CAN-2002-1670 - upgrade overwrites previous security-relevant patches

[*] interaction errors with other patches

[*] overwrite of configuration to less secure options

[*] WIFFs that arise from the patching process itself

[*] incomplete vulnerability fix. Typically this involves fixing a specific WIFF but not considering other manipulations, alternate channels, etc.
- CAN-2005-0206 - incomplete patch misses 64-bit architecture

[*] other errors
- CVE-1999-1047 - patches applied in a particular sequence allows firewall bypass and does not log events

****

GENESIS.DOC. Documentation Error

Common problems in this phase include:

[*] Omission of security-critical information

[*] Error/typo causes user to introduce a vulnerability or risk

[*] Specific recommendation of insecure practices

****

GENESIS.PORT. Porting

A product may be ported to a different environment (e.g. OS, language, or hardware platform). The product must consider differences with the original environment, otherwise vulnerabilities may be introduced that are specific to the new environment.

For example, a product that was originally developed and secured on Unix could be ported to a Windows platform and become subject to very basic Windows-specific bugs, e.g. directory traversal using

"\" instead of "/". The reverse is also true, of course, although examples are not immediately available.

Common type of ports are:

[*] port to different OS
[*] port to different hardware / architecture (e.g. chip)
[*] port to different programming language
[*] port from single-user to multi-user
[*] port from non-networked to networked

****

GENESIS.CONFIG. Configuration

Note: configuration errors are vastly under-studied, especially in terms of classification. They can be more complex than vulnerabilities, which are often discrete and easily separable. In addition, configuration overlaps with the general area of "policy," which can have elements that are not always considered to be relevant to security.

Common configuration problems include:

[*] Default password
[*] Default, non-essential service or component
[*] Default less-secure operating mode
[*] Administrator capability accessible to arbitrary hosts

****

GENESIS.OPENV. Genesis - Operating Environment

The product might be deployed into an operating environment or context that violates its most basic assumptions, introducing entire classes of WIFFs that were not previously relevant. For example, a program designed for local users might be called from a CGI wrapper, thus rendering all inputs under possible control by an untrusted party.

Common operating environment changes are:

[*] make program setuid
[*] port local program to networked
[*] single-user to multi-user environment

SECTION.9. [WIFF] WIFFs: Weaknesses, Idiosyncrasies, Faults, Flaws

The bulk of this document covers a large variety of WIFFs, with a large number of real-world vulnerability examples.

The order of presentation, and the categorization implied by the different sections, is not intended to be authoritative.

Each WIFF attempts to include a definition, notes on terminology, research gaps, common overlap with other WIFFs, and other information. Two or three examples are provided for each WIFF. For many WIFFs, an appendix lists additional examples that further illustrate the subtlety and variety of vulnerabilities. Multi-factor examples may be included.

The examples use CVE identifiers (CVE-yyyy-nnnn or CAN-yyyy-nnnn) for specific vulnerabilities that demonstrate the given category. The identifiers can be accessed from the search form at http://cve.mitre.org/cve

Following is a summary of the main categories.

[BUFF] Buffer overflows, format strings, etc.

> Buffer Boundary Violations ("buffer overflow"), Unbounded Transfer ("classic overflow"), Boundary beginning violation ("buffer underflow" ?), Out-of-bounds Read, Buffer over-read, Buffer under-read, Array index overflow, Length Parameter Inconsistency, Other length calculation error, Format string vulnerability

[SVM] Structure and Validity Problems

> Missing Value Error, Missing Parameter Error, Missing Element Error, Extra Value Error, Extra Parameter Error, Undefined Parameter Error, Undefined Value Error, Wrong Data Type, Incomplete Element, Inconsistent Elements

[SPEC] Special Elements (Characters or Reserved Words)

> General Special Element Problems, Parameter Delimiter, Value Delimiter, Record Delimiter, Line Delimiter, Section Delimiter, Input Terminator, Input Leader, Quoting Element, Escape, Meta, or

Control Character / Sequence, Comment Element, Macro Symbol, Substitution Character, Variable Name Delimiter, Wildcard or Matching Element, Whitespace, Grouping Element / Paired Delimiter, Delimiter between Expressions or Commands, Null Character / Null Byte

[SPECM] Common Special Element Manipulations

Special Element Injection, Equivalent Special Element Injection, Leading Special Element, Multiple Leading Special Elements, Trailing Special Element, Multiple Trailing Special Elements, Internal Special Element, Multiple Internal Special Element, Missing Special Element, Extra Special Element, Inconsistent Special Elements

[SPECTS] Technology-Specific Special Elements

Cross-site scripting (XSS), Basic XSS, XSS in error pages, Script in IMG tags, XSS using Script in Attributes, XSS using Script Via Encoded URI Schemes, Doubled character XSS manipulations, e.g. "<<script", Null Characters in Tags, Alternate XSS syntax, OS Command Injection, Argument Injection or Modification, SQL injection, LDAP injection, XML injection (aka Blind Xpath injection), Custom Special Character Injection, CRLF Injection, Improper Null Character Termination

[PATH] Path Traversal and Equivalence Errors

Path Traversal, Relative Path Traversal, "../filedir", "/../filedir", "/directory/../filename", "directory/../../filename", "..\filename" ("dot dot backslash"), "\..\filename" ("leading dot dot backslash"), "\directory\..\filename", "directory\..\..\filename", "..." (triple dot), "...." (multiple dot), "....//" (doubled dot dot slash), ".../...//", Absolute Path Traversal, /absolute/pathname/here, \absolute\pathname\here ("backslash absolute path"), "C:dirname" or C: (Windows volume or "drive letter"), "\\UNC\share\name\" (Windows UNC share), Path Equivalence, Trailing Dot - "filedir.", Internal Dot - "file.ordir", Multiple Internal Dot - "file...dir", Multiple Trailing Dot - "filedir....", Trailing Space - "filedir ", Leading Space - " filedir", file[SPACE]name (internal space), filedir/ (trailing slash, trailing /), //multiple/leading/slash ("multiple leading slash"), /multiple//internal/slash ("multiple internal slash"), /multiple/trailing/slash// ("multiple trailing slash"), \multiple\\internal\backslash, filedir\ (trailing

backslash), /./ (single dot directory), filedir* (asterisk / wildcard), dirname/fakechild/../realchild/filename, Windows 8.3 Filename, Link Following, UNIX symbolic link (symlink) following, UNIX hard link, Windows Shortcut Following (.LNK), Windows hard link, Virtual Files, Windows MS-DOS device names, Windows ::DATA alternate data stream, Apple ".DS_Store", Apple HFS+ alternate data stream

[CP] Channel and Path Errors

Channel Errors, Unprotected Primary Channel, Unprotected Alternate Channel, Alternate Channel Race Condition, Proxied Trusted Channel, Unprotected Windows Messaging Channel ("Shatter"), Alternate Path Errors, Direct Request aka "Forced Browsing", Miscellaneous alternate path errors, Untrusted Search Path, Uncontrolled Search Path Element, Unquoted Search Path or Element

[CCC] Cleansing, Canonicalization, and Comparison Errors

Encoding Error, Alternate Encoding, Double Encoding, Mixed Encoding, Unicode Encoding, URL Encoding (Hex Encoding), Case Sensitivity (lowercase, uppercase, mixed case), Early Validation Errors, Validate-Before-Canonicalize, Validate-Before-Filter, Collapse of Data into Unsafe Value, Permissive Whitelist, Incomplete Blacklist, Regular Expression Error, Overly Restrictive Regular Expression, Partial Comparison

[INFO] Information Management Errors

Information Leak (information disclosure), Discrepancy Information Leaks, Response discrepancy infoleak, Behavioral Discrepancy Infoleak, Internal behavioral inconsistency infoleak, External behavioral inconsistency infoleak, Timing discrepancy infoleak, Product-Generated Error Message Infoleak, Product-External Error Message Infoleak, Cross-Boundary Cleansing Infoleak, Intended information leak, Process information infoleak to other processes, Infoleak Using Debug Information, Sensitive Information Uncleared Before Use, Sensitive memory uncleared by compiler optimization, Information loss or omission, Truncation of Security-relevant Information, Omission of Security-relevant Information, Obscured Security-relevant Information by Alternate Name

[RACE] Race Conditions

Race condition enabling link following, Signal handler race condition, Time-of-check Time-of-use race condition, Context Switching Race Condition, Alternate Channel Race Condition, Other race conditions

[PPA] Permissions, Privileges, and ACLs

Privilege / sandbox errors, Incorrect Privilege Assignment, Unsafe Privilege, Privilege Chaining, Privilege Management Error, Privilege Context Switching Error, Privilege Dropping / Lowering Errors, Insufficient privileges, Misc. privilege issues, Permission errors, Insecure Default Permissions, Insecure inherited permissions, Insecure preserved inherited permissions, Insecure execution-assigned permissions, Fails poorly due to insufficient permissions, Permission preservation failure, Ownership errors, Unverified Ownership, Access Control List (ACL) errors, User management errors

[HAND] Handler Errors

Handler errors, Missing Handler, Dangerous handler not cleared/disabled during sensitive, Unparsed Raw Web Content Delivery, Unrestricted File Upload

[UI] User Interface Errors

Product UI does not warn user of unsafe actions, Insufficient UI warning of dangerous operations, User interface inconsistency, Unimplemented or unsupported feature in UI, Obsolete feature in UI, The UI performs the wrong action, Multiple Interpretations of UI Input, UI Misrepresentation of Critical Information

[INT] Interaction Errors

Multiple Interpretation Error (MIE), Extra Unhandled Features, Behavioral Change, Expected behavior violation, Unintended proxy/intermediary, HTTP response splitting, HTTP Request Smuggling

[INIT] Initialization and Cleanup Errors

Insecure default variable initialization, External initialization of trusted variables or values, Non-exit on Failed Initialization, Missing Initialization, Incorrect initialization, Incomplete Cleanup

[RES] Resource Management Errors

Memory leak, Resource leaks, UNIX file descriptor leak, Improper resource shutdown, Asymmetric resource consumption (amplification), Network Amplification, Algorithmic Complexity, Data Amplification, Insufficient Resource Pool, Insufficient Locking, Missing Lock Check

[NUM] Numeric Errors

Off-by-one Error, Integer Signedness Error (aka "signed integer" error), Integer overflow (wrap or wraparound), Integer underflow (wrap or wraparound), Numeric truncation error, Numeric Byte Ordering Error

[AUTHENT] Authentication Error

Authentication Bypass by Alternate Path/Channel, Authentication bypass by alternate name, Authentication bypass by spoofing, Authentication bypass by replay, Man-in-the-middle (MITM), Authentication Bypass via Assumed-Immutable Data, Authentication Logic Error, Missing Critical Step in Authentication, Authentication Bypass by Primary WIFF, No Authentication for Critical Function, Multiple Failed Authentication Attempts not Prevented, Miscellaneous Authentication Errors

[CRYPTO] Cryptographic errors

Plaintext Storage of Sensitive Information, Plaintext Storage in File or on Disk, Plaintext Storage in Registry, Plaintext Storage in Cookie, Plaintext Storage in Memory, Plaintext Storage in GUI, Plaintext Storage in Executable, Plaintext Transmission of Sensitive Information, Key Management Errors, Missing Required Cryptographic Step, Weak Encryption, Reversible One-Way Hash, Miscellaneous Crypto Problems

[RAND] Randomness and Predictability

Insufficient Entropy, Small Space of Random Values, PRNG Seed Error, Same Seed in PRNG, Predictable Seed in PRNG, Small Seed Space in PRNG, Predictable from Observable State, Predictable Exact Value from Previous Values, Predictable Value Range from Previous Values

[CODE] Code Evaluation and Injection

Direct Dynamic Code Evaluation ("Eval Injection"), Direct Static Code Injection, Server-Side Includes (SSI) Injection, PHP File Inclusion

[ERS] Error Conditions, Return Values, Status Codes

Unchecked Error Condition, Missing Error Status Code, Wrong Status Code, Unexpected Status Code or Return Value

[VER] Insufficient Verification of Data

Improperly Verified Signature, Use of Less Trusted Source, Untrusted Data Appended with Trusted Data, Improperly Trusted Reverse DNS, Insufficient Type Distinction, Cross-Site Request Forgery (CSRF), Other Insufficient Verification

[MAID] Modification of Assumed-Immutable Data

Web Parameter Tampering, PHP External Variable Modification

[MAL] Product-Embedded Malicious Code

Back Door, Back Door, Developer-Introduced Back Door, Outsider-Introduced Back Door, Hidden User-Triggered Functionality, Logic Bomb, Time Bomb

[ATTMIT] Common Attack Mitigation Failures

Insufficient Replay Protection, Susceptibility to Brute Force Attack, Susceptibility to Spoofing

[CONT] Containment errors (container errors)

Sensitive Entity in Accessible Container, Sensitive Data Under Web Root, Sensitive Data Under FTP Root

[MISC] Miscellaneous WIFFs

Double-Free Vulnerability, Incomplete Internal State Distinction, Other Types of Truncation Errors, Signal Errors, Improperly Implemented Security Check for Standard, Misinterpretation Error, Business Rule Violations or Logic Errors

==========

SECTION.9.1. [BUFF] Buffer overflows, format strings, etc.

==========

Note: while buffer overflows have been widely studied, a large number of related WIFFs have been discovered. A more systematic analysis of overflows - and related out-of-bounds buffer operations - is needed.

****

BUFF. Buffer Boundary Violations ("buffer overflow")

Functional Area: memory management

Terminology Note: some prominent vendors and researchers use the term "buffer overrun," but most people use "buffer overflow."

Terminology Note: many issues that are now called "buffer overflows" are substantively different than the "classic" overflow, including entirely different bug types that rely on overflow exploit techniques, such as integer signedness errors, integer overflows, and format string bugs. This imprecise terminology can make it difficult to determine which variant is being reported.

Note: this checklist does not distinguish between stack-based and heap-based buffer overflows, which may require different discovery and exploit techniques, but they are not inherently different from a programming perspective.

****

BUFF.OVER. Unbounded Transfer ("classic overflow")

Note: at the programmer level, stack-based and heap-based overflows do not differ significantly, so they are not distinguished here. Obviously, from the exploit perspective using shellcode, they can be quite different.

CVE-2000-1094 - buffer overflow using command with long argument

CVE-1999-0046 - buffer overflow in local program using long environment variable

CVE-2002-1337 - buffer overflow in comment characters, when product increments a counter for a ">" but does not decrement for "<"

****

BUFF.UNDER. Boundary beginning violation ("buffer underflow" ?)

   Definition: the product writes at least part of the data before the
   beginning of the buffer that it intended to write.

   Note: this could be resultant from several errors, including a bad
   offset or an array index that decrements before the beginning of
   the buffer (see array index overflows).

   Terminology Note: some prominent vendors and researchers use the
   term "buffer underrun".

   Note: this term has probably been used for multiple issue types;
   the concept seems understudied

   Reference:

   Ref: VULN-DEV:20040110 Buffer UNDERFLOWS: What do you know about it?

   Examples:

   BUGTRAQ:20020911 Buffer over/underflows in ssldump prior to 0.9b3

   CAN-2004-1176

   CAN-2003-0082

   CVE-2004-2620 - buffer underflow due to mishandled special chars

****

BUFF.READ. Out-of-bounds Read

   Definition: the product reads data past the end, or before the
   beginning, of the intended buffer.

   Note: needs study.

   Terninology Note: some people say "read buffer overflow", but the
   "overflow" term might be so closely associated with "write" that
   this term could cause confusion. An alternate term might be

"buffer over-read"

Research Gaps: under-studied and under-reported. Most issues are probably labeled as buffer overflows.

CAN-2004-0112 - out-of-bounds read due to improper length check

CAN-2004-0183, CAN-2004-0221 - packet with large number of specified elements cause out-of-bounds read.

CAN-2004-0184 - out-of-bounds read, resultant from integer underflow

CAN-2004-1940 - large length value causes out-of-bounds read

CAN-2004-0421 - malformed image causes out-of-bounds read

****

BUFF.OREAD. Buffer over-read

Definition: the product reads data past the end of the intended buffer.

****

BUFF.UREAD. Buffer under-read

Definition: the product reads data past the end of the intended buffer.

Note: needs study.

Research Gaps: under-studied.

****

BUFF.INDEX. Array index overflow

Definition: a buffer overflow or underflow that occurs when an attacker-influenced value is used as an array index.

Alternate terms: "out-of-bounds array index" or "index-out-of-range" or "array index underflow"

Overlaps: integer signedness errors, parameter tampering,

out-of-bounds read, overflows.

Note: a single fault could allow both an overflow and underflow of the array index.

Note: an index overflow exploit might use buffer overflow techniques, but this can often be exploited without having to provide "large inputs."

Note: array index overflows can also trigger out-of-bounds read operations, or operations on the wrong objects; i.e., "buffer overflows" are not always the result.

Examples:

CAN-2005-0369 - large ID in packet used as array index

CAN-2001-1009 - negative array index as argument to POP LIST command

CAN-2003-0721 - Integer signedness error leads to negative array index

CAN-2004-1189 - product does not properly track a count and a maximum number, which can lead to resultant array index overflow.

****

BUFF.LEN. Length Parameter Inconsistency

Alternate Term: length manipulation, length tampering

Definition: the attacker can manipulate the length parameter associated with an input so that it is inconsistent with the actual length of the input.

Note: probably overlaps other categories

Note: can overlap zero-length issues (CVE-2001-0825)

Examples:

CVE-2001-1186, CVE-2001-0191, CAN-2003-0429, CVE-2000-0655, CAN-2004-0492, CAN-2004-0201, CVE-2003-0825, CVE-2004-0095, CAN-2004-0826, CAN-2004-0808, CAN-2004-0808, CAN-2002-1357, CAN-2004-0774, CAN-2004-0940, CAN-2004-0989, CAN-2004-0568, CAN-2003-0327, CAN-2003-0345, CAN-2004-0430, CAN-2005-0064, others

CAN-2004-0413 - leads to memory consumption, integer overflow, and heap overflow

CAN-2004-0940 is effectively an accidental double increment of a counter that prevents a length check conditional from exiting a loop.

CAN-2002-1235 - length field of a request not verified

CVE-2005-3184 - buffer overflow by modifying a length value

SECUNIA:18747 - length field inconsistency crashes cell phone

****

BUFF.LENCALC. Other length calculation error

Note: this is a broad category. Some examples include: (1) simple math errors, (2) incorrectly updating parallel counters, (3) not accounting for size differences when "transforming" one input to another format (e.g. URL canonicalization or other transformation that can generate a result that's larger than the original input, i.e. "expansion").

Note: this level of detail is rarely available in public reports, so it is difficult to find good examples.

Examples:

substitution overflow:

CAN-2004-1363 - buffer overflow using environment variables that are expanded after the length check is performed

CAN-2004-0747 - buffer overflow using expansion of environment variables

CAN-2005-2103 - buffer overflow using a large number of substitution strings

transformation overflow:

CAN-2005-3120 - product adds extra escape characters to incoming data, but does not account for them in the buffer length (or is

this expansion?)

CAN-2003-0899 - buffer overflow when expanding ">" to ">", etc.

expansion overflow:

CVE-2001-0334 - buffer overflow using wildcards

CAN-2001-0248, CAN-2001-0249 - long pathname + glob = overflow

others:

CVE-2002-0184 - special characters in argument are not properly expanded (needs closer investigation to determine if substitution, expansion, or transformation)

CAN-2004-0434 - small length value leads to heap overflow

CAN-2004-0940

CAN-2002-1347 - multiple variants

CAN-2005-0490 - needs closer investigation, but probably expansion-based

==========

BUFF.FORMAT. Format string vulnerability

Functional Area: logging, errors, general output

Frequently targeted entities: file names, process names, identifiers

Reference: [Newsham]

Research gaps: format string issues are under-studied for languages other than C. Memory or disk consumption, control flow or variable alteration, and data corruption may result from format string exploitation in applications written in other languages such as Perl, PHP, Python, etc.

Research Gaps: since format strings often occur in rarely-occurring erroneous conditions, it is highly that many latent issues exist in executables that do not have associated source code (or equivalent source).

Examples:

CAN-2002-1825 - format string in Perl program

CVE-2001-0717, CVE-2002-0573 - format string in bad call to syslog function

CAN-2002-1788 - format strings in NNTP server responses

SECTION.9.2. [SVM] Structure and Validity Problems

These problems are known more by their manipulations, or their consequences, than the underlying WIFFs. Terminology does not exist for most of these WIFFs.

Functional Area: non-specific, parsing

Research Gaps: the general problem of "malformed input" is under-studied from the standpoint of underlying programming errors. Most efforts have been in developing attack methods, which rarely suggest the nature of the underlying error. Attack-related research in this type of "malformed input" is scattered but ongoing, e.g. see fuzzers, suite-based testing (PROTOS style), and fault injection. The effect of these is often a denial of service, although other impacts may be under-studied.

Diagnosis: the specific underlying cause is rarely diagnosed by the researcher, although diagnosis is not always feasible with available time or resources. This is especially problematic when researchers report that "a number of random inputs were provided, which led to a crash."

Note: this can overlap with special character mismanagement, and it probably needs some more precise and well-defined sub-categories than the ones listed below.

****

SVM.VAL.MISS. Missing Value Error

Definition: the product does not handle when a parameter, field, or argument name is specified, but the associated value is missing,

i.e. it is empty, blank, or null.

Note: some "crash by port scan" bugs are probably due to this, but lack of diagnosis makes it difficult to be certain.

CAN-2002-0422 - blank Host header triggers resultant infoleak

CVE-2000-1006 - blank "charset" attribute in MIME header triggers crash

CAN-2004-1504, CAN-2005-2053 - blank parameter causes external error infoleak

****

SVM.PAR.MISS. Missing Parameter Error

Definition: the product does not handle when a parameter, field, or argument name is not specified. Typically, the element is either required or frequently specified.

Examples:

CVE-2004-0276, CAN-2002-1488, CVE-2002-1169, CVE-2000-0521, CVE-2001-0590, CVE-2002-1236, CAN-2003-0239, CAN-2003-0477, CAN-2003-0422, CVE-2002-1531, CAN-2002-1077, CAN-2002-1358, CAN-2002-1023

CVE-2002-1236, CAN-2003-0422 - CGI crashes when called without any arguments

CVE-2002-1531, CAN-2002-1077 - crash in HTTP request without a Content-Length field

CAN-2002-1358 - empty elements/strings in protocol test suite affect many SSH2 servers/clients

CAN-2003-0477 - FTP server crashes in PORT command without an argument

CVE-2002-0107 - resultant infoleak in web server via GET requests without HTTP/1.0 version string

CAN-2002-0596 - GET reqeust with empty parameter leads to error message infoleak (path disclosure)

****

SVM.ELT.MISS. Missing Element Error

  Definition: the product does not handle when an expected element is
  not provided.

  Note: can overlap other problems.

****

SVM.VAL.EXT. Extra Value Error

  Definition: the product does not handle when more values are
  specified than expected.

  Note: This typically occurs in situations when only one value is
  expected.

  Note: this can overlap buffer overflows.

****

SVM.PAR.EXT. Extra Parameter Error

  Definition: the product does not handle when a parameter, field, or
  argument name is specified two or more times.

  Note: This typically occurs in situations when only one element is
  expected to be specified.

  Note: this type of problem has a big role in multiple interpretation
  vulnerabilities and various HTTP attacks.

  CAN-2003-1014 - MIE. multiple gateway/security products allow
  restriction bypass using multiple MIME fields with the same name,
  which are interpreted differently by clients.

****

SVM.PAR.UNDEF. Undefined Parameter Error

  Definition: the product does not handle when a parameter, field, or
  argument name is not defined or supported by the product.

Examples:

CVE-2001-0650

CAN-2002-1488 - crash in IRC client via PART message from a channel the user is not in

CVE-2001-0650 - router crash or bad route modification using BGP updates with invalid transitive attribute

****

SVM.VAL.UNDEF. Undefined Value Error

Definition: the product does not handle when a value is not defined or supported for the associated parameter, field, or argument name.

CVE-2000-1003 - client crash when server returns unknown driver type

****

SVM.WTYPE. Wrong Data Type

Definition: the application does not properly handle when a particular element is of the wrong type, e.g. it expects a digit (0-9) but is provided with a letter (A-Z).

Research gaps: probably under-studied.

CVE-1999-1156 - FTP server crash via PORT command with non-numeric character

CVE-2004-0270 - anti-virus product has assert error when line length is non-numeric

****

SVM.INCOMP. Incomplete Element

Definition: the application does not properly handle when a particular element is not completely specified.

Note: overlaps incomplete resource release

Examples:

CVE-2002-1532, CAN-2003-0195

CVE-2002-1532 - HTTP GET without \r\n\r\n CRLF sequences causes product to wait indefinitely and prevents other users from accessing it

CAN-2003-0195 - partial request is not timed out

CAN-2005-2526 - MFV. CPU exhaustion in printer via partial printing request then early termination of connection.

CVE-2002-1906 - CPU consumption by sending incomplete HTTP requests and leaving the connections open.

****

SVM.INC. Inconsistent Elements

Definition: the product does not handle when multiple parameters, fields, arguments, or values should be consistent, but are not.

Note: this can overlap other WIFFs such as Length Parameter Inconsistency.

==========

SECTION.9.3. [SPEC] Special Elements (Characters or Reserved Words)

==========

This section deals with various problems that involve special elements such as special characters and reserved words.

Note that special characters, or reserved words, can have varying functions depending on the context. For example, a double quote could be a comment in one context and an escape in another; or a semicolon could be a field separator or a value separator.

Research Gaps: while much research has been conducted on special characters, reserved/special words are under-studied.

****

SPEC.GEN. General Special Element Problems

Every language has its own special elements such as characters and reserved words. The following WIFFs are expressed in general terms.

Technology-specific problems that involve special elements, such as cross-site scripting and SQL injection, are covered in another section.

Functional Area: non-specific, parsing

Note: some of these types of special chars have been observed at one point or another, but it's difficult to find suitable examples after the fact. In an attempt to be complete about what kinds of "special characters" exist, some types may have been added to this list without any publicly reported vulnerability for those types.

****

SPEC.DELIM.PARAM. Parameter Delimiter

Examples:

CAN-2003-0307 - attacker inserts field separator into input to specify admin privs

****

SPEC.DELIM.VAL. Value Delimiter

CAN-2000-0293 - multiple internal space, insufficient quoting - program does not use proper delimiter between values

****

SPEC.DELIM.REC. Record Delimiter

CAN-2004-1982 - carriage returns in subject field allow adding new records to data file

CVE-2001-0527 - attacker inserts carriage returns and "|" field separator characters to add new user/privileges

****

SPEC.DELIM.LINE. Line Delimiter

Note: CRLF injection is covered in the tech-specific section.

Note: depending on the language and syntax being used, this could be the same as the record delimiter.

CVE-2002-0267 - linebreak in field of PHP script allows admin privileges when written to data file

****

SPEC.DELIM.SECTION. Section Delimiter

One example of a section delimiter is the boundary string in a multipart MIME message. In many cases, doubled line delimiters can serve as a section delimiter.

Note: CRLF injection is covered in the tech-specific section.

Note: depending on the language and syntax being used, this could be the same as the record delimiter.

****

SPEC.INPTERM. Input Terminator

Example: a "." in SMTP signifies the end of mail message data, whereas a null character can be used for the end of a string.

CVE-2000-0319, CVE-2000-0320 - MFV. mail server does not properly identify terminator string to signify end of message, causing corruption, possibly in conjunction with off-by-one error.

CAN-2001-0996 - mail server does not quote end-of-input terminator if it appears in the middle of a message.

CAN-2002-0001 - improperly terminated comment or phrase allows commands.

****

SPEC.INPLEAD. Input Leader

****

SPEC.QUOTE. Quoting Element

Examples: CAN-2003-1016, CAN-2004-0956

CAN-2003-1016 - MIE. MFV too? bypass AV/security with fields that should not be quoted, duplicate quotes, missing leading/trailing

quotes.

****

SPEC.ESCAPE. Escape, Meta, or Control Character / Sequence

CVE-2002-0542 - mail program handles special "~" escape sequence even when not in interactive mode.

CVE-2000-0703 - setuid program does not filter escape sequences before calling mail program

CVE-2002-0986 - mail function does not filter control characters from arguments, allowing mail message content to be modified

CVE-2003-0020, CAN-2003-0083 - Multi-channel issue. Terminal escape sequences not filtered from log files

CVE-2003-0021, CVE-2003-0022, CVE-2003-0023, CVE-2003-0063, CAN-2000-0476 - terminal escape sequences not filtered by terminals when displaying files

CAN-2001-1556 - MFV. (multi-channel). Injection of control characters into log files that allow information hiding when using raw Unix programs to read the files.

****

SPEC.COMMENT. Comment Element

CAN-2002-0001 - mail client command execution due to improperly terminated comment in address list

CAN-2004-0162 - MIE. RFC822 comment fields may be processed as other fields by clients.

CAN-2004-1686 - well-placed comment bypasses security warning

CAN-2005-1909, CAN-2005-1969 - information hiding using a manipulation involving injection of comment code into product. Note: these vulns are likely vulnerable to more general XSS problems, although a regexp might allow "<!--" while denying most other tags.

****

SPEC.MACRO. Macro Symbol

   Research Gaps: under-studied.

   CAN-2002-0770 - server trusts client to expand macros, allows macro
   characters to be expanded to trigger resultant infoleak.

****

SPEC.SUBST. Substitution Character

   Research Gaps: under-studied.

   CAN-2002-0770 - server trusts client to expand macros, allows macro
   characters to be expanded to trigger resultant infoleak.

****

SPEC.VARNAME. Variable Name Delimiter

  Example: "$" for an environment variable.

  Research Gaps: under-studied.

  CAN-2005-0129 - "%" variable is expanded by wildcard function into
  disallowed commands

  CAN-2002-0770 - server trusts client to expand macros, allows macro
  characters to be expanded to trigger resultant infoleak.

****

SPEC.WILDCARD. Wildcard or Matching Element

   Research Gaps: under-studied.

   CAN-2002-0433, CAN-2002-1010 - bypass file restrictions using
   wildcard character

   CVE-2001-0334 - wildcards generate long string on expansion

   CAN-2004-1962 - SQL injection involving "/**/" sequences

****

SPEC.WHITESPACE. Whitespace

Alternate Term: white space

Note: this can include space, tab, etc.

Note: can overlap other separator characters or delimiters

Examples:

CAN-2002-0637 - MIE. virus protection bypass with RFC violations involving extra whitespace, or missing whitespace.

CAN-2004-0942 - CPU consumption with MIME headers containing lines with many space characters, probably due to algorithmic complexity (RESOURCE.AMP.ALG).

CAN-2003-1015 - MIE. whitespace interpreted differently by mail clients.

****

SPEC.GROUPING. Grouping Element / Paired Delimiter

Description: does not properly handle the characters that are used to mark the beginning and ending of a group of entities, such as parentheses, brackets, and braces.

Research Gaps: under-studied.

Examples:

[*] "<" and ">" angle brackets
[*] "(" and ")" parentheses
[*] "{" and "}" braces
[*] "[" and "]" square brackets
[*] '"' and '"' double quotes
[*] "'" and "'" single quotes

CAN-2004-0956 - crash via missing paired delimiter (open double-quote but no closing double-quote)

CVE-2000-1165 - crash via message without closing ">"

CVE-2005-2933 - buffer overflow via malbox name with an opening double quote but missing a closing double quote, causing a larger copy than expected

****

SPEC.DELIM.EXPR. Delimiter between Expressions or Commands

  Note: shell metacharacters (covered elsewhere) is one example.

****

SPEC.NULL. Null Character / Null Byte

   Note: this can be a factor in multiple interpretation errors, other
   interaction errors, filename equivalence, etc.

   CAN-2005-2008, CVE-2005-3293 - source code disclosure using trailing
null

   CAN-2005-2061 - trailing null allows file include

   CAN-2002-1774 - null character in MIME header allows detection bypass

    CVE-2000-0149, CVE-2000-0671, CVE-2001-0738, CAN-2001-1140

    CAN-2002-1031, CAN-2002-1025

    CAN-2003-0768

    CVE-2004-0189 - decoding function in proxy allows regular
    expression bypass in ACLs via URLs with null characters

  CVE-2005-3153, CVE-2005-4155 - null byte bypasses PHP regexp check
  (interaction error)

==========
SECTION.9.4. [SPECM] Common Special Element Manipulations
==========

The variety of manipulations that involve special elements is
staggering. This is one reason why they are so frequently reported.

Research Gaps: Customized languages and grammars, even those that are
specific to a particular product, are potential sources of WIFFs that
are related to special elements. However, most researchers
concentrate on the most commonly used representations for data
transmission, such as HTML and SQL. Any representation that is

commonly used is likely to be a rich source of WIFFs; researchers are encouraged to investigate previously unexplored representations.

As previously discussed, precise terminology for the underlying WIFFs does not exist. Therefore, these WIFFs use the terminology associated with the manipulation.

WIFFs involving special characters do not always require special manipulations besides injection of the special character, but sometimes they do.

Note: This list is FAR from complete.

****

SPECM.INJ. Special Element Injection

This is the most common type of WIFF.

****

SPECM.INJ.EQ. Equivalent Special Element Injection

This WIFF frequently occurs when the product has protected itself against special element injection.

Note: can include encoded special characters.

****

SPECM.LEADING. Leading Special Element

****

SPECM.LEADING.MULT. Multiple Leading Special Elements

****

SPECM.TRAILING. Trailing Special Element

****

SPECM.TRAILING.MULT. Multiple Trailing Special Elements

****

SPECM.INTERNAL. Internal Special Element

****

SPECM.INTERNAL.MULT. Multiple Internal Special Element

****

SPECM.MISS. Missing Special Element

  Definition: the product does not handle when a special element
  (character or reserved word) is missing.

  Note: this can overlap incomplete input.

  CVE-2002-1362 - crash via message type without separator character

  CVE-2002-0729 - missing special character (separator) causes crash

  CVE-2002-1532 - HTTP GET without \r\n\r\n CRLF sequences causes
  product to wait indefinitely and prevents other users from accessing
  it

****

SPECM.EXTRA. Extra Special Element

  Definition: the product does not handle when a special element
  (character or reserved word) is used more than once.

  CVE-2000-0116, CAN-2001-1157 - extra "<" in front of SCRIPT tag.

  CAN-2002-2086 - XSS using "<<script" - probably a cleansing error

****

SPECM.INC. Inconsistent Special Elements

  Definition: the product does not handle when an inconsistency exists
  between two or more special characters or reserved words, e.g. if
  paired characters appear in the wrong order, or if the special
  characters are not properly nested.

==========

SECTION.9.5. [SPECTS] Technology-Specific Special Elements

==========

Note that special elements problems can arise from designs or languages that (1) do not separate "code" from "data" or (2) mix meta-information with information.

==========
SPECTS.XSS. Cross-site scripting (XSS)

Definition: the product does not properly cleanse HTML or script from an input before it is inserted into a web page, in a way that causes it to be processed on the web client.

Terminology Note: "CSS" was once used as the acronym for this problem, but this can cause confusion with the "Cascading Style Sheets," so its use has declined significantly, and its use is discouraged by the author.

Terminology Note: the terminology is imprecise for this category, which has a number of variants.

There are multiple ways to define this term or to split it into smaller categories.

One way is based on the "longevity" of the manipulated data. One could distinguish between "stored" data and "reflected" data. Stored data is inserted into a file or database, and later sent to a victim unquoted. Reflected data is first sent to a victim by injecting the data into the URL, which the victim's browser then sends to the server, which reflects that data back to the user.

In terms of channels, stored data is attacker-to-server-to-victim, in which all channels involve the web application. Reflected data is attacker-to-victim-to-server-to-victim, in which the attacker-to-victim channel can be external to the web application.

From the standpoint of the underlying WIFFs, however, both types of issues require a cleansing problem by the server.

One could also categorize these issues based on the type of manipulation involved. Full-blown injection of HTML tags such as SCRIPT, complete with unquoted "<" and ">" characters, is a different manipulation than the injection of "javascript:" or encoded "jav&#X41sc
ript:" schemes within a particular attribute of a tag. Many products protect against one manipulation

but not the other. The underlying faults, or other WIFFs, are often different.

PLOVER currently categorizes XSS WIFFs based on their associated manipulations.

Terminology Note: some people distinguish between XSS and "HTML injection." The distinction is generally made as follows: XSS would apply to "sending a link to a victim which then injects script or HTML into a page that the victim generates when clicking on or otherwise activating the link," whereas "HTML injection" applies to "inserting HTML/script into the server directly into a page that the server later replays to another user or set of users." From the server standpoint, this bug is the same - the server is sent some script, then returns it unfiltered to the victim. From the attack standpoint, these are different.

***

SPECTS.XSS.BASIC. Basic XSS

Definition: "Basic" XSS involves a complete lack of cleansing of any special characters, including the most fundamental XSS elements such as "<", ">", and "&".

CVE-2002-0938 - XSS in parameter in a link

CAN-2002-1495 - XSS in web-based email product via attachment filenames

CAN-2003-1136 - HTML injection in posted message

CAN-2004-2171 - XSS not quoted in error page

***

SPECTS.XSS.ERR. XSS in error pages

Note: this can overlap unhandled error conditions, and external or internal error message infoleaks.

CVE-2002-0840 - XSS in default error page from Host: header

CVE-2002-1053 - XSS in error message

CAN-2002-1700 - XSS in error page from targeted parameter

***

SPECTS.XSS.IMG. Script in IMG tags

   CAN-2002-1649, CAN-2002-1803, CAN-2002-1804, CAN-2002-1805,
   CAN-2002-1806, CAN-2002-1807, CAN-2002-1808 - javascript URI scheme
   in IMG tag

***

SPECTS.XSS.ATTRIB. XSS using Script in Attributes

  Definition: the product does not filter "javascript:" or other URI's
  from dangerous attributes within tags, such as onmouseover, onload,
  onerror, or style.

  CAN-2001-0520 - bypass filtering of SCRIPT tags using onload in
  BODY, href in A, BUTTON, INPUT, and others

  CVE-2002-1493 - guestbook XSS in STYLE or IMG SRC attributes

  CAN-2002-1965 - Javascript in onerror attribute of IMG tag

  CAN-2002-1495 - XSS in web-based email product via onmouseover event

  CAN-2002-1681 - XSS via script in <P> tag

  CAN-2003-1136 - javascript in onmouseover attribute

  CAN-2004-1935 - onload, onmouseover, and other events in an e-mail
  attachment

  CAN-2005-0945 - onmouseover and onload events in img, link, and mail
  tags.

  CAN-2003-1136 - onmouseover attribute in e-mail address or URL

****

SPECTS.XSS.ENCODED. XSS using Script Via Encoded URI Schemes

  CAN-2005-0563 - scheme "jav&#X41sc
ript:" in IMG tag

  CAN-2005-2276 - scheme "j&#X41vascript" in IMG tag

CAN-2005-0692 - encoded script within BBcode IMG tag

CVE-2002-0117, CAN-2002-0118 - encoded "javascript" in IMG tag

****

SPECTS.XSS.DOUBLE. Doubled character XSS manipulations, e.g. "<<script"

CAN-2002-2086 - XSS using "<<script"

CVE-2000-0116, CAN-2001-1157 - extra "<" in front of SCRIPT tag

****

SPECTS.XSS.INVTAGS. Invalid Characters in Identifiers

Definition: the product does not strip out invalid characters in the middle of tag names, schemes, and other identifiers, which are still rendered by some web browsers that ignore the characters.

Note: Overlaps interpretation conflict, incomplete blacklist.

Note: commonly used characters include null, CRLF, and other non-standard whitespace.

CAN-2004-0595 - XSS filter doesn't filter null characters before looking for dangerous tags, which are ignored by web browsers. MIE and validate-before-cleanse.

****

SPECTS.XSS.ALTSYN. Alternate XSS syntax

CVE-2002-0738 - XSS using "&={script}"

****

SPECTS.INJ.OS. OS Command Injection

Alternate Name: shell injection, shell metacharacters

Note: overlaps special character mismanagement, argument injection

Fault: unquoted special characters, input restriction error

Functional Area: program invocation

Examples: many, including CVE-1999-0067, CVE-2001-1246,
CVE-2002-0061, CAN-2003-0041

CAN-2002-1898 - shell metacharacters in a telnet:// link (this is an MFV: a metachar manipulation using an alternate channel)

****

SPECTS.INJ.ARG. Argument Injection or Modification

Definition: the product does not properly safeguard OS command or other arguments from modification by an attacker before execution, leading to security-relevant changes

Note: at one layer of abstraction, this can overlap other WIFFs that have whitespace problems, e.g. injection of javascript into attributes of HTML tags.

Fault: unquoted special characters, input restriction error, unquoted special terms, whitespace

Canonical Example: CVE-1999-0113

Examples: CVE-2004-0121, CAN-2003-0907, CVE-1999-0113,
CAN-2004-0480, CAN-2004-0489, CVE-2002-0985,
CVE-2001-0150, CVE-2001-0667, CAN-2004-0473,
CAN-2004-0411

CVE-2000-1220 - argument injection allows code execution by specifying alternate configuration file

****

SPECTS.INJ.SQL. SQL injection

Definition: the product does not properly filter or quote special characters or reserved words that are used in SQL queries, allowing attackers to modify the syntax, content, or commands of the resulting SQL query before it is executed.

Factors: resultant to special character mismanagement, MAID, or blacklist/whitelist problems. Can be primary to authentication errors.

Examples: many, including CAN-2004-0366, CAN-2004-0343,
CAN-2003-0779, CAN-2003-0500, CAN-2003-0377

****

SPECTS.INJ.LDAP. LDAP injection

Definition: the product does not properly filter or quote special characters or reserved words that are used in LDAP queries or responses, allowing attackers to modify the syntax, contents, or commands of the LDAP query before it is executed.

Research Gaps: under-reported. This is likely found very frequently by third party code auditors, but there are very few publicly reported examples.

Factors: resultant to special character mismanagement, MAID, or blacklist/whitelist problems. Can be primary to authentication and verification errors.

Ref: [SPI]

Web Applications and LDAP Injection (SPI Dynamics)

****

SPECTS.INJ.XML. XML injection (aka Blind Xpath injection)

Definition: the product does not properly filter or quote special characters or reserved words that are used in XML, allowing attackers to modify the syntax, content, or commands of the XML before it is processed by an end system.

Fault: unquoted special characters, input restriction error

Research Gaps: under-reported. This is likely found regularly by third party code auditors, but there are very few publicly reported examples.

Reference: [SanctumX]

****

SPECTS.INJ.CUSTOM. Custom Special Character Injection

Definition: the product does not properly filter or quote special

characters or reserved words that are used in a custom or proprietary language or representation that is used by the product, allowing attackers to modify the syntax, content, or commands before they are processed by an end system.

Research Gaps: under-studied. It is likely that these issues are fairly common in applications that use their own custom format for configuration files, logs, meta-data, messaging, etc. They would only be found by accident or with a focused effort based on an understanding of the format.

Factors: can be primary to interaction errors.

CVE-2001-0677 - read arbitrary files from mail client by providing a special MIME header that is internally used to store pathnames for attachments.

CVE-2000-0703 - Setuid program does not cleanse special escape sequence before sending data to a mail program, causing the mail program to process those sequences

CVE-2003-0020, CAN-2003-0083 - Multi-channel issue. Terminal escape sequences not filtered from log files

****

SPECTS.CRLF. CRLF Injection

Definition: the product uses CRLF (carriage return line feeds) as a special sequence, e.g. to separate lines or records, but it does not properly handle inputs that contain CRLF sequences.

Reference: CRLF Injection [Harnhammar]

Research Gaps: probably under-studied, although gaining more prominence in 2005 as a result of interest in HTTP response splitting.

Factors: primary to HTTP Response Splitting

CAN-2002-1771 - CRLF injection enables spam proxy (add mail headers) using email address or name

CAN-2002-1783 - CRLF injection in API function arguments modify headers for outgoing requests

CAN-2004-1513 - spoofed entries in web server log file via carriage returns

****

SPECTS.NULLTERM. Improper Null Termination

Definition: the product does not properly terminate a string or array with a null character or equivalent terminator.

Null termination errors frequently occur in two different ways. An off-by-one error could cause a null to be written out of bounds, leading to an overflow. Or, a program could use a strncpy() function call incorrectly, which prevents a null terminator from being added at all. Other scenarios are possible.

Note: conceptually, this does not just apply to the C language; any languagre or representation that involves a terminator could have this sort of problem.

Factors: this is usually resultant from other WIFFs such as off-by-one errors, but it can be primary to boundary condition violations such as buffer overflows. In buffer overflows, it can act as an expander for assumed-immutable data.

Note: overlaps missing input terminator.

CAN-2000-0312 - attacker does not null-terminate argv[] when invoking another program

CAN-2003-0777 - interrupted step causes resultant lack of null termination

CAN-2004-1072 - fault causes resultant lack of null termination, leading to buffer expansion

CAN-2001-1389 - multiple vulns related to improper null termination.

CAN-2003-0143 - product does not null terminate a message buffer after snprintf-like call, leading to overflow

==========

SECTION.9.6. [PATH] Path Traversal and Equivalence Errors

==========

Files, directories, and folders are so central to information technology that many different WIFFs and variants have been discovered. The manipulations generally involve special characters or sequences in pathnames, or the use of alternate references or channels. They can be used to access files outside of a restricted directory (path traversal or link following) or to access files that are otherwise protected (path equivalence).

****

PATH.TRAV. Path Traversal

Definition: the product, when constructing file or directory names from input, does not properly cleanse special character sequences that resolve to a file or directory name that is outside of a restricted directory.

Alternate Term: directory traversal

Functional Area: file processing

Attack: use alternate names, encoding

Note: many different manipulations exist; most of them are intended to bypass various cleansing schemes.

Terminology Note: "path traversal" is preferred over "directory traversal."

Terminology Note: Like other WIFFs, terminology is often based on the types of manipulations used, instead of the underlying WIFFs.

Terminology note: some people use "directory traversal" only to refer to the injection of ".." and equivalent sequences whose specific meaning is to traverse directories. Other variants like "absolute pathname" and "drive letter" have the *effect* of directory traversal, but some people may not call it such, since it doesn't involve ".." or equivalent.

Note: pathname equivalence can be regarded as a type of canonicalization error.

Note: some pathname equivalence issues are not directly related to directory traversal, rather are used to bypass security-relevant checks for whether a file/directory can be accessed by the attacker (e.g. a trailing "/" on a filename could bypass access rules that

don't expect a trailing /, causing a server to provide the file when it normally would not).

Note: this item should be split into multiple sub-categories, but for the sake of consistency with the numbering convention in earlier versions of this checklist, they were combined.

Note: Incomplete diagnosis or reporting of vulnerabilities can make it difficult to know which variant is affected. For example, a researcher might say that "..\" is vulnerable, but not test "../" which may also be vulnerable.

Note: any combination of the items below can provide its own variant, e.g. "//../" is not listed (CAN-2004-0325).

Note: most of these issues are probably under-studied.

****

PATH.TRAV.REL. Relative Path Traversal

Definition: the product can construct a path that contains relative traversal sequences such as "..", which is then processed by the operating system to create a path that is "above" or outside of a restricted path that was intended by the developer.

****

PATH.TRAV.REL.1. "../filedir"

****

PATH.TRAV.REL.2. "/../filedir"

CVE-2005-1918 - a patch that protects against most ".." issues does not protect against "/../file/here".

****

PATH.TRAV.REL.3. "/directory/../filename"

****

PATH.TRAV.REL.4. "directory/../../filename"

CAN-2002-0298

****

PATH.TRAV.REL.5. "..\filename"   ("dot dot backslash")

    Examples: many, including CAN-2002-0661, CVE-2002-0946,
              CAN-2002-1042, CAN-2002-1209, CVE-2002-1178

****

PATH.TRAV.REL.6. "\..\filename" ("leading dot dot backslash")

    Examples:

    CAN-2002-1987 - \..
    CAN-2005-2142 - \..

****

PATH.TRAV.REL.7. "\directory\..\filename"

    Examples: CVE-2002-1987

****

PATH.TRAV.REL.8. "directory\..\..\filename"

    Examples: CVE-2002-0160

****

PATH.TRAV.REL.9. "..." (triple dot)

  Note: This manipulation is effective in two different contexts: (1)
  it is equivalent to "..\.." on Windows, or (2) it can take advantage
  of insufficient filtering, e.g. if the programmer does a single-pass
  removal of "./" in a string (collapse of data into unsafe value)

  Overlaps: Collapse of Data into Unsafe Value

  Examples:

  CVE-2001-0467 - \... in web server

  CVE-2001-0615 - "..." or "...." in chat server

CVE-2001-0963, CVE-2001-1193, CAN-2001-1131 - "..." in cd command in FTP server

CAN-2001-0480 - "..." in GET or CD command in FTP server

CAN-2002-0288 - ... in web server

CAN-2002-0784 - HTTP server protects against ".." but allows "..."

CAN-2003-0313 - directory listing of web server using "..."

CAN-2005-1658 - triple dot

****

PATH.TRAV.REL.10. "...." (multiple dot)

Note: see "..." notes.

Examples:

CVE-2000-0240 - read files via "/.........../" in URL

CVE-2000-0773, CVE-2000-0773 - read files via "...." in web server

CAN-1999-1082, CAN-2004-2121 - read files via "......" in web server (doubled triple dot?)

CAN-2001-0491 - multiple attacks using "..", "...", and "...." in different commands

CVE-2001-0615 - "..." or "...." in chat server

****

PATH.TRAV.REL.11. "..../" (doubled dot dot slash)

Note: this could occur due to a cleansing error that removes a single "../" from "..../"

Examples: Merak Mail Server with Icewarp, Sep. 10, 2004

****

PATH.TRAV.REL.12. ".../.../"

Note: this is effective when a programmer uses a "../" regular expression in an attempt to remove sequences. Removing the two "../" sequences from ".../...//" yields "../".

Examples:

CAN-2005-2169

CAN-2005-0202 - ".../..../" bypasses regexp's that remove "./" and "../"

==========

PATH.TRAV.ABS. Absolute Path Traversal

Definition: the product can construct a path that contains absolute path sequences such as "/path/here," which is then processed by the operating system to access a file or resource that is "above" or outside of a restricted path that was intended by the developer.

****

PATH.TRAV.ABS.1. /absolute/pathname/here

Examples:

CAN-2002-1345 - multiple FTP clients write arbitrary files via absolute paths in server responses

CAN-2001-1269 - ZIP file extractor allows full path

CAN-2002-1818, CAN-2002-1913, CAN-2005-2147 - path traversal using absolute pathname

****

PATH.TRAV.ABS.2. \absolute\pathname\here ("backslash absolute path")

Examples: many, including CVE-1999-1263, CAN-2003-0753, CAN-2002-1344, CAN-2002-1525, CAN-2000-0614,

****

PATH.TRAV.ABS.3. "C:dirname" or C: (Windows volume or "drive letter")

Examples: CAN-2001-0038, CAN-2001-0255, CAN-2001-0687,

CAN-2001-0933, CAN-2002-0466, CAN-2002-1483

CVE-2004-2488 - FTP server read/access arbitrary files using "C:\" filenames

****

PATH.TRAV.ABS.4. "\\UNC\share\name\" (Windows UNC share)

Examples: CAN-2001-0687

==========

PATH.EQ. Path Equivalence

Alternate Term: pathname equivalence

Path equivalence involves the use of special characters in file and directory names. However, it is different because the associated manipulations are all intended to generate multiple names for the same object, whereas in path traversal, the manipulations are performed to generate a name for a different object.

Note: some of these manipulations could be effective in path traversal issues, too.

****

PATH.EQ.TDOT. Trailing Dot - "filedir."

CAN-2000-1114, CAN-2002-1986, CAN-2004-2213, CVE-2005-3293 - source code disclosure using trailing dot

CAN-2004-0061, CAN-2000-1133 - bypass directory access restrictions using trailing dot in URL

CVE-2001-1386 - bypass check for ".lnk" extension using ".lnk."

****

PATH.EQ.IDOT. Internal Dot - "file.ordir"

Note: this variant does not have any easily findable, publicly reported vulnerabilities, but it can be an effective manipulation in WIFFs such as validate-before-cleanse, which might remove a dot from a string to produce an unexpected string.

****

PATH.EQ.IDOT.MULT. Multiple Internal Dot - "file...dir"

  Note: this variant does not have any easily findable, publicly
  reported vulnerabilities, but it can be an effective manipulation in
  WIFFs such as validate-before-cleanse, which might use a regular
  expression that removes ".." sequences from a string to produce an
  unexpected string.

****

PATH.EQ.TDOT.MULT. Multiple Trailing Dot - "filedir...."

   Examples:

   BUGTRAQ:20040205 Apache + Resin Reveals JSP Source Code ...

   CAN-2004-0281 - multiple trailing dot allows directory listing

****

PATH.EQ.TSPACE. Trailing Space - "filedir "

   CAN-2001-0693, CAN-2001-0778, CAN-2001-1248, CAN-2004-0280,
   CAN-2004-2213, CAN-2005-0622, CAN-2005-1656, CAN-2002-1603 - source
   disclosure via trailing encoded space "%20"

   CVE-2001-0054 - MFV. directory traversal and other issues in
   FTP server using Web encodings such as "%20"; certain manipulations
   have unusual side effects

   CAN-2002-1451 - trailing space ("+" in query string) leads to
   source code disclosure

***

PATH.EQ.LSPACE. Leading Space - " filedir"

***

PATH.EQ.ISPACE. file[SPACE]name (internal space)

   Note: This is not necessarily an equivalence issue, but it can also
   be used to spoof icons or conduct information hiding via
   information truncation (see user interface errors).

This WIFF is likely to overlap quoting problems, e.g. the "Program Files" untrusted search path variants. It also could be an equivalence issue if filtering removes all extraneous spaces.

CAN-2000-0293 - filenames with spaces allow arbitrary file deletion when product does not properly quote them; some overlap with path traversal

CVE-2001-1567 - "+" characters in query string converted to spaces before sensitive file/extension (internal space), leading to bypass of access restrictions to the file

***

PATH.EQ.TSLASH. filedir/ (trailing slash, trailing /)

Examples: CAN-2002-0253 (overlaps infoleak), CAN-2001-0446, CAN-2004-0334, CAN-2001-0893, CAN-2001-0892, CAN-2004-1814, BID:3518

***

PATH.EQ.MLSLASH. //multiple/leading/slash ("multiple leading slash")

Examples: CVE-2000-1050, CAN-2002-1483, CVE-1999-1456, CAN-2004-0235, CAN-2004-0578, CVE-2002-0275, CVE-2000-1050, CVE-2001-1072, CAN-2004-1032, CAN-2002-1238, CAN-2004-1878, CAN-2005-1365 (leading slash with "..")

CVE-2000-1050 - access directory using multiple leading slash

CVE-2001-1072 - bypass access restrictions via multiple leading slash, which causes a regular expression to fail

CAN-2004-0235 - archive extracts to arbitrary files using multiple leading slash in filenames in the archive.

***

PATH.EQ.MISLASH. /multiple//internal/slash ("multiple internal slash")

CAN-2002-1483 - read files with full pathname using multiple internal slash

***

PATH.EQ.MTSLASH. /multiple/trailing/slash// ("multiple trailing slash")

   CAN-2002-1078 - directory listings in web server using multiple
   trailing slash

***

PATH.EQ.MIBSLASH. \multiple\\internal\backslash

***

PATH.EQ.TBSLASH. filedir\   (trailing backslash)

   Examples: CAN-2004-0847

***

PATH.EQ.DOTDIR. /./ (single dot directory)

   Examples: CVE-2000-0004, CAN-2002-0304, BID:6042, possibly
   CAN-1999-1083 (could be a cleansing error), CAN-2004-0815
   ("/./////etc" cleansed to ".///etc" then "/etc"), CAN-2002-0112

***

PATH.EQ.WILD. filedir* (asterisk / wildcard)

   CAN-2004-0696 - list directories using desired path and "*"

   CAN-2002-0433 - list files in web server using "*.ext"

***

PATH.EQ.INOUT. dirname/fakechild/../realchild/filename

  Note: this is a manipulation that uses an injection for one
  consequence (containment violation using relative path) to achieve a
  different consequence (equivalence by alternate name)

   Examples: CAN-2001-1152, CVE-2000-0191

   CAN-2005-1366 - CGI source disclosure using "dirname/../cgi-bin"

****

PATH.EQ.83FNAME. Windows 8.3 Filename

  Definition: on later Windows operating systems, a file can have a "long name" and a short name that is compatible with older Windows file systems, with up to 8 characters in the filename and 3 characters for the extension. These "8.3" filenames, therefore, have the "alternate name" property for files with long names, so are useful pathname equivalence manipulations.

  Research Gaps: probably under-studied.

  Functional Area: file processing

  Property: alternate name

  CVE-1999-0012 - multiple web servers allow restriction bypass using 8.3 names instead of long names

  CAN-2001-0795 - source code disclosure using 8.3 file name

  CAN-2005-0471 - MFV. Product generates temporary filenames using long filenames, which become predictable in 8.3 format.

==========

PATH.LINK. Link Following

Link following WIFFs involve insufficient protection against links or shortcuts that can reference a file other than the one that was intended.

Research Gaps: UNIX hard links, and Windows hard/soft links are under-studied and under-reported.

Terminology Note: some people use the phrase "insecure temporary file" when referring to a link following WIFF, but other WIFFs can product insecure temporary files without any symlink involvement at all.

Functional Area: file processing, temporary files

Properties: alternate reference, pathname equivalence, container escape

Factors: race conditions, permissions, predictability

OS-specific: no

Note: link following vulnerabilities are MFV. They are the combination of multiple elements: file or directory permissions, filename predictability, etc.

Note: Windows soft links can be exploited remotely since a ".LNK" file can be uploaded like a normal file.

***

PATH.LINK.UNIX.SYM. UNIX symbolic link following

Fault: filename predictability, insecure directory permissions, non-atomic operations, race condition

Alias: symlink following, symlink vulnerability

Note: these are typically reported for temporary files or privileged programs

Reference: [Colley2004]

Research Gaps: symlink vulnerabilities are regularly found in C and shell programs, but all programming languages can have this problem.

Research Gaps: "second-order symlink vulnerabilities" may exist in programs that invoke other programs that follow symlinks. They are rarely reported but are likely to be fairly common when process invocation is used. Reference: [Christey2005]

Examples: many, including CVE-1999-1386, CVE-2000-0972, CVE-2000-1178, CAN-2004-0217, CAN-2003-0517

CAN-2004-0689 - Possible interesting example

CAN-2005-1879, CAN-2005-1880 - second-order symlink vulns

CAN-2005-1916 - symlink in Python program

CVE-2000-0972 - setuid product allows file reading by replacing a file being edited with a symlink to the targeted file, leaking the result in error messages when parsing fails.

CAN-2005-0824 - signal causes a dump that follows symlinks

***

PATH.LINK.UNIX.HARD. UNIX hard link

  Research Gaps: under-studied. It is likely that programs that
  check for symbolic links could be vulnerable to hard links.

  Examples:

  CAN-2001-1494 - hard link attack, file overwrite; interesting
  because program checks against soft links

  CAN-2002-0793

   CAN-2003-0578

   CVE-1999-0783

   CAN-2004-1603

   CAN-2004-1901

   CAN-2005-1111

   hard link race condition
   BUGTRAQ:20030203 ASA-0001: OpenBSD chpass/chfn/chsh file content leak
   URL:http://www.securityfocus.com/archive/1/309962

***

PATH.LINK.WIN.LNK. Windows Shortcut Following (.LNK)

  Alternate name: Windows symbolic link following (symlink)

   Research Gaps: under-studied. Windows .LNK files are more
   "portable" than Unix symlinks and have been used in remote
   exploits. Some Windows API's will access LNK's as if they are
   regular files, so one would expect that they would be reported more
   frequently.

   Examples: CVE-2000-0342, CAN-2001-1042, CVE-2001-1043,
   CAN-2005-0587

   CAN-2001-1386 - ".LNK." - .LNK with trailing dot

CVE-2003-1233 - rootkits can bypass file access restrictions to Windows kernel directories using NtCreateSymbolicLinkObject function to create symbolic link

***

PATH.LINK.WIN.HARD. Windows hard link

Research Gaps: under-studied.

Examples: CAN-2002-0725, CAN-2003-0844

==========

PATH.VIRT. Virtual Files

Virtual file names are represented like normal file names, but they are effectively aliases for other resources that do not behave like normal files. Depending on their functionality, they could be alternate entities. They are not necessarily listed in directories.

Functional Area: file processing

****

PATH.VIRT.DOSDEV. Windows MS-DOS device names

Note: Historically, there was a bug in the Windows operating system that caused a blue screen of death, but even after that issue was fixed, DOS device names continue to be a factor.

Examples: CAN-2002-0106, CAN-2002-0200, CAN-2002-1052, CVE-2001-0493, CVE-2001-0558, CVE-2000-0168, CAN-2001-0492, CAN-2004-0552, CAN-2005-2195

***

PATH.VIRT.WIN.ALTSTREAM. Windows ::DATA alternate data stream

Properties: alternate name, alternate channel

Fault: multiple identifiers, non-atomic object

Examples: CVE-1999-0278, CVE-2000-0927 (note: there may be others with different attack vectors and impacts)

***

PATH.VIRT.MAC.DS. Apple ".DS_Store"

   Research Gaps: under-studied.

   Examples:

   BUGTRAQ:20010910 More security problems in Apache on Mac OS X

***

PATH.VIRT.MAC.ALTSTREAM. Apple HFS+ alternate data stream

   Fault: multiple identifiers, non-atomic object

   Research Gaps: under-studied.

   Example: CAN-2004-1084

SECTION.9.7. [CP] Channel and Path Errors

A number of vulnerabilities are specifically related to problems in creating, managing, or removing alternate channels and alternate paths.

Some of these can overlap virtual file problems (CHAP.VIRTFILE), and they can are commonly used in "bypass" attacks, such as those that exploit authentication errors.

Research Gaps: most of these issues are probably under-studied. Only a handful of public reports exist.

****

CP.CHAN. Channel Errors

****

CP.CHAN.PRIM. Unprotected Primary Channel

  Definition: the product uses a primary channel for administration or restricted functionality, but it does not properly protect the

channel.

****

CP.CHAN.ALT. Unprotected Alternate Channel

Definition: the product protects a primary channel, but it does not use the same level of protection for an alternate channel.

Factors: this can be primary to authentication errors, and resultant from unhandled error conditions.

CVE-2002-0567 - DB server assumes that local clients have performed authentication, allowing attacker to directly connect to a process to load libraries and execute commands; a socket interface also exists (another alternate channel), so attack can be remote.

CAN-2002-1578 - product does not restrict access to underlying database, so attacker can bypass restrictions by directly querying the database.

CAN-2003-1035 - user can avoid lockouts by using an API instead of the GUI to conduct brute force password guessing

CAN-2002-1863 - FTP service can not be disabled even when other access controls would require it

CVE-2002-0066 - Windows named pipe created without authentication/access control, allowing configuration modification

CVE-2004-1461 - router management interface spawns a separate TCP connection after authentication, allowing hijacking by attacker coming from the same IP address.

****

CP.CHAN.ALT.RACE. Alternate Channel Race Condition

Definition: the product opens an alternate channel for communication with an authorized user, but the channel is unprotected and a race condition allows an attacker to access the channel instead.

Attack: hijack

Note: predictability can be a factor in some issues.

CVE-1999-0351 - FTP "Pizza Thief" vulnerability. Attacker can connect to a port that was intended for use by another client.

CAN-2003-0230 - hijack alternate named pipe while another user is authenticating.

CAN-2003-0230 - product creates Windows named pipe during authentication that another attacker can hijack by connecting to it.

****

CP.CHAN.PROXIED. Proxied Trusted Channel

Definition: the product controls and trusts both endpoints of a channel, but one endpoint can be accessed by an attacker and used as a proxy to interact with the product.

Note: this can overlap some other categories, such as those exploited by "bounce" attacks, but the idea here is that both endpoints are under control of the product.

****

CP.CHAN.ALT.WM. Unprotected Windows Messaging Channel ("Shatter")

Definition: the product does not properly verify the source of a message in the Windows Messaging System while running at elevated privileges, creating an alternate channel through which an attacker can directly send a message to the product.

Reference: [Paget]

Note: alternate channel attacks likely exist in other operating systems and messaging models, e.g. in privileged X Windows applications, but examples are not readily available.

Note: overlaps privilege errors and UI errors.

Research Gaps: possibly under-reported, probably under-studied. It is suspected that a number of publicized vulnerabilities that involve local privilege escalation on Windows systems may be related to Shatter attacks, but they are not labeled as such.

CAN-2002-0971 - bypass GUI and access restricted dialog box

CVE-2002-1230 - gain privileges via Windows message

CAN-2003-0350 - a control allows a change to a pointer for a callback function using Windows message

CAN-2003-0908 - product launches Help functionality while running with raised privileges, allowing command execution using Windows message to access "open file" dialog.

CAN-2004-0213 - attacker uses Shatter attack to bypass GUI-enforced protection for CAN-2003-0908.

CAN-2004-0207 - user can call certain API functions to modify certain properties of privileged programs

==========

CP.ALTPATH. Alternate Path Errors

Note: this is partially covered by other categories.

****

CP.ALTPATH.DREQ. Direct Request aka "Forced Browsing"

Definition: the web product does not sufficiently prevent a restricted or supporting script from being accessed directly by an attacker without going through the expected path.

Terminology Note: the "forced browsing" term could be misinterpreted to include WIFFs such as CSRF or XSS, so its use is discouraged.

Note: "Forced browsing" is a step-based manipulation involving the omission of one or more steps, whose order is assumed to be immutable; the application does not verify that the first step was performed. The consequence is typically "authentication bypass" or "path disclosure," although it can expose all kinds of WIFFs, especially in languages such as PHP, which allow external modification of assumed-immutable variables.

Note: overlaps MAID, authorization errors, container errors; often primary to other WIFFs such as XSS and SQL injection.

CAN-2004-2144 - bypass authentication via direct request

CAN-2005-1892 - infinite loop or infoleak triggered by direct requests

CAN-2004-2257 - bypass auth/auth via direct request

CAN-2005-1688, CAN-2005-1697, CAN-2005-1698 - direct request leads to infoleak by error

CAN-2005-1685, CAN-2005-1827 - authentication bypass via direct request

CAN-2005-1654 - authorization bypass using direct request

CAN-2005-1668 - access privileged functionality using direct request

CAN-2002-1798 - upload arbitrary files via direct request

****

CP.ALTPATH.MISC. Miscellaneous alternate path errors

CVE-1999-1111 - Buffer overflow protection mechanism allows bypass using alternate path

CAN-2005-2148 - web product cleanses input from POST requests but not the URL, which allows an alternate path attack by specifying good values in POST, and bad values in the URL.

CVE-2005-3300 - PHP script filters dangerous variables from inputs, but forgets to include _FILES. Overlaps blacklist/whitelist errors and MAID.

CAN-2002-2059 - bypass of hardware restrictions via alternate path (pressing F8 key)

CVE-2004-2352 - XSS in PHP script via an alternate path using cookies instead of POST data

CAN-2001-0919 - cookie preferences can be bypassed by setting a cookie using Javascript

CAN-2005-1590 - disable password protection and access interface by using Windows code to find a hidden window

CVE-2003-1127 - web server allows source disclosure using HTTP TRACE method.

CAN-2002-1715 - restricted shell bypass by uploading a script to a world-writable directory. Permissions, privileges, alternate path.

==========

CP.UPATH. Untrusted Search Path

Definition: The product uses a set of one or more "paths" to search for desired resources, such as code libraries or configuration files, but an attacker can modify the path so that it references an attacker-controlled resource.

Alternate Term: Untrusted Path, Mutable Search Path (PLOVER 0.18 and earlier)

Functional Area: program invocation, code libraries

Research Gaps: search path issues on Windows are under-studied and possibly under-reported.

CVE-1999-1120, CAN-2002-0470 - application relies on its PATH environment variable to find and execute program

***

CP.UPATH.ELEMENT. Uncontrolled Search Path Element

Definition: One or more elements in a static search path is under control of the attacker, e.g. "." or "/tmp", even though the attacker cannot modify the search path itself.

Note: PHP file inclusion is an instance of this (see PHP-specific issues); trusted environment variables is another.

Examples: CAN-2002-1576, CAN-2000-1128, CAN-1999-1461, CVE-1999-1318, CAN-2003-0579, CVE-2001-0507, CVE-2000-0854, CAN-2001-0943, CAN-2001-0942, CAN-2001-0507, CAN-2002-2017

CVE-1999-0690

CVE-2001-0912 - error during packaging causes product to include a hard-coded, non-standard directory in search path

CVE-2001-0289, CAN-2005-1705 - product searches current working directory for configuration file

CVE-2005-1307 - product executable other program from current working directory.

CAN-2002-2040 - untrusted path

CAN-2005-2072 - modification of trusted environment variable leads to untrusted path vuln

CAN-2005-1632 - product searches /tmp for modules before other paths

***

CP.UPATH.UNQUOT. Unquoted Search Path or Element

Note: This covers "C:\Program Files" and space-in-search-path issues

Notes: this theoretically could apply to other OSes besides Windows, especially those that make it easy for spaces to be in files or folders.

Research Gaps: under-studied, probably under-reported.

Functional area: program invocation

Fault: missing quoting

Weakness: whitespace allowed in identifiers

Examples:

CAN-2005-1185, small handful of others. Program doesn't quote the "C:\Program Files\" path when calling a program to be executed - or any other path with a directory or file whose name contains a space - so attacker can put a malicious program.exe into C:.

CVE-2005-2938 - CreateProcess() and CreateProcessAsUser() can be misused by applications to allow "program.exe" style attacks in C:

CAN-2000-1128 applies to "Common Files" folder, with a malicious common.exe, instead of "Program Files"/program.exe.

==========

SECTION.9.8. [CCC] Cleansing, Canonicalization, and Comparison Errors

==========

Note: this section needs more work. Most of the key concepts are already covered by special characters and alternate encodings.

****

CCC.ENC. Encoding Error

  Note: partially overlaps path traversal and equivalence

  Note: many other types of encodings should be listed here

****

CCC.ENC.ALT. Alternate Encoding

  Definition: the product does not properly handle when an input has been modified to use an alternate encoding.

****

CCC.ENC.DOUBLE. Double Encoding

  Definition: the product does not properly handle when an input has been encoded twice.

  Research Gaps: probably under-studied.

  Examples: CVE-2001-0333, CAN-2005-0054, CAN-2004-1315, CAN-2004-1938, CAN-2004-1939

  CVE-2001-0333 - directory traversal using double encoding

  CAN-2004-1938 - "%2527" (double-encoded single quote) used in SQL injection

  CAN-2005-1945 - double hex-encoded data

  CAN-2005-0054 - browser executes HTML at higher privileges via URL with hostnames that are double hex encoded, which are decoded twice to generate a malicious hostname.

****

CCC.ENC.MIXED. Mixed Encoding

Definition: the product does not properly handle when the same input uses multiple (mixed) encodings.

****

CCC.ENC.UNI. Unicode Encoding

Examples: CVE-2000-0884, CAN-2001-0709, CAN-2001-0669 (overlaps interaction error)

****

CCC.ENC.URL. URL Encoding (Hex Encoding)

Definition: the product does not properly handle when all or part of an input has been URL encoded.

CVE-2000-0900, CAN-2005-2256, CAN-2004-2121 - hex-encoded path traversal variants - "%2e%2e", "%2e%2e%2f", "%5c%2e%2e"

CAN-2004-0280, CAN-2003-0424, CAN-2001-0693, CAN-2001-0778 - "%20" (encoded space)

CAN-2002-1831 - crash via hex-encoded space "%20"

CVE-2000-0671, CVE-2004-0189, CAN-2002-1291, CVE-2002-1031, CAN-2001-1140, CAN-2004-0760, CVE-2002-1025 - "%00" (encoded null)

CAN-2004-0072 - "%2e" (encoded dot)

CAN-2002-1213 - "%2f" (encoded slash)

CAN-2004-0072, CAN-2004-0847 - "%5c" (encoded backslash)

CAN-2002-1575 - "%0a" (overlaps CRLF)

****

CCC.CASE. Case Sensitivity (lowercase, uppercase, mixed case)

Improperly handled case sensitive data can lead to several possible consequences, including:

[*] case-insensitive passwords reducing the size of the key space, making brute force attacks easier

[*] bypassing filters or access controls using alternate names
[*] multiple interpretation errors using alternate names

Functional Area: file processing, passwords

Property: alternate names

Research Gaps: these are probably under-studied in Windows environments, where file names are case-insensitive and thus are subject to equivalence manipulations involving case.

Examples: CVE-2000-0497, CVE-2000-0498, CAN-2001-0766, CAN-2001-0795, CAN-2001-1238, CAN-2003-0411, CAN-2002-0485 (leads to interpretation error), CVE-1999-0239, CAN-2005-0269, CAN-2004-1083, CAN-2004-2154 (overlaps ACL bypass), CVE-2000-0499

CVE-2002-2119 - case insensitive passwords lead to search space reduction

CVE-2004-2214 - HTTP server allows bypass of access restrictions using URIs with mixed case

CAN-2004-2154 - mixed upper/lowercase allows bypass of ACLs

CAN-2004-2214 - bypass access restrictions using mixed case

CVE-2005-4509 - bypass malicious script detection by using tokens that aren't case sensitive.

CAN-2002-1820 - mixed case problem allows "admin" to have "Admin" rights (alternate name property)

****

CCC.EVE. Early Validation Errors

Products need to validate data at the proper time, after data has been canonicalized and cleansed. Early validation is susceptible to various manipulations that result in dangerous inputs that are produced by canonicalization and cleansing.

Note: since early validation errors usually arise from improperly implemented defensive mechanisms, it is likely that these will be reported more frequently as secure programming becomes implemented

more widely.

Research Gaps: these errors are mostly reported in path traversal vulnerabilities, but the concept applies anyplace where filtering occurs.

****

CCC.VBC. Validate-Before-Canonicalize

Definition: a program "validates" data before it is canonicalized, which leaves it vulnerable to certain manipulations that are later removed during canonicalization. Invalid data can then avoid detection before it is produced by canonicalization.

Functional Area: non-specific

Note: this overlaps other categories

Examples: CAN-2002-0433, CAN-2003-0332, CVE-2002-0802, CVE-2000-0191 (overlaps "fakechild/../realchild")

CVE-2004-2363 - product checks URI for "<" and other literal characters, but does it before hex decoding the URI, so "%3E" and other sequences are allowed.

****

CCC.FILTER. Validate-Before-Filter

Definition: a program validates data before it has been filtered or cleansed, which could produce dangerous data after the filtering step.

Alternate Term: validate-before-cleanse

Functional Area: non-specific

Note: this category is probably under-studied.

Examples: CAN-2002-0934, CAN-2003-0282, possibly CAN-2003-0417, apexec.pl

****

CCC.COLLAPSE. Collapse of Data into Unsafe Value

Definition: the product cleanses or filters data in a way that causes the data to "collapse" into an unsafe value.

Note: overlaps regular expressions, although an implementation might not necessarily use regexp's.

CAN-2004-0815 - "/.////" in pathname collapses to absolute path.

CVE-2005-3123 - "/.//../////././" is collapsed into "/.././" after ".." and "//" sequences are removed.

CAN-2002-0325 - ".../.../" collapsed to "..." due to removal of "./" in web server

CAN-2002-0784 - "///./../.../" claimed to work - "./" removal would produce "///..."

CAN-2005-2169 - MFV. Regular expression intended to protect against directory traversal reduces ".../.../" to "../".

****

CCC.WHITELIST. Permissive Whitelist

Definition: an application uses a "whitelist" of acceptable values, but the whitelist permits at least one unsafe value.

Note that a permissive whitelist produces resultant vulnerabilities.

****

CCC.BLACKLIST. Incomplete Blacklist

Definition: an application uses a "blacklist" of prohibited values, but the blacklist is incomplete.

Note: an incomplete blacklist frequently produces resultant WIFFs. Exploitation of those WIFFs using the obvious manipulations might fail, but minor variations might succeed.

Note: some incomplete blacklist issues might arise from multiple interpretation errors, e.g. a blacklist for dangerous shell metacharacters might not include a metacharacter that only has meaning in one particular shell, not all of them; or a blacklist for XSS manipulations might ignore an unusual construct that's supported

by one web browser, but not others.

CVE-2005-2782 - PHP remote file inclusion in web application that filters "http" and "https" URLs, but not "ftp".

CAN-2004-0542 - programming language does not filter certain shell metacharacters in Windows environment.

CAN-2004-0595 - XSS filter doesn't filter null characters before looking for dangerous tags, which are ignored by web browsers. MIE and validate-before-cleanse.

CVE-2005-3287 - web-based mail product doesn't restrict dangerous extensions such as ASPX on a web server, even though others are prohibited.

CVE-2004-2351 - resultant XSS from incomplete blacklist (only <script> and <style> are checked)

CVE-2005-2959 - privileged program does not clear sensitive environment variables that are used by bash. Overlaps multiple interpretation error.

CAN-2005-1824 - SQL injection protection scheme does not quote the "\" special character.

CAN-2005-2184 - incomplete blacklist prevents user from automatically executing .EXE files, but allows .LNK, allowing resultant Windows symbolic link

****

CCC.REGEXP. Regular Expression Error

Definition: a regular expression is incorrectly specified in a way that causes data to be improperly filtered, compared, or cleansed.

Keywords: regexp

Note: this can seem to overlap whitelist/blacklist problems, but it is intended to deal with improperly written regular expressions, regardless of the values that those regular expressions use.

Note: can overlap partial comparison.

Note: interacts with null byte in PHP.

Research Gaps: regexp errors are likely a primary factor in many MFVs, especially those that require multiple manipulations to exploit. However, they are rarely diagnosed at this level of detail.

CVE-2002-2109 - regexp isn't "anchored" to the beginning or end, which allows spoofed values that have trusted values as substrings.

CAN-2005-1949 - regexp for IP address isn't anchored at the end, allowing appending of shell metacharacters

CVE-2001-1072 - bypass access restrictions via multiple leading slash, which causes a regular expression to fail

CAN-2000-0115 - local user DoS via invalid regular expressions

CAN-2002-1527 - error infoleak via malformed input that generates a regular expression error

CAN-2005-0603 - error infoleak via regular expression with invalid syntax

CAN-2005-1061 - certain strings are later used in a regexp, leading to a resultant crash.

CAN-2005-2169 - MFV. Regular expression intended to protect against directory traversal reduces ".../...//" to "../".

CAN-2005-0603 - malformed regexp syntax leads to error infoleak

CAN-2005-1820 - code injection due to improper quoting of regular expression

CVE-2005-3153, CVE-2005-4155 - null byte bypasses PHP regexp check

****

CCC.REGEXP.REST. Overly Restrictive Regular Expression

Definition: a regular expression is overly restrictive, which prevent dangerous values from being detected.

Note: can overlap whitelist/blacklist errors.

CAN-2005-1604 - MIE. ".php.ns" bypasses ".php$" regexp but is still parsed as PHP by Apache. (manipulates an equivalence property under Apache)

****

CCC.PCOMP. Partial Comparison

Definition: User input is only partially compared to the desired input before a match is determined.

For example, an attacker might succeed in authentication by providing a small password that matches the associated portion of the larger, correct password.

Note: this is conceptually similar to other WIFFs, such as insufficient verification and regular expression errors. it is primary to some WIFFs.

Examples:

CAN-2004-1012 - argument parser of an IMAP server treats a partial command "body[p" as if it is "body.peek", leading to index error and out-of-bounds corruption.

CAN-2004-0765 - web browser only checks the hostname portion of a certificate when the hostname portion of the URI is not a fully qualified domain name (FQDN), which allows remote attackers to spoof trusted certificates.

CVE-2002-1374 - one-character password by attacker checks only against first character of real password

==========
SECTION.9.9. [INFO] Information Management Errors
==========

****

INFO.LEAK. Information Leak (information disclosure)

Definition: an information leak is the intentional or unintentional disclosure of information that either (1) is regarded as sensitive

within the product's own functionality, such as a private message, or (2) provides information about the product or its environment that could be useful in an attack but is normally not available to the attacker, such as the installation path of a product that is remotely accessible.

Many information leaks are resultant (e.g. path disclosure in PHP script error), but they can also be primary (e.g. timing discrepancies in crypto).

There are many different types of problems that involve information leaks. Their severity can range widely depending on the type of information that is leaked. In addition, information leaks are often resultant.

****

INFO.LEAK.DIS. Discrepancy Information Leaks

Definition: a discrepancy information leak is an information leak in which the product behaves differently, or sends different responses, in a way that reveals security-relevant information about the state of the product, such as whether a particular operation was successful or not.

****

INFO.LEAK.DIS.RES. Response discrepancy infoleak

Definition: A response discrepancy information leak occurs when the product sends different messages in direct response to an attacker's request, in a way that allows the attacker to learn about the inner state of the product.

The leaks can be inadvertent (bug) or intentional (design).

Note: can overlap errors related to escalated privileges

Examples:

CVE-2002-2094 - this, and others, use ".." attacks and monitor error responses, so there is overlap with directory traversal.

CAN-2001-1483 - user enumeration by infoleak from inconsistent responses

CAN-2001-1528 - account number enumeration via inconsistent response infoleak

CAN-2004-2150 - user enumeration via error message discrepancy infoleak

CAN-2005-1650 - infoleak by inconsistent responses

Other examples: CAN-2004-0294, CAN-2004-0243, CAN-2002-0514, CAN-2002-0515, CAN-2001-1387, CAN-2004-0778, CAN-2004-1428

****

INFO.LEAK.DIS.BEH. Behavioral Discrepancy Infoleak

Definition: a behavioral discrepancy information leak occurs when the product's actions indicate important differences based on (1) the internal state of the product or (2) differences from other products in the same class.

Attacks such as OS fingerprinting rely heavily on both behavioral and response discrepancies.

****

INFO.LEAK.DIS.BEH.INT. Internal behavioral inconsistency infoleak

Definition: Two separate operations in a product cause the product to behave differently in a way that is observable to an attacker and reveals security-relevant information about the internal state of the product, such as whether a particular operation was successful or not.

Examples:

CAN-2002-2031 - file existence via infoleak monitoring whether "onerror" handler fires or not

CAN-2005-2025 - valid groupname enumeration via behavioral infoleak (sends response if valid, doesn't respond if not)

CAN-2001-1497 - behavioral infoleak in GUI allows attackers to distinguish between alphanumeric and non-alphanumeric characters in a password, thus reducing the search space

CAN-2003-0190 - product immediately sends an error message when

user does not exist instead of waiting until the password is provided, allowing username enumeration.

****

INFO.LEAK.BEH.EXT. External behavioral inconsistency infoleak

Definition: the product behaves differently than other products like it, in a way that is observable to an attacker and reveals security-relevant information about which product is being used, or its operating state.

Examples:

CAN-2002-0208 - product modifies TCP/IP stack and ICMP error messages in unusual ways that show the product is in use

CAN-2004-2252 - behavioral infoleak by responding to SYN-FIN packets

CVE-2000-1142 - honeypot generates an error with a "pwd" command in a particular directory, allowing attacker to know they are in a honeypot system

****

INFO.LEAK.TIM. Timing discrepancy infoleak

Definition: Two separate operations in a product require different amounts of time to complete, in a way that is observable to an attacker and reveals security-relevant information about the state of the product, such as whether a particular operation was successful or not.

Functional Area: cryptography, authentication

Attack: Timing attack

Note: overlaps crypto error, can be thought of as overlapping inconsistent response

Examples: CVE-2003-0078, CAN-2000-1117, CAN-2003-0637, CAN-2003-0190, CAN-2004-1602, CAN-2005-0918

****

INFO.LEAK.ERR.PGEN. Product-Generated Error Message Infoleak

Definition: the product identifies an error condition and creates its own diagnostic or error messages that contain sensitive information.

Functional Area: non-specific

Attack: trigger error, monitor responses

Examples: various

CAN-2005-1745 - infoleak of sensitive information in error message (physical access required)

****

INFO.LEAK.ERR.EXT. Product-External Error Message Infoleak

Definition: the product performs an operation that triggers a diagnostic or error message that is not under direct control of the product, e.g. an error generated by the programming language that the product uses.

This is inherently a resultant vulnerability from a WIFF within the product or an interaction error. It might be controllable by configuration, e.g. in PHP error messages.

Functional Area: non-specific

Attack: trigger error, monitor responses

Note: PHP applications are often targeted for having this issue when the PHP interpreter generates the error outside of the application's control. However, it's not just restricted to PHP, as other languages/environments exhibit the same issue.

Examples: CAN-2004-1581, CAN-2004-1579, CAN-2005-0459, CAN-2005-0443, CAN-2005-0433, CAN-2005-0326, CAN-2004-1101 (VisualBasic)

****

INFO.LEAK.CBC. Cross-Boundary Cleansing Infoleak

Definition: the product does not properly remove sensitive data from a source when preparing it for, or transferring it to, an untrusted destination.

Note: this is intended to be different from infoleaks that are resultant from initialization or reuse errors, although those could be viewed as cross-boundary. It could be regarded as a type of privacy leak. In some cases, it could be a resultant vulnerability, multiple interpretation error, or interaction error.

Some examples include Word or PDF files that did not remove sensitive supporting information, such as the edit history, when copying or exporting.

CAN-2005-0406 - some image editors modify a JPEG image, but the original EXIF thumbnail image intact within the JPEG. (Also an interaction error).

CVE-2002-0704 - NAT feature in firewall leaks internal IP addresses in ICMP error messages.

****

INFO.LEAK.INT. Intended information leak

Definition: a product's design or configuration includes functionality that is specifically designed to publish information that is security-sensitive.

Note: this overlaps other categories, but it is distinct from the error message infoleaks.

Note: it's not always clear whether an infoleak is intentional or not. For example, CVE-2005-3261 identifies a PHP script that lists file versions, but it could be that the developer did not intend for this information to be public, but introduced a direct request issue instead.

CAN-2002-1725, CVE-2004-0033, CAN-2003-1181, CAN-2004-1422, CAN-2004-1590 - script calls phpinfo()

CAN-2003-1038 - product lists DLLs and full pathnames

CAN-2005-1205, CAN-2005-0488 - Telnet protocol allows servers to obtain sensitive environment information from clients

****

INFO.LEAK.PRIVACY. Privacy Leak

Definition: a privacy leak occurs when a product exports information about the product's user, in which the information has no impact on the secure operation of the product itself, but the information is regarded as sensitive by the user.

****

INFO.LEAK.PROC. Process information infoleak to other processes

Certain information about a process could be obtained from other processes within the operating system, including arguments and environment variables. This can be an externally controlled infoleak, but some protective mechanisms may exist that could make it internally controlled.

Research Gaps: under-studied, especially environment variables.

CAN-2005-1387, CAN-2005-2291 - password passed on command line

CAN-2001-1565, CAN-2004-1948 - username/password on command line allows local users to view via "ps" or other process listing programs

CAN-1999-1270 - PGP passphrase provided as command line argument

CAN-2004-1058 - kernel race condition allows reading of environment variables of a process that is still spawning

****

INFO.LEAK.DEBUG. Infoleak Using Debug Information

Note: this overlaps other categories.

CAN-2004-2268 - debug information infoleak of password.

CAN-2002-0918 - CGI script includes sensitive information in debug messages when an error is triggered.

CAN-2003-1078 - FTP client with debug option enabled shows password to the screen.

****

INFO.MGT.UNCLEAR. Sensitive Information Uncleared Before Use

Definition: the product does not fully clear previously used information in a data structure, file, or other resource, before making that resource available to another party that did not have access to the original information.

Note: This typically involves memory in which the new data is not as long as the old data, which leaves portions of the old data still available ("memory disclosure"). However, equivalent errors can occur in other situations where the length of data is variable but the associated data structure is not.

Research gaps: currently frequently found for network packets, but it can also exist in local memory allocation, files, etc.

Functional Area: non-specific, memory management, networking

Note: can overlap cryptographic errors, cross-boundary cleansing infoleaks

Note: can be resultant from other WIFFs.

Examples:

CAN-2003-0001 - Ethernet NIC drivers do not pad frames with null bytes, leading to infoleak from malformed packets.

CAN-2003-0291 - router does not clear information from DHCP packets that have been previosuly used

CAN-2005-1406, CAN-2005-1858, CAN-2005-3180 - products do not fully clear memory buffers when less data is stored into the buffer than previous.

CVE-2005-3276 - product does not clear a data structure before writing to part of it, yielding information leak of previously used memory

CAN-2002-2077 - memory not properly cleared before reuse

****

INFO.MGT.COMP. Sensitive memory uncleared by compiler optimization

Definition: sensitive memory is cleared according to the source

code, but compiler optimizations leave the memory untouched when it is not read from again, ak "dead store removal."

Note: this is also an interaction error.

Note: this can be hard to diagnose.

References: [Howard2002] [Wagner]

****

INFO.LOSS. Information loss or omission

Definition: the product does not record, or improperly records, security-relevant information, e.g. for monitoring.

Note: these can be resultant vulns, e.g. a buffer overflow might trigger a crash before the product can log the event.

****

INFO.LOSS.TRUNC. Truncation of Security-relevant Information

Definition: The application truncates the display, recording, or processing of security-relevant information in a way that can obscure the source or nature of an attack.

CAN-2005-0585 - web browser truncates long sub-domains or paths, facilitating phishing

CAN-2004-2032 - bypass URL filter via a long URL with a large number of trailing hex-encoded space characters.

CAN-2003-0412 - does not log complete URI of a long request (truncation)

****

INFO.LOSS.OMIT. Omission of Security-relevant Information

Definition: The application does not record or display information that would be important for identifying the source or nature of an attack.

Examples:

CAN-1999-1029 - login attempts not recorded if user disconnects before maximum number of tries

CAN-2002-1839 - sender's IP address not recorded in outgoing e-mail

CVE-2000-0542 - failed authentication attempt not recorded if later attempt succeeds

****

INFO.LOSS.OBS. Obscured Security-relevant Information by Alternate Name

Definition: The product records security-relevant information according to an alternate name of the affected entity, instead of the canonical name.

CAN-2002-0725 - attacker performs malicious actions on a hard link to a file, obscuring the real target file.

==========

SECTION.9.10. [RACE] Race Conditions

==========

****

RACE.LINK. Race condition enabling link following

Note: this is already covered by PATH.LINK. It is included here because so many people associate race conditions with link problems; however, not all link following issues involve race conditions.

****

RACE.SIGNAL. Signal handler race condition

Functional Area: signals, interprocess communication

Note: probably under-studied.

Examples: CVE-2001-1349, CAN-2004-0794, CAN-2004-2259

****

RACE.TOCTOU. Time-of-check Time-of-use race condition

Definition: the product performs a verification check on an object, but the object (or its reference) can change before the product performs an operation on that object.

Note that TOCTOU issues do not always involve symlinks, not is every symlink issue a TOCTOU problem.

Non-symlink TOCTOU issues are not reported frequently, but they are likely to occur in code that attempts to be secure.

Examples: CAN-2003-0813, CAN-2004-0594, others

****

RACE.CSWITCH. Context Switching Race Condition

Definition: a product performs a series of non-atomic actions to switch between contexts that cross privilege or other security boundaries, but a race condition allows an attacker to modify or misrepresent the product's behavior during the switch.

Note: this is commonly seen in web browser vulnerabilities, in which the attacker can perform certain actions while the browser is transitioning from a trusted to an untrusted domain, or vice versa, and the browser performs the actions on one domain using the trust level and resources of the other domain.

Note: can be resultant or primary.

Note: can overlap signal handler race conditions.

Research Gaps: under-studied as a concept. Frequency unknown; few vulnerability reports give enough detail to know when a context switching race condition is a factor.

CAN-2004-2260 - browser updates address bar as soon as user clicks on a link instead of when the page has loaded, allowing spoofing by redirecting to another page using onUnload method. ** this is one example of the role of "hooks" and context switches, and should be captured somehow - also a race condition of sorts **

CVE-2004-0191 - XSS when web browser executes Javascript events in the context of a new page while it's being loaded, allowing interaction with previous page in different domain.

CVE-2004-2491 - web browser fills in address bar of clicked-on link

before page has been loaded, and doesn't update afterward.

****

RACE.ALTCHAN. Alternate Channel Race Condition

Note: this is already covered by CP.CHAN.ALT.RACE.

****

RACE.MISC. Other race conditions

Note: see "Alternate Channel Race Condition"

CAN-2005-2306 - race condition causes same token to be assigned to multiple sessions (same name property)

CAN-2005-1680 - authentication bypass by (1) insufficient access control (anyone from same IP address) or (2) "race condition" by being the first to access the software

CAN-2005-2174 - race condition allows infoleak

SECTION.9.11. [PPA] Permissions, Privileges, and ACLs

PPA.PRIV. Privilege / sandbox errors

A variety of vulnerabilities occur with improper handling, assignment, or management of privileges. These are especially present in sandbox environments, although it could be argued that any privilege problem occurs within the context of some sort of sandbox.

Note: can heavily overlap authorization errors

Research Gaps: many of the following concepts require deeper study. Most privilege problems are not classified at such a low level of detail, and terminology is very sparse. Certain classes of software, such as web browsers and software bug trackers, provide a rich set of examples for further research; operating systems have matured to the point that these kinds of WIFFs are rare.

****

PPA.PRIV.ASSIGN. Incorrect Privilege Assignment

   Definition: a product incorrectly assigns a privilege to a
   particular entity.

   Note: overlaps user management errors

   CVE-1999-1193 - untrusted user placed in unix "wheel" group

   CVE-2005-2741 - product allows users to grant themselves certain
   rights that can be used to escalate privileges

   CAN-2005-2496 - product uses group ID of a user instead of the
   group, causing it to run with different privileges. This is
   resultant from some other unknown issue.

    CVE-2004-0274 - product mistakenly assigns a particular status to an
    entity, leading to increased privileges

****

PPA.PRIV.UNS. Unsafe Privilege

   Definition: a particular privilege, role, capability, or right can
   can be used to perform unsafe actions that were not intended, even
   when it is assigned to the correct entity.

   Note: there are 2 separate sub-categories here:

     [*] privilege incorrectly allows entities to perform certain actions
     [*] object is incorrectly accessible to entities with a given
privilege

   This overlaps authorization and access control problems.

   Accessible entities:

     CAN-2002-1981 - roles have access to dangerous procedures

     CAN-2002-1671 - untrusted object/method gets access to clipboard

     CAN-2004-2204 - gain privileges using functions/tags that should

be restricted

CVE-2000-0315 - traceroute program allows unprivileged users to modify source address of packet

CAN-2004-0380 - bypass domain restrictions using a particular file that references unsafe URI schemes

CVE-2002-1154 - script does not restrict access to an update command, leading to resultant disk consumption and filled error logs.

Unsafe privileged actions:

CAN-2002-1145 - "public" database user can use stored procedure to modify data controlled by the database owner

CVE-2000-0506 - user with capability can prevent setuid program from dropping privileges

CAN-2002-2042 - allows attachment to and modification of privileged processes

CVE-2000-1212 - user with privilege can edit raw underlying object using unprotected method

CAN-2005-1742 - inappropriate actions allowed by a particular role

CAN-2001-1480 - untrusted entity allowed to access the system clipboard

CAN-2001-1551 - extra Linux capability allows bypass of system-specified restriction

CVE-2001-1166 - user with debugging rights can read entire process

CAN-2005-1816 - non-root admins can add themselves or others to the root admin group

CAN-2005-2173 - users can change certain properties of objects to perform otherwise unauthorized actions

CAN-2005-2027 - certain debugging commands not restricted to just the administrator, allowing registry modification and infoleak

****

PPA.PRIV.CHAIN. Privilege Chaining

   Definition: two distinct privileges, roles, capabilities, or rights
   can be combined in a way that allows an entity to perform unsafe
   actions that would not be allowed without that combination.

   Note: it is difficult to find good examples for this WIFF. There
   is some conceptual overlap with Unsafe Privilege.

   CAN-2005-1736 - chaining of user rights

   CAN-2002-1772 - gain certain rights via privilege chaining in
   alternate channel

   CAN-2005-1973 - application is allowed to assign extra permissions
   to itself

   CAN-2003-0640 - "operator" user can overwrite usernames and
   passwords to gain admin privileges

***

PPA.PRIV.MGT. Privilege Management Error

   Definition: a product does not properly grant, track, modify,
   record, or reset the privileges that are intended.

   CAN-2001-1555 - terminal privileges are not reset when a user logs out

   CAN-2001-1514 - does not properly pass security context to child
   processes in certain cases, allows privilege escalation

   CVE-2001-0128 - does not properly compute roles

***

PPA.PRIV.CONTEXT. Privilege Context Switching Error

   Definition: the product does not properly manage privileges while
   it is switching between different contexts that cross privilege
   boundaries.

   Note: this concept needs more study.

CAN-2002-1688, CAN-2003-1026 - web browser cross domain problem when user hits "back" button

CAN-2002-1770 - cross-domain issue - third party product passes code to web browser, which executes it in unsafe zone

CAN-2005-2263 - run callback in different security context after it has been changed from untrusted to trusted. * note that "context switch before actions are completed" is one type of problem that happens frequently, espec. in browsers.

****

PPA.PRIV.DROP. Privilege Dropping / Lowering Errors

CAN-2000-1213 - program does not drop privileges after acquiring the raw socket

CVE-2001-0559 - setuid program does not drop privileges after a parsing error occurs, then calls another program to handle the error

CVE-2001-0787, CVE-2002-0080 - does not drop privileges in related groups when lowering privileges

CVE-2001-1029 - does not drop privileges before determining access to certain files

CVE-1999-0813 - finger daemon does not drop privileges when executing programs on behalf of the user being fingered.

CVE-1999-1326 - FTP server does not drop privileges if a connection is aborted during file transfer

CVE-2000-0172 - program only uses setgeuid to drop privileges

CVE-2004-2504 - Windows program running as SYSTEM does not drop privileges before executing other programs (many others like this, especially involving the Help facility)

CAN-2004-0806 - setuid program does not drop privileges before executing program specified in an environment variable

CAN-2004-0828 - setuid program does not drop privileges before processing file specified on command line

CAN-2004-2070 - service on Windows does not drop privileges before using "view file" option, allowing code execution.

****

PPA.PRIV.INSUFF. Insufficient privileges

Definition: the product does not handle when it has insufficient privileges to perform an operation.

Note: overlaps dropped privileges, insufficient permissions

Note: can produce resultant WIFFs

CAN-2001-1564 - system limits are not properly enforced after privileges are dropped

CVE-2005-3286 - firewall crashes when it can't read a critical memory block that was protected by a malicious process

CAN-2005-1641 - does not give admin sufficient privileges to overcome otherwise legitimate user actions

****

PPA.PRIV.MISC. Misc. privilege issues

CAN-2005-2087 - browser follows references to non-standard objects

CAN-2001-1504 - automatically executing code in an email; sandboxing error?

CAN-2004-1121 - web browser allows modification of URL in the status bar via TABLE tags.

==========

PPA.PERM. Permission errors

Functional Area: file processing, non-specific

Note: overlaps insufficient privileges

****

PPA.PERM.DEF. Insecure Default Permissions

Definition: a program, upon installation, sets insecure permissions for an object.

CAN-2005-1941 - executables installed world-writable

CAN-2002-1713 - home directories installed world-readable

CAN-2001-1550 - world-writable log files allow information loss; world-readable file has cleartext passwords

CAN-2002-1711 - world-readable directory

CAN-2002-1844 - Windows product uses insecure permissions when installing on Solaris (genesis: port error)

CVE-2001-0497 - insecure permissions for a shared secret key file. Overlaps cryptographic problem.

CAN-1999-0426 - default permissions of a device allow IP spoofing

****

PPA.PERM.INH.ASSIGNED. Insecure Inherited Permissions

Definition: a product defines a set of insecure permissions that are inherited by objects that are created by the program.

Examples:

CAN-2005-1841 - user's umask is used when creating temp files

CAN-2002-1786 - insecure umask for core dumps [is the umask preserved or assigned?]

****

PPA.PERM.INH.PRESERVED. Insecure preserved inherited permissions

Definition: a product inherits a set of insecure permissions for an object, e.g. when copying from an archive file, without user awareness or involvement.

CAN-2005-1724 - does not obey specified permissions when exporting

****

PPA.PERM.ASSIGNED. Insecure execution-assigned permissions

Definition: a product, while it is executing, changes the permissions of an object in an insecure way that cannot be controlled by the user.

Examples: many, such as CVE-2002-0265, CAN-2003-0876

CAN-2002-1694 - log files opened read/write

****

PPA.PERM.INSUFF. Fails poorly due to insufficient permissions

Fault: unchecked error condition

Research Gaps: this type of issue is probably under-studied, since researchers often concentrate on whether an object has too many permissions, instead of not enough.

Examples:

CAN-2003-0501 - special file system allows attackers to prevent ownership/permission change of certain entries by opening them before calling setuid program

CVE-2004-0148

****

PPA.PERM.PRESERVE. Permission Preservation Failure

Definition: the product does not properly preserve permissions when copying, restoring, or sharing objects, which can cause them to have less restrictive permissions than intended.

Note: can be resultant.

Examples:

SUNALERT:27807

CAN-2001-1515 - automatic modification of permissions inherited from another file system

CAN-2005-1920 - permissions on backup file are created with defaults, possibly less secure than original file

CVE-2001-0195 - file is made world-readable when being cloned

==========

PPA.OWN. Ownership errors

Definition: the product assigns the wrong ownership, or does not properly verify the ownership, of an object or resource.

CAN-1999-1125 - program runs setuid root but relies on a configuration file owned by a non-root user.

****

PPA.OWN.UNVERIFIED. Unverified Ownership

Definition: the product does not properly verify that a critical resource is owned by the proper entity.

Note: this overlaps verification errors, permissions, and privileges.

Note: this can be a factor in other vulnerabilities.

CVE-2001-0178 - program does not verify the owner of a UNIX socket that is used for sending a password

CAN-2004-2012 - owner of special device not checked, allowing root

****

PPA.ACL. Access Control List (ACL) errors

Note: this item needs more work.

Possible sub-categories include:

[*] Trusted group includes undesired entities
[*] Group can perform undesired actions
[*] ACL parse error does not fail closed

****

PPA.USER. User management errors

   Note: this item needs more work.

   Possible sub-categories include:

     [*] user in wrong group
     [*] user with insecure profile / "configuration"

==
SECTION.9.12. [HAND] Handler Errors
==

Note: this concept is under-defined and needs more research.

****

HAND. Handler errors

   Note: may be resultant

****

HAND.WRONG. Improper Handler Deployment

   Definition: the wrong "handler" is assigned to process an object,
   e.g. calling a servlet to reveal source code of a .JSP file, or
   automatically "determines" type even if contradictory to an
   explicitly specified type).

   Factors: usually resultant.

   Note: can overlap Unrestricted File Upload.

   CVE-2001-0004 - source code disclosure via manipulated file
   extension that causes parsing by wrong DLL

   CVE-2002-0025 - web browser does not properly handle the
   Content-Type header field, causing a different application to
   process the document

   CAN-2000-1052 - source code disclosure by directly invoking a

servlet

CAN-2002-1742 - arbitrary Perl functions can be loaded by calling a non-existent function that activates a handler

****

HAND.MISSING. Missing Handler

Definition: a handler is not available or implemented.

****

HANDLER.DANG. Dangerous handler not cleared/disabled during sensitive operations

****

HAND.UNPARSED. Unparsed Raw Web Content Delivery

Definition: raw content or supporting code is stored under the web root with an extension that is not specially handled by the server such as ".inc" or ".pl", causing the content or code to be delivered to the user without the pre-processing that was expected, typically resulting in an information leak.

Note: this can overlap containment errors, but it is not necessarily the same thing.

Note: also overlaps direct requests, alternate path, permissions, sensitive file under web root

CAN-2002-1886, CAN-2002-2065, CAN-2005-2029, SECUNIA:11394 - ".inc" file stored under web document root and returned unparsed by the server

CVE-2001-0330 - direct request to .pl file leaves it unparsed

CAN-2002-0614 - .inc file

CVE-2004-2353 - unparsed config.conf file

****

HAND.UPLOAD. Unrestricted File Upload

Definition: the product allows the attacker to upload or transfer files of dangerous types that can be automatically processed within the product's environment.

Alternate term: formerly called "File Upload of Dangerous Type"

Note: this can overlap incomplete blacklist / permissive whitelist errors when the product tries, but fails, to properly limit which types of files are allowed.

Note: this can also overlap containment errors.

Note: this can also overlap multiple interpretation errors for intermediaries, e.g. anti-virus products that do not filter attachments with certain file extensions that can be processed by client systems.

Research Gaps: PHP applications are most targeted, but this likely applies to other languages that support file upload, as well as non-web technologies.

CVE-2001-0901 - web-based mail product stores ".shtml" attachments that could contain SSI

CAN-2002-1841 - PHP upload does not restrict file types

CAN-2005-1868 - upload and execution of .php file

CAN-2005-1881 - upload file with dangerous extension

CAN-2005-0254 - program does not restrict file types

CAN-2004-2262 - improper type checking of uploaded files

SECTION.9.13. [UI] User Interface Errors

Research Gaps: user interface errors that are relevant to security have not been studied at a high level.

****

UI.NOWARN. Product UI does not warn user of unsafe actions

Note: often resultant, e.g. in unhandler error conditions

Note: can overlap privilege errors, conceptually at least

Examples:

CVE-1999-1055, CVE-1999-0794, CVE-2000-0277 - product does not warn user when document contains certain dangerous functions or macros

CVE-2000-0517 - product does not warn user about a certificate if it has already been accepted for a different site. Possibly resultant.

CAN-2005-0602 - file extractor does not warn user it setuid/setgid files could be extracted. Overlaps privileges/permissions.

CVE-2000-0342 - e-mail client allows bypass of warning for dangerous attachments via a Windows .LNK file that refers to the attachment

****

UI.WARN.INSUFF. Insufficient UI warning of dangerous operations

Definition: a user interface provides a warning to a user regarding dangerous or sensitive operations, but the warning is not noticeable enough to warrant attention.

****

UI.INC. User interface inconsistency

Definition: a user interface - whether a GUI or not - behaves inconsistently with respect to the operations that are actually performed on the system, e.g. checking a security option does nothing, or the user tells the interface to "restrict ALL" when it is implemented as "restrict SOME".

Note: this is often resultant.

CAN-1999-1446 - UI inconsistency; visited URLs list not cleared when "Clear History" option is selected

****

UI-INC.UNIMP. Unimplemented or unsupported feature in UI

Definition: A UI function appears to be supported and gives

feedback to the user that suggests that it is supported, but the underlying functionality is not implemented.

CVE-2000-0127 - GUI configuration tool does not enable a security option when a checkbox is selected, although that option is honored when manually set in the configuration file.

CVE-2001-0863, CVE-2001-0865 - router does not implement a specific keyword when it is used in an ACL, allowing filter bypass

CAN-2004-0979 - web browser does not properly modify security setting when the user sets it.

****

UI.INC.OBS. Obsolete feature in UI

Definition: A UI function is obsolete and the product does not warn the user.

****

UI.WRONGACT. The UI performs the wrong action

Definition: the UI performs the wrong action with respect to the user's request.

CAN-2001-1387 - network firewall accidentally implements one command line option as if it were another, possibly leading to behavioral infoleak.

CVE-2001-0081 - command line option correctly suppresses a user prompt but does not properly disable a feature, although when the product promtps the user, the feature is properly disabled

CAN-2002-1977 - product does not "time out" according to user specification, leaving sensitive data available after it has expired

****

UI.MULTINT. Multiple Interpretations of UI Input

Definition: The UI has multiple interpretations of user input but does not warn the user, or selects the less secure interpretation.

****

UI.MISREP. UI Misrepresentation of Critical Information

  Definition: the UI does not properly represent critical information
  to the user, allowing the information - or its source - to be
  obscured or spoofed. This is often a component in phishing attacks.

  Research Gaps: misrepresentation problems are frequently studied in
  web browsers, but there are no known efforts for categorizing these
  problems in terms of the shortcomings of the interface. In
  addition, many misrepresentation issues are resultant.

   Note: this category needs refinement.

   Overlaps Wheeler's "Semantic Attacks"

   Here are some examples of misrepresentation:

    [*] icon manipulation (making a .EXE look like a .GIF)
    [*] homographs: letters from different character sets/languages that
      look similar. The use of homographs is effectively a
      manipulation of a visual equivalence property.
    [*] a race condition can cause the UI to present the user with
      "safe" or "trusted" feedback before the product has fully
      switched context. The race window could be extended
      indefinitely if the attacker can trigger an error.
    [*] "Window injection" vulnerabilities (though these are usually
      resultant from privilege problems)
    [*] status line modification (e.g. CAN-2004-1104)
    [*] various other web browser issues.
    [*] GUI truncation (e.g. filename with dangerous extension not
      displayed to GUI because of truncation)
      - CAN-2004-2227 - GUI truncation enables information hiding
    [*] injected internal spaces (e.g. "filename.txt   .exe"
      - though this overlaps truncation
    [*] Also consider DNS spoofing problems - can be used for
misrepresentation

  CAN-2001-0398 - attachment with many spaces in filename bypasses
  "dangerous content" warning and uses different icon. Likely
  resultant.

  CAN-2001-0643 - misrepresentation and equivalence issue

  CAN-2005-0593 - lock spoofing from several different WIFFs

== wrong status / state notifier

CAN-2005-0143 - lock icon displayed when an insecure page loads a binary file loaded from a trusted site

CAN-2005-0144 - secure "lock" icon is presented for one channel, while an insecure page is being simultaneously loaded in another channel

CAN-2004-0761 - certain redirect sequences cause security lock icon to appear in web browser, even when page is not encrypted

CAN-2004-2219 - spoofing via multi-step attack that causes incorrect information to be displayed in browser address bar

== overlay ==

CAN-2004-0537 - wide "favorites" icon can overlay and obscure address bar

OSVDB:5703 - GUI overlay vulnerability (misrepresentation)

== visual distinction ==

CAN-2005-2271, CAN-2005-2272, CAN-2005-2273, CAN-2005-2274 - web browsers do not clearly associate a Javascript dialog box with the web page that generated it, allowing spoof of the source of the dialog. "origin validation error" of a sort?

CAN-2001-1410 - browser allows attackers to create chromeless windows and spoof victim's display using unproptected Javascript method.

CVE-2002-0197 - chat client allows remote attackers to spoof encrypted, trusted messages with lines that begin with a special sequence, which makes the message appear legitimate.

CAN-2005-0831 - product allows spoofing names of other users by registering with a username containing hex-encoded characters.

== visual truncation ==

CAN-2003-1025 - special character in URL causes web browser to truncate the user portion of the "user@domain" URL, hiding real domain in the address bar.

CAN-2005-0243 - chat client does not display long filenames in file

dialog boxes, allowing dangerous extensions via manipulations including (1) many spaces and (2) multiple file extensions.

CAN-2005-1575 - web browser file download type hiding using whitespace

CVE-2004-2530 - visual truncation in chat client using whitespace to hide dangerous file extension

CAN-2005-0590 - dialog box in web browser allows user to spoof the hostname via a long "user:pass" sequence in the URL, which appears before the real hostname.

OSVDB:6009 - GUI obfuscation (visual truncation) in web browser - obscure URLs using a large amount of whitespace. Note - "visual truncation" covers a couple variants.

CAN-2004-1451 - null character in URL prevents entire URL from being displayed in web browser

== miscellaneous ==

[step-based attack, GUI]
CAN-2004-2258 - password-protected tab can be bypassed by switching to another tab, then back to original tab

CAN-2005-1678 - dangerous file extensions not displayed

CVE-2002-0722 - web browser allows remote attackers to misrepresent the source of a file in the File Download dialogue box.

==========
SECTION.9.14. [INT] Interaction Errors
==========

Definition: An interaction error occurs when two entities work correctly when running independently, but they interact in ways when they are run together. This could apply to products, systems, components, etc.

Terminology Note: some use "Interaction Error" to describe products that behave according to specification. However, the PLOVER definition includes products that do not necessarily comply with standard specifications.

****

INT.MULT. Multiple Interpretation Error (MIE)

Alternate Term: Interpretation Conflict

Definition: Product A handles inputs or steps differently than Product B, which causes A to perform incorrect actions based on its perception of B's state.

Note: this is generally found in proxies, firewalls, anti-virus software, and other intermediary devices that allow, deny, or modify traffic based on how the client or server is expected to behave.

Reference: [Christey2005b], [PtacekNewsham]

The classic multiple interpretation flaws were reported in a paper that described the limitations of intrusion detection systems. [PtacekNewsham] showed that OSes varied widely in their behavior with respect to unusual network traffic, which made it difficult or impossible for intrusion detection systems to properly detect certain attacker manipulations that took advantage of the OS differences.

Another classic multiple interpretation error is the "poison null byte" [RFP], in which null characters have different interpretations in Perl and C, which have security consequences when Perl invokes C functions. Similar problems have been reported in ASP [Moore] and PHP.

Some of the more complex web-based attacks, such as HTTP request smuggling, also involve multiple interpretation errors.

Note: a comment on a way to manage these problems is in [Skoll].

Manipulations are major factors in MIEs, such as doubling, inconsistencies between related fields, and whitespace.

CAN-2005-1215 - bypass filters or poison web cache using requests with multiple Content-Length headers, a non-standard behavior.

CAN-2002-0485 - anti-virus product allows bypass via

Content-Type and Content-Disposition headers that are mixed case, which are still processed by some clients.

CAN-2002-1978, CAN-2002-1979 - FTP clients sending a command with "PASV" in the argument can cause firewalls to misinterpret the server's error as a valid response, allowing filter bypass.

CAN-2002-0637 - virus product bypass with spaces between MIME header fields and the ":" separator, a non-standard message that is accepted by some clients.

CAN-2002-1777 - AV product detection bypass using inconsistency manipulation (file extension in MIME Content-Type vs. Content-Disposition field)

CVE-2005-3310 - CMS system allows uploads of files with GIF/JPG extensions, but if they contain HTML, Internet Explorer renders them as HTML instead of images.

CVE-2005-4260 - interpretation conflict allows XSS via invalid "<" when a ">" is expected, which is treated as ">" by many web browsers.

CVE-2005-4080 - interpretation conflict (non-standard behavior) enables XSS because browser ignores invalid characters in the middle of tags.

****

INT.MULT.EXTRAFEAT. Extra Unhandled Features

Definition: B has features that A does not handle or model.

****

INT.BEH.CHANGE. Behavioral Change

Definition: A's behavior or functionality changes with a new version of A, or a new environment, which is not known (or manageable) by B.

Alternate Term: functional change

CAN-2002-1976 - Linux kernel 2.2 and above allow promiscuous mode using a different method than previous versions, and ifconfig is not aware of the new method (alternate path property).

CAN-2005-1711 - anti-virus product uses a defunct method in another product that does not return an error code, allowing viruses to avoid detection.

CAN-2005-1711 - product uses defunct method from another product that does not return an error code and allows detection avoidance

****

INT.BEH.VIO. Expected behavior violation

Definition: A feature, API, or function being used by a product behaves differently than the product expects.

Property: consistency

CAN-2003-0187 - inconsistency in support of linked lists causes program to use large timeouts on "undeserving" connections

CAN-2003-0465 - "strncpy" in Linux kernel acts different than libc on x86, leading to expected behavior difference - sort of a multiple interpretation error?

CVE-2005-3265 - buffer overflow in product stems to the use of a third party library function that is expected to have internal protection against overflows, but doesn't.

****

INT.PROXY. Unintended proxy/intermediary

Definition: a product can be used as an intermediary or proxy between an attacker and the ultimate target, so that the attacker can either bypass access controls or hide activities.

Property: Alternate Channel

CVE-1999-0168 - portmapper could redirect service requests from an attacker to another entity, which thinks the requests came from the portmapper.

CAN-2005-0315 - FTP server does not ensure that the IP address in a PORT command is the same as the FTP user's session, allowing port scanning by proxy.

CAN-2002-1484 - web server allows attackers to request a URL from another server, including other ports, which allows proxied scanning.

CAN-2004-2061 - CGI script accepts and retrieves incoming URLs

CAN-2002-1484 - server in debug mode allows remote attackers to use it as an intermediary for port scanning via a request for a URL that specifies the target IP address and port, then monitoring the resulting error message

CAN-2001-1484 - MFV - bounce attack allows access to TFTP from trusted side

CVE-1999-0017 - FTP bounce attack. Protocol allows attacker to modify the PORT command to cause the FTP server to connect to other machines besides the attacker's. Similar to proxied trusted channel.

****

INT.WEB.HTTP-SPLIT. HTTP response splitting

Factors: resultant from CRLF injection, primary to multiple interpretation error

Note that HTTP response splitting is probably only multi-factor in an environment that uses intermediaries.

CAN-2005-1951, CAN-2004-2146 - application accepts CRLF in an object ID, allowing HTTP response splitting

CAN-2004-1620, CAN-2004-1656, CAN-2004-1687 - HTTP response splitting via CRLF in parameter related to URL

CAN-2005-2060, CAN-2005-2065 - bulletin board allows response splitting via CRLF in parameter

CVE-2004-2512 - response splitting via CRLF in PHPSESSID.

****

INT.WEB.HTTP-SMUG. HTTP Request Smuggling

Note: request smuggling can be performed due to a multiple

interpretation error, where the target is an intermediary or monitor, via a consistency manipulation (Transfer-Encoding and Content-Length headers).

Note: resultant from CRLF injection.

CAN-2005-2088, CAN-2005-2089, CAN-2005-2090, CAN-2005-2091, CAN-2005-2092, CAN-2005-2093, CAN-2005-2094 - web servers allow requets smuggling via inconsistent Transfer-Encoding and Content-Length headers.

==========
SECTION.9.15. [INIT] Initialization and Cleanup Errors
==========

Note: most of these initialization errors are significant factors in other WIFFs. Researchers tend to ignore these, concentrating instead on the resultant WIFFs, so their frequency is uncertain, at least based on published vulnerabilities.

****

INIT.DEF. Insecure default variable initialization

Definition: The product, by default, initializes an internal variable with an insecure or less secure value than is possible.

Note: this overlaps other categories, probably should be split into separate items.

****

INIT.EXTINIT. External initialization of trusted variables or values

Note: overlaps Missing variable initialization, especially in PHP

Note: this is a significant factor in a number of resultant WIFFs.

Note: overlaps other categories, e.g. PHP

CVE-2000-0959 - does not clear dangerous environment variables, enabling symlink attack

CVE-2001-0033 - specify alternate configuration directory in environment variable, enabling untrusted path.

CVE-2001-0872 - dangerous environment variable not cleansed

CAN-2001-0084 - specify arbitrary modules using environment variable

****

INIT.FAIL. Non-exit on Failed Initialization

Definition: the product does not exit or otherwise modify its operation when security-relevant errors occur during initialization, such as when a configuration file has a format error.

Research Gaps: under-studied. These issues are not frequently reported, and it is difficult to find published examples.

CAN-2005-1345 - product does not trigger a fatal error if missing or invalid ACLs are in a configuration file

****

INIT.MISS. Missing Initialization

Definition: The product does not initialize critical variables, which causes the execution environment to use unexpected values.

Note: this WIFF is a major factor in a number of resultant WIFFs, especially in web applications that allow global variable initialization (such as PHP) with libraries that can be directly requested.

Research Gaps: it is highly likely that a large number of resultant WIFFs have missing initialization as a primary factor, but researcher reports generally do not provide this level of detail.

CAN-2005-2978 - product uses uninitialized variables for size and index, leading to resultant buffer overflow

CAN-2005-2109 - internal variable in PHP application is not initialized, allowing external modification.

CAN-2005-2193 - array variable not initialized in PHP application, leading to resultant SQL injection.

****

INIT.INC. Incorrect Initialization

  Note: might overlap default initialization

  CAN-2001-1471 - invalid value prevents certain variables from being initialized, leading to resultant diect code injection.

  CAN-2005-1036 - permission bitmap is not properly initialized, leading to resultant privilege elevation or DoS.

****

INIT.CLEANUP.TMP. Incomplete Cleanup

  Definition: the product does not properly "clean up" and remove temporary or supporting resources after they have been used.

  Alias: Insufficient Cleanup

  Note: overlaps other categories. Concept needs further development.

  Note: this could be primary (e.g. leading to infoleak) or resultant (e.g. resulting from unhandled error condition or early termination).

  Functional Area: file processing

  Note: overlaps other categories such as permissions and containment.

  CVE-2000-0552 - world-readable temporary file not deleted after use.

  CAN-2005-2293 - temporary file not deleted after use, leaking database usernames and passwords.

  CVE-2002-0788 - interaction error creates a temporary file that can not be deleted due to strong permissions

  CAN-2002-2066, CAN-2002-2067, CAN-2002-2068, CAN-2002-2069, CAN-2002-2070 - alternate data streams for NTFS files are not cleared when files are wiped (alternate channel / infoleak)

  CAN-2005-1744 - users not logged out when application is restarted after security-relevant changes were made.

SECTION.9.16. [RES] Resource Management Errors

Resource management errors can lead to consumption, exhaustion, etc.

Note: often a resultant vulnerability

****

RES.MEMLEAK. Memory leak

Definition: the product does not sufficiently track and release allocated memory after it has been used, which slowly consumes remaining memory. This is often triggered by improper handling of malformed data or unexpectedly interrupted sessions.

Note: this is often a resultant WIFF due to improper handling of malformed data or early termination of sessions.

Functional Area: memory management

Terminology Note: "memory leak" has sometimes been used to describe other kinds of issues, e.g. for information leaks in which the contents of memory are inadvertently leaked (CAN-2003-0400 is one such example of this terminology conflict).

CAN-2005-3119 - memory leak because function does not free() an element of a data structure.

CAN-2004-0427, CVE-2002-0574 - memory leak when counter variable is not decremented

CAN-2005-3181 - kernel uses wrong function to release a data structure, preventing data from being properly tracked by other code

CAN-2004-0222 - memory leak via unknown manipulations as part of protocol test suite

CVE-2001-0136 - memory leak via a series of the same command

****

RES.LEAK. Resource leaks

****

RES.LEAK.FILEDESC. UNIX file descriptor leak

   Definition: a process does not close sensitive file descriptors
   before invoking a child process, which allows the child to perform
   unauthorized I/O operations using those descriptors.

   Functional Area: program invocation

   CAN-2002-0767, CAN-2003-0740 - privileged file descriptor not
   closed before executing child process

   CVE-2000-0378, CAN-2004-2215 - does not properly close file
   descriptors after logout

   CAN-2005-0205 - file descriptor not closed, allowing DNS spoofing

****

RES.RELEASE. Improper resource shutdown

   Definition: a resource is not properly cleared and made available
   for re-use.

   Note: can be resultant from improper error handling or insufficient
   resource tracking.

   Note: overlaps memory leaks, asymmetric resource consumption,
   malformed input errors.

   Functional Area: non-specific

   Examples:

   CVE-1999-1127 - does not shut down named pipe connections if
   malformed data is sent

   CVE-2001-0830 - sockets not properly closed when attacker
   repeatedly connects and disconnects from server

   CVE-2002-1372 - return values of file/socket operations not
   checked, allowing resultant consumption of file descriptors

==========

RES.AMP. Asymmetric resource consumption (amplification)

Definition: an attacker can force a victim to consume more resources than should be allowed for the attacker's current level of access.

Functional Area: non-specific

Note: There are probably several sub-types besides these.

Note: Sometimes this is a factor in "flood" attacks, but other types of amplification exist.

****

RES.AMP.NETWORK. Network Amplification

Definition: a product or network sends more network traffic to a recipient (usually spoofed) than is warranted for the access level of the recipient.

Note: spoofing is often a factor. Applications that use UDP are typically targeted, although this problem can exist in other protocols or contexts.

Note: network amplification, when performed with spoofing, is normally a multi-channel attack from attacker (acting as user) to amplifier, and amplifier to user.

CVE-1999-0513 - Smurf attack, spoofed ICMP packets to broadcast addresses.

CVE-1999-1379 - DNS query with spoofed source address causes more traffic to be returned to spoofed address than was sent by the attacker.

CVE-2000-0041 - large datagrams are sent in response to malformed datagrams.

CAN-1999-1066 - game server sends a large amount of traffic in response to an initial connection request.

****

RES.AMP.ALG. Algorithmic Complexity

Definition: an algorithm in a product has an inefficient worst-case

computational complexity that can be triggered by an attacker, typically using crafted manipulations that ensure that the worst case is being reached.

Note: the typical consequence is CPU consumption, but memory consumption and consumption of other resources can also occur.

Note: similar issues can occur in cryptography.

Reference: Algorithmic Complexity Attacks [Crosby]

CAN-2003-0244, CAN-2003-0364 - CPU consumption via inputs that cause many hash table collisions

CAN-2002-1203 - product performs unnecessary processing before dropping an invalid packet.

CAN-2001-1501 - CPU and memory consumption using many wildcards

CVE-2004-2527 - product allows attackers to cause multiple copies of a program to be loaded more quickly than the program can detect that other copies are running, then exit. This type of error should probably have its own category, where teardown takes more time than initialization.

CAN-2005-1792 - memory leak by performing actions faster than the software can clear them

****

RES.AMP.EARLY. Early Amplification

Definition: the product allows an entity to perform a legitimate but expensive operation before the entity has proven that the operation should be allowed.

Note: overlaps authentication errors.

CVE-2004-2458 - tool creates directories before authenticating user. general class of issue? step problem on product's side.

****

RES.AMP.DATA. Data Amplification

Definition: the product does not properly handle a compressed input with a very high compression ratio that produces a large output.

An example of data amplification is a "decompression bomb," a small ZIP file that can produce a large amount of data when it is decompressed.

==========

RES.POOL. Insufficient Resource Pool

Definition: the software's resource pool is not large enough to handle peak demand, which allows an attacker to prevent others from accessing the resource by using a (relatively) large number of requests for resources. Frequently the consequence is a "flood" of connection or sessions.

Functional Area: non-specific

Note: "large" is relative to the size of the resource pool, which could be very small. See examples.

Note: floods often cause a crash or other problem besides denial of the resource itself; these are likely examples of *other* vulnerabilities, not an insufficient resource pool.

CVE-1999-1363 - large number of locks on file exhausts the pool and causes crash

CAN-2001-1340 - product supports only one connection and does not disconnect a user who does not provide credentials

CVE-2002-0406 - large number of connections without providing credentials allows connection exhaustion

==========

RES.LOCK.UNRES. Unrestricted Critical Resource Lock

Definition: a critical resource can be locked or controlled by an attacker, indefinitely, in a way that prevents access to that resource by others, e.g. by obtaining an exclusive lock or mutex, or modifying the permissions of a shared resource.

Note: this overlaps Insufficient Resource Pool when the "pool" is of size 1. It can also be resultant from race conditions, although the

timing window could be quite large in some cases.

CVE-2001-0682, CAN-2002-1914, CAN-2002-1915 - program can not execute when attacker obtains a lock or mutex.

CVE-2000-0338 - predictable file names used for locking, allowing attacker to create the lock beforehand. Resultant from permissions and randomness.

CAN-2000-1198 - lock files with predictable names. Resultant from randomness.

CVE-2002-0051 - overlaps permissions, large-window race condition. Critical file can be opened with exclusive read access by user.

CVE-2002-1914 - users prevent execution of a dump program by locking a file

CAN-2002-1869 - product does not check if it can write to a log file, allowing attackers to avoid logging by accessing the file using an exclusive lock. Overlaps unchecked error condition.

****

RES.LOCK.INSUFF. Insufficient Resource Locking

Definition: a product does not completely lock access to a resource, in a way that either (1) allows an attacker to simultaneously access those resources, or (2) causes other errors that lead to a resultant WIFF. This can be due to unusual conditions, inability to detect when locking should occur, or incomplete actions.

Note: can be primary or resultant.

Examples:

CVE-2002-0638 - temporary file not properly locked when modifying critical file, allowing privilege escalation using a complex race condition.

CAN-2002-1749 - product doesn't properly lock itself if left idle

CAN-2002-1933 - window not locked if it's minimized

CAN-2005-2019 - certain resources are not sufficiently locked, allowing race condition, causing data corruption, and enabling

restriction bypass.

****

RES.LOCKCHECK. Missing Lock Check

Definition: a product does not check to see if a lock is present before performing sensitive operations on a resource.

CAN-2004-1056 - product does not properly check if a lock is present, allowing other attackers to access functionality

SECTION.9.17. [NUM] Numeric Errors

NUM.OBO. Off-by-one Error

Definition: a product uses an incorrect maximum or minimum value that is 1 more, or 1 less, than the correct value.

Resultant: can produce resultant buffer overflows

Note: this is not always a buffer overflow. For example, an off-by-one error could be a factor in a partial comparison, a read from the wrong memory location, an incorrect conditional, etc.

Research Gaps: under-studied. It requires careful code analysis or black box testing, where inputs of excessive length might not cause an error. Off-by-ones are likely triggered by extensive fuzzing, with the attendant diagnostic problems.

Terminology: an "off-by-five" error was reported for sudo in 2002 (CVE-2002-0184), but that is more like a "length calculation" error.

References: [Flake2001], [Christey2004a], [klog1999], [McHog] "buffer overflow" chapter)

Examples: CAN-2003-0466, CAN-2003-0252, CAN-2003-0625, CVE-2001-1391, CVE-2002-0083, CVE-2002-0653, CVE-2002-0844, CVE-1999-1568, CAN-2004-0346, CAN-2004-0005, CAN-2003-0356, CAN-2001-1496, CAN-2004-0342 (this is an interesting example that might not be an off-by-one), CAN-2001-0609 (an

off-by-one enables a terminating null to be overwritten, which causes 2 strings to be merged and enable a format string)

CAN-2002-1745 - off-by-one error allows source code disclosure of files with 4 letter extensions that match an accepted 3-letter extension.

CAN-2002-1816 - off-by-one buffer overflow

CAN-2002-1721 - off-by-one error causes an snprintf call to overwrite a critical internal variable with a null value.

CAN-2003-0466 - off-by-one error in function used in many products leads to a buffer overflow during pathname management, as demonstrated using multiple commands in an FTP server

CAN-2003-0625 - off-by-one error allows read of sensitive memory via a malformed request

****

NUM.SIGN. Integer Signedness Error (aka "signed integer" error)

Definition: a signed integer is cast to an unsigned integer in a manner that has security implications. Generally, this occurs when the attacker provides an input that contains a negative signed integer, which is cast to a large positive unsigned integer.

Reference: [Younan2003], section 5.4.3; [blexim], chapter 3

Functional Area: non-specific, memory management

Note: Some signedness errors arise as a result of positive inputs that are used in mathematical calculations that produce a negative value. Others involve comparisons in a signed context ("signed comparison") that fail to account for the use of the value in an unsigned context (e.g. comparing a negative signed integer to a positive "maximum" signed integer when that negative integer is later cast to an unsigned value that is greater than the maximum). It is not clear whether these should be treated as separate WIFFs.

Note: buffer overflows and array index errors can be resultant vulnerabilities.

Terminology Note: signedness errors are sometimes referred to as

integer overflows. Since they can also lead to buffer overflows, they may be referred to as buffer overflows.

Note: there can be cases in which an integer signedness error leads to an integer overflow, e.g. if an application reads a -1 value in a signed context but increments that value in an unsigned context (e.g. when allocating "input+1" bytes of memory).

Note: there are likely some cases in which providing a negative integer is not necessarily a signedness error, but still security relevant; consider the "$var[-1]" construct in Perl, which identifies the last element of the array.

Research Gaps: research seems to have concentrated exclusively on the security implications of using signed negative integers in an unsigned context, but there has been little or no work on the implications of using unsigned integers in a signed context (though it may be possible, e.g. if an unsigned is explicitly cast to a signed).

CVE-2001-0653 - large value in argument allows memory modification when argument is interpreted as a negative number

CAN-2003-0166 - integer signedness error in memory allocation function for interpreter allows memory consumption or arbitrary code via negative arguments to interpreter's API functions

CAN-2005-1402 - negative value not caught by maximum value signed comparison, later used in memory allocation, triggering memory consumption or crash (presumably from unhandled error condition)

CAN-2004-0661 - Integer signedness error in DHCP product allows making long DHCP lease (13 years) via -1 LEASETIME option.

CAN-2003-0619 - integer signedness error leads to kernel panic via negative size value.

****

NUM.OVERFLOW. Integer overflow (wrap or wraparound)

Definition: integer overflow: two values are added together such that they exceed the maximum integer value (MAXINT), which produces a value that is not equal to the correct result. This can happen in signed and unsigned cases; in the unsigned case on a 32 bit system, adding 1 to 0xffffffff leads to 0, whereas in the signed case,

adding 1 to 0x7fffffff changes the signed value from 2147483647 to -2147483648.

Reference: [Younan2003], section 5.4.3; [blexim], chapter 3

Functional Area: non-specific, memory management, counters

Terminology Note: "integer overflow" is used to cover several types of errors, including signedness errors, or buffer overflows that involve manipulation of integer data types instead of characters. Part of the confusion results from the fact that 0xffffffff is -1 in a signed context.

Note: integer overflows can be primary to buffer overflows

CVE-2002-0391 - integer overflow via a large number of arguments

CAN-2005-1141 - image with large width and height leads to integer overflow

CAN-2005-0102, CAN-2004-2013 - length value of -1 leads to allocation of 0 bytes and resultant heap overflow

***

NUM.UNDERFLOW. Integer underflow (wrap or wraparound)

Definition: integer underflow: one value is subtracted from the other such that it is less than the minimum integer value, which produces a value that is not equal to the correct result. This can happen in signed and unsigned cases.

Terminology Note: "integer underflow" is sometimes used to identify signedness errors in which an originally positive number becomes negative as a result of subtraction. However, there are cases of bad subtraction in which unsigned integers are involved, so it's not always a signedness issue.

Terminology Note: "integer underflow" is occasionally used to describe array index errors in which the index is negative.

Research Gaps: under-studied.

CAN-2004-0816 - integer underflow in firewall via malformed packet

CAN-2004-1002 - integer underflow by packet with invalid length

CAN-2005-0199 - long input causes incorrect length calculation

CAN-2005-1891 - malformed icon causes integer underflow in loop counter variable

***

NUM.TRUNC. Numeric truncation error

Definition: the product truncates a number, e.g. due to casting or other conversion between numeric types, in a way that has security implications.

Research Gaps: under-studied and under-reported.

***

NUM.BYTEORD. Numeric Byte Ordering Error

Definition: the product mixes up the order in which bytes are processed (e.g. big-endian and little-endian), causing a wrong number in a security-critical context.

Note: under-reported, but probably not likely to occur frequently, as byte ordering bugs are usually very noticeable even with normal inputs. This bug is more likely to occur in rarely triggered error conditions.

==========

SECTION.9.18. [AUTHENT] Authentication Error

==========

Definition: the product does not properly ensure that the user has proven their identity.

Consequence: authentication bypass

Terminology: an alternate term is "authentification", which appears to be most commonly used by people from non-English-speaking countries.

Note: this can be resultant from SQL injection vulnerabilities and other issues.

Functional Area: authentication

****

AUTHENT.ALTPC. Authentication Bypass by Alternate Path/Channel

Definition: a product requires authentication, but the product has an alternate path or channel that does not require authentication.

Note: this is often seen in web applications that assume that access to a particular CGI program can only be obtained through a "front" screen. But this problem is not just in web apps.

Note: overlaps Unprotected Alternate Channel

Examples: CVE-2000-1179, CAN-1999-1454, CVE-2000-0944, CAN-1999-1077, CAN-2003-1035 (overlaps brute force), CAN-2003-0304, CAN-2002-0870, CAN-2004-0213 (non-web), many web applications

CVE-2002-0066 - bypass authentication via direct request to named pipe

CAN-2003-1035 - user can avoid lockouts by using an API instead of the GUI to conduct brute force password guessing

****

AUTHENT.ALTNAME. Authentication bypass by alternate name

Definition: the software performs authentication based on the name of the resource being accessed, but there are multiple names for the resource, and not all names are checked.

Note: overlaps equivalent encodings, canonicalization, authorization, multiple trailing slash, trailing space, mixed case, and other equivalence issues.

Note: "alternate name" itself is a rather general class of data-driven manipulation.

Examples: CAN-2003-0317, CAN-2004-0847

****

AUTHENT.SPOOF. Authentication bypass by spoofing

Note: resultant vuln from insufficient verification

****

AUTHENT.REPLAY. Authentication bypass by replay

****

AUTHENT.MITM. Man-in-the-middle (MITM)

****

AUTHENT.MAID. Authentication Bypass via Assumed-Immutable Data

Definition: the authentication scheme or implementation uses key data elements that are assumed to be immutable, but can be controlled or modified by the attacker, e.g. if a web application relies on a cookie "Authenticated=1"

Examples: CVE-2002-0367 (DebPloit), CVE-2004-0261 (web auth)

CAN-2002-1730, CAN-2002-1734 - authentication bypass by setting certain cookies to "true"

CAN-2002-2064 - admin access by setting a cookie

CAN-2002-2054 - gain privileges by setting cookie

CAN-2004-1611 - product trusts authentication information in cookie

CAN-2005-1708 - authentication bypass by setting admin-testing variable to true.

CAN-2005-1787 - bypass auth and gain privs by setting a variable

***

AUTHENT.LOGIC. Authentication Logic Error

Examples: CAN-2003-0750 (conditional should have been an 'or' not an 'and')

****

AUTHENT.STEPMISS. Missing Critical Step in Authentication

Note: this overlaps insufficient verification.

CAN-2004-2163 - shared secret not verified in a RADIUS response packet, allowing authentication bypass by spoofing server replies.

****

AUTHENT.RESULTANT. Authentication Bypass by Primary WIFF

Definition: the authentication algorithm is sound, but the implemented mechanism can be bypassed as the result of a separate WIFF that is primary to the authentication error.

Note: most "authentication bypass" errors are resultant, not primary.

Examples: CVE-2002-1374, CVE-2000-0979, CAN-2001-0088

****

AUTHENT.NONE. No Authentication for Critical Function

Definition: the product does not perform any authentication for functionality that requires a provable user identity or consumes a significant amount of resources.

Note: this is separate from "bypass" issues in which authentication exists, but is faulty.

CAN-2002-1810 - MFV. Access TFTP server without authentication and obtain configuration file with sensitive plaintext information

****

AUTHENT.MULTFAIL. Multiple Failed Authentication Attempts not Prevented

Definition: the product does not implement sufficient measures to prevent multiple failed authentication attempts within in a short time frame, making it more susceptible to brute force attacks.

Note: common protection mechanisms include disconnecting a user, implementing a timeout, locking out a targeted account, or requiring a computational task on the user's part.

CAN-1999-1152, CVE-2001-1291, CAN-2001-0395, CAN-2001-1339, CAN-2002-0628 - product does not disconnect or timeout after

multiple failed logins

CVE-1999-1324 - user accounts not disabled when they exceed a threshold; possibly a resultant vuln

****

AUTHENT.MISC. Miscellaneous Authentication Errors

Note: these examples include multiple sub-categories that should be created.

CAN-2005-1680 - authentication bypass by (1) insufficient access control (anyone from same IP address) or (2) "race condition" by being the first to access the software

CAN-2001-1425 - challenge-response authentication allows remote attackers to compute the response based on observable information. Resultant from information leak.

CAN-2004-1685 - router allows authentication bypass by connecting to it from the same IP address as logged-in admin.

CVE-2004-2458 - tool creates directories before authenticating user. general class of issue? step problem on product's side.

CAN-2005-1831 - step-driven interruption attack allows authentication bypass

CVE-2005-3327 - authentication bypass by step-based manipulation (skipped step)

==========

SECTION.9.19. [CRYPTO] Cryptographic errors

==========

Note: this category is incomplete and needs refinement, as there is good documentation of cryptographic flaws and related attacks.

Note: some of these can be resultant.

Functional Area: cryptography

****

CRYPTO.PTEXT. Plaintext Storage of Sensitive Information

****

CRYPTO.PTEXT.DISK. Plaintext Storage in File or on Disk

   CAN-2001-1481 - plaintext credentials in world-readable file

   CAN-2005-1828, CAN-2005-2209 - password in cleartext in config file

   CAN-2002-1696 - decrypted copy of a message written to disk given a combination of options and when user replies to an encrypted message

   CVE-2004-2397 - plaintext storage of private key and passphrase in log file when user imports the key

****

CRYPTO.PTEXT.REG. Plaintext Storage in Registry

   CAN-2005-2227 - plaintext passwords in registry key

****

CRYPTO.PTEXT.COOKIE. Plaintext Storage in Cookie

   CAN-2002-1800 - admin password in plaintext in a cookie

   CAN-2001-1537 - default configuration has cleartext usernames/passwords in cookie

   CAN-2001-1536 - usernames/passwords in cleartext in cookies

   CAN-2005-2160 - authentication information stored in cleartext in a cookie

****

CRYPTO.PTEXT.MEM. Plaintext Storage in Memory

   Note: this could be a resultant WIFF, e.g. if the compiler removes code that was intended to wipe memory.

   Note: it could be argued that such problems are usually only exploitable by those with administrator privileges. However, swapping could cause the memory to be written to disk and leave it

accessible to physical attack afterwards.

CAN-2001-1517, BID:10155 - sensitive authentication information in cleartext in memory

CAN-2001-0984 - password protector leaves passwords in memory when window is minimized, even when "clear password when minimized" is set

CAN-2003-0291 - SSH client does not clear credentials from memory

****

CRYPTO.PTEXT.GUI. Plaintext Storage in GUI

CAN-2002-1848 - unencrypted passwords stored in GUI dialog may allow local users to access the passwords

****

CRYPTO.PTEXT.EXEC. Plaintext Storage in Executable

CAN-2005-1794 - product stores RSA private key in a DLL and uses it to sign a certificate, allowing spoofing of servers and MITM attacks.

****

CRYPTO.PTEXT.TRANS. Plaintext Transmission of Sensitive Information

CAN-2002-1949 - passwords transmitted in cleartext

****

CRYPTO.KEYMGT. Key Management Errors

Note: this category should probably be split into multiple sub-categories.

CAN-2005-2146 - insecure permissions when generating secret key, allowing spoofing

CAN-2001-1527 - administration passwords in cleartext in executable

CVE-2000-0762 - default installation of product uses a default encryption key, allowing others to spoof the administrator

== static key / global shared key ==

CAN-2002-1947 - "global shared key" - product uses same SSL key for all installations, allowing attackers to eavesdrop or hijack session

CVE-2005-4002 - "global shared key" - product uses same secret key for all installations, allowing attackers to decrypt data.

CAN-2005-2196 - product uses default WEP key when not connected to a known or trusted network, which can cause it to automatically connect to a malicious network. Overlaps: default

== end ==

== exposed or accessible private key ==

Note: overlaps information leak

CAN-2005-1794 - private key stored in executable

CVE-2001-0072 - crypto program imports both public and private keys but does not tell the user about the private keys, possibly breaking the web of trust

== misc ==

CAN-2005-3256 - encryption product accidentally selects the wrong key if the key doesn't have additional fields that are normally expected, leading to infoleak to the owner of that wrong key

****

CRYPTO.STEPMISS. Missing Required Cryptographic Step

Note: overlaps incomplete/missing security check

Note: can be resultant

Examples: BID:2356

****

CRYPTO.WEAKENC. Weak Encryption

Note: a variety of encryption algorithms exist, with various

weaknesses. This category could probably be split into smaller sub-categories.

CAN-2001-1546 - weak encryption

CAN-2004-2172 - weak encryption (chosen plaintext attack)

CAN-2002-1682 - weak encryption

CAN-2002-1697 - weak encryption produces same ciphertext from the same plaintext blocks

CAN-2002-1739 - weak encryption

CAN-2005-2281 - weak encryption scheme

CAN-2002-1872 - weak encryption (XOR)

CAN-2002-1910 - weak encryption (reversible algorithm)

CAN-2002-1946 - weak encryption (one-to-one mapping)

CAN-2002-1975 - encryption error uses fixed salt, simplifying brute force / dictionary attacks (overlaps randomness)

****

CRYPTO.REVHASH. Reversible One-Way Hash

Definition: a hashing algorithm produces results that can allow an attacker to determine the original input - or generate an input that produces the same hash - using feasible brute force or custom attacks.

****

CRYPTO.MISC. Miscellaneous Crypto Problems

Examples: CVE-1999-0982, CVE-2000-0402, CAN-1999-1256, CAN-2002-0344, many others

CAN-2002-1762 - sensitive information stored in cleartext

CAN-2003-0987 - crypto error (integrity checking)

CAN-2005-1797 - timing attack on AES (Rijndael) as a result of

design limitations of S-boxes

CVE-2004-2524 - attacker can retrieve plaintext credentials by using accessible "encryption" routine on desired username, then sending the encrypted username. Product then sends back username/password in plaintext. Poor cryptography?

CAN-2002-1892 - username/pass stored in readable format after backup

SECTION.9.20. [RAND] Randomness and Predictability

The product may use insufficiently random numbers or values in a security context that requires unpredictable numbers.

Factors: can be primary to cryptographic errors, authentication errors, symlink following, information leaks, and others.

Functional Area: non-specific, cryptography, authentication, session management

****

RAND.ENT. Insufficient Entropy

Definition: the product uses an algorithm or scheme that produces insufficient entropy, leaving patterns or clusters of values that are more likely to occur than others.

Examples:

CAN-2001-0950 - insufficiently random data used to generate session tokens using C rand(). Also, for certificate/key generation, uses a source that does not block when entropy is low

****

RAND.SPACE. Small Space of Random Values

Definition: the number of possible random values is smaller than needed by the product, making it more susceptible to brute force attacks.

CAN-2002-0583 - product uses 5 alphanumeric characters for filenames of expense claim reports, stored under web root.

CVE-2002-0903 - product uses small number of random numbers for a code to approve an action, and also uses predictable new user IDs, allowing attackers to hijack new accounts.

CVE-2003-1230 - SYN cookies implementation only uses 32-bit keys, making it easier to brute force ISN

CAN-2004-0230 - complex predictability / randomness (reduced space)

****

RAND.SEED. PRNG Seed Error

A Pseudo-Random Number Generator can use seeds incorrectly, in various ways.

****

RAND.SEED.SAME. Same Seed in PRNG

Definition: a PRNG uses the same seed each time the product is initialized.

****

RAND.SEED.PRED. Predictable Seed in PRNG

Definition: a PRNG is initialized from a predictable seed, e.g. using process ID or system time.

****

RAND.SEED.SPACE. Small Seed Space in PRNG

Definition: a PRNG uses a relatively small space of seeds.

Note: overlaps predictable from observable state

Examples: CVE-2002-0872

****

RAND.PRED.STATE. Predictable from Observable State

Definition: a number or object is predictable based on observations that the attacker can make about the state of the system or network, such as time, process ID, etc.

Examples:

CVE-2002-0389, CVE-2001-1141

CVE-2000-0335 - DNS resolver library uses predictable IDs, which allows a local attacker to spoof DNS query results.

CAN-2005-1636 - MFV. predictable filename and insecure permissions allows file modification to execute SQL queries

****

RAND.PRED.PREV. Predictable Exact Value from Previous Values

Definition: an exact value or random number can be precisely predicted by observing previous values.

Examples: CVE-2002-1463

CVE-1999-0074 - Listening TCP ports are sequentially allocated, allowing spoofing attacks

CVE-1999-0077 - Predictable TCP sequence numbers allow spoofing.

CVE-2000-0335- DNS resolver uses predictable IDs, allowing a local user to spoof DNS query results.

****

RAND.PRED.RANGE. Predictable Value Range from Previous Values

Definition: a relatively small set of likely values or random numbers can be predicted, typically by observing previous values or general non-random patterns within the generator, and simplifying a brute force attack.

Examples: [Zalewski2001]

RAND.STATIC. Static Value in Unpredictable Context

Definition: The product's execution context requires or assumes that certain values must be variable and unpredictable, but the value is the same.

Note: may be primary or resultant.

Note: overlaps default configuration.

Note: this is often a factor in attacks on web browsers, in which known or predictable filenames become necessary to exploit browser vulnerabilities.

CAN-2002-0980 - component for web browser writes an error message to a known location, which can then be referenced by attackers to process HTML/script in a less restrictive context

==========

SECTION.9.21. [CODE] Code Evaluation and Injection

==========

Code can be highly portable, and it can be transferred from one endpoint to another for the purpose of automatic execution on the receiving endpoint. Vulnerabilities can arise if the code can be controlled or influenced by an untrusted source.

Research Gaps: Many of these WIFFs are under-studied, and terminology is not sufficiently precise.

****

CODE.EVAL. Direct Dynamic Code Evaluation ("Eval Injection")

Definition: The product allows inputs to be fed directly into a function (e.g. "eval") that dynamically evaluates and executes the input as code, usually in the same interpreted language that the product uses.

Alternate term: direct code injection

Factors: special character errors can play a role in increasing the variety of code that can be injected, although some vulnerabilities do not require special characters at all, e.g. when a single function without arguments can be referenced and a terminator character is not necessary.

CAN-2002-1750, CAN-2002-1751, CAN-2002-1752, CAN-2002-1753, CAN-2005-1527, CAN-2005-2837 - direct code injection into Perl eval function

CAN-2005-1921, CAN-2005-2498 - MFV. code injection into PHP eval statement using nested constructs that should not be nested.

CVE-2005-3302 - code injection into Python eval statement from a field in a formatted file.

CAN-2001-1471 - MFV. invalid value prevents initialization of variables, which can be modified by attacker and later injected into PHP eval statement.

****

CODE.STAT. Direct Static Code Injection

Definition: The product allows inputs to be fed directly into an output file that is later processed as code, e.g. a library file or template.

Note: "HTML injection" (see XSS) could be thought of as an example of this, but it is executed on the client side, not the server side. Server-Side Includes (SSI) are an example of direct static code injection.

CVE-2002-0495 - Perl code directly injected into CGI library file from parameters to another CGI program

CAN-2005-1876 - direct PHP code injection into supporting template file

CAN-2005-1894 - direct code injection into PHP script that can be accessed by attacker

CAN-2003-0395 - PHP code from User-Agent HTTP header directly inserted into log file implemented as PHP script.

****

CODE.STAT.SSI. Server-Side Includes (SSI) Injection

Definition: a web product allows the injection of sequences that cause the server to treat as server-side includes.

Note: this can be resultant from XSS/HTML injection because the same special characters can be involved. However, this is server-side code execution, not client-side.

****

CODE.STAT.PHPINC. PHP File Include

Definition: a PHP product uses "require" or "include" statements, or equivalent statements, that use attacker-controlled data to identify code or HTML to be directly processed by the PHP interpreter before inclusion in the script.

Alternate Term: PHP file inclusion

Note: this is frequently a functional consequence of other WIFFs. It is usually multi-factor with other factors (e.g. MAID), although not all inclusion bugs involve assumed-immutable data. Direct request WIFFs frequently play a role.

Overlaps: Untrusted search path, direct request

Research Gaps: other interpreted languages with "require" and "include" functionality could also product vulnerable applications, but as of 2005, PHP has been the focus.

Reference: [Clowes]

Note: can overlap directory traversal in local inclusion problems.

CAN-2004-0285, CAN-2004-0030, CVE-2004-0068, CAN-2005-2157, CAN-2005-2162, CAN-2005-2198 - modification of assumed-immutable configuration variable in include file allows file inclusion via direct request

CVE-2004-0128 - modification of assumed-immutable variable in configuration script leads to file inclusion

CAN-2005-1864, CAN-2005-1869, CAN-2005-1870 - PHP file inclusion

CAN-2005-2154 - PHP local file inclusion

CAN-2002-1704, CAN-2002-1707, CAN-2005-1964, CAN-2005-1681, CAN-2005-2086 - PHP remote file include

CAN-2004-0127, CAN-2005-1971 - directory traversal vulnerability in PHP include statement

CVE-2005-3335 - PHP file inclusion issue, both remote and local; local include uses ".." and "%00" characters as a manipulation, but many remote file inclusion issues probably have this vector.

==========
SECTION.9.22. [ERS] Error Conditions, Return Values, Status Codes
==========

Keywords: error code, status code, return code, return value, error checking

If a function in a product does not generate the correct return/status codes, or if the product does not handle all possible return/status codes that could be generated by a function, then security issues may result.

This type of problem is most often found in conditions that are rarely encountered during the normal operation of the product. Presumably, most bugs related to common conditions are found and eliminated during development and testing.

In some cases, the attacker can directly control or influence the environment to trigger the rare conditions.

Note: this WIFF is often primary to a variety of other WIFFs.

Research Gaps: many researchers focus on the resultant WIFFs and do not necessarily diagnose whether a rare condition is the primary factor. However, in 2005 it seems to be reported more frequently than in the past. This subject needs more study.

****

ERS.UNCH. Unchecked Return Value

Definition: the product does not check the return value from a function or other entity in a way that leads to a vulnerability.

Note: this falls into 2 sub-categories: (1) Unchecked Error Return Value and (2) Unchecked Valid Return Value. The former involves the inability of the product to detect and handle when an error occurs. The latter might exist when the product handles errors correctly but

does not account for all possible return values, thus missing valid (but rarely occurring) return values.

Alternate names: Unchecked return value, Unchecked error condition

CAN-2002-1870 - failure to check results of recv call leads to resultant heap corruption

CAN-2002-1952 - error return codes not checked for database operations, allowing authentication bypass if database errors occur

CAN-2005-2183 - long input (non-overflow) leads to unhandled error condition and resultant authentication bypass

CAN-2005-2244 - MFV. Buffer overflow conditions due to improperly handled error condition (memory allocation failure)

CVE-2000-0536 - authentication bypass when connecting host does not have a reverse DNS entry.

CVE-2005-2708 - unchecked return code in kernel leads to system panic under low memory conditions

CAN-2004-0427 - Kernel does not properly decrement a counter in certain error conditions, leading to resultant memory leak.

CVE-2004-2396 - unchecked return code with unknown consequences

****

ERS.MISSERR. Missing Error Status Code

Definition: the product encounters an error but does not return a status code or return value to indicate that an error has occurred.

Note: may be primary or resultant.

CVE-2004-0063 - function returns "OK" even if another function returns a different status code than expected, leading to accepting an invalid PIN number.

CVE-2002-1446 - error checking routine in PKCS#11 library returns "OK" status even when invalid signature is detected, allowing spoofed messages.

CAN-2002-0499 - kernel function truncates long pathnames without

generating an error, leading to operation on wrong directory.

CAN-2005-2459 - function returns non-error value when a particular erroneous condition is encountered, leading to resultant null dereference.

****

ERS.WRONGCODE. Wrong Status Code

Definition: a function or operation returns an incorrect value or status code that does not indicate an error, but causes the product to modify its behavior based on the incorrect result, in a way that leads to a vulnerability.

Note: this can produce resultant vulnerabilities, and it might overlap other categories.

CAN-2003-1132 - DNS server returns wrong response code for non-existent AAAA record, which effectively says that the domain is inaccessible

CAN-2001-1509 - hardware-specific implementation of system call causes incorrect results from geteuid

CAN-2001-1559 - system call returns wrong value, leading to a resultant null dereference

****

ERS.UNEXPCODE. Unexpected Status Code or Return Value

Definition: the product does not properly check when a function or operation returns a value that is legitimate for the function, but is not expected by the product.

Note: this can produce resultant vulnerabilities.

CAN-2004-1395 - certain packets (zero byte and other lengths) cause a recvfrom call to produce an unexpected return code that causes a server's listening loop to exit

CVE-2002-2124 - unchecked return code from recv() leads to infinite loop

CAN-2005-2553 - kernel function does not properly handle when a null is returned by a function call, causing it to call another function

that it shouldn't.

CAN-2005-1858 - memory not properly cleared when read() function call returns fewer bytes than expected

CVE-2000-0536, CAN-2001-0910 - bypass access restrictions when connecting from IP whose DNS reverse lookup does not return a hostname.

CVE-2004-2371 - game server doesn't check return values for functions that handle text strings and associated size values.

CAN-2005-1267 - resultant infinite loop when function call returns -1 value

****

ERS.UNREP. Silent Failure (Unreported Error Condition)

Definition: the product encounters an error condition but does not report it, leading to vulnerabilities.

Note: could be primary or resultant. Overlaps other categories related to error conditions.

CVE-2005-4342 - component silently fails instead of throwing an exception when another key component is disabled, allowing security bypass

==========

SECTION.9.23. [VER] Insufficient Verification of Data

==========

Definition: the product does not sufficiently verify the origin or authenticity of data, in a way that leads to a vulnerability.

Terminology Note: "origin validation" could fall under this

Common manipulations: spoofing, replay.

****

VER.OVE. Origin Validation Error

Definition: the product does not properly verify that the source of data or communication is valid.

Note: this is a factor in many WIFFs, both primary and resultant. The problem could be due to design or implementation. This is a fairly general class.

Examples:

CAN-2000-1218, CAN-2005-0877 - DNS server can accept DNS updates from hosts that it did not query, leading to cache poisoning

CAN-2001-1452 - DNS server caches glue records received from non-delegated name servers

CAN-2005-2188 - user ID obtained from untrusted source (URL)

CAN-2003-0174 - LDAP service does not verify if a particular attribute was set by the LDAP server

CAN-1999-1549 - product does not sufficiently distinguish external HTML from internal, potentially dangerous HTML, allowing bypass using special strings in the page title. Overlaps special elements.

CAN-2003-0981 - product records the reverse DNS name of a visitor in the logs, allowing spoofing and resultant XSS.

****

VER.SIG. Improperly Verified Signature

Definition: the product does not verify, or improperly verifies, the cryptographic signature for data.

CAN-2002-1796 - does not properly verify signatures for "trusted" entities

CAN-2005-2181, CAN-2005-2182 - insufficient verification allows spoofing

CAN-2002-1706 - accepts a configuration file without a Message Integrity Check (MIC) signature

****

VER.LTRUST. Use of Less Trusted Source

Definition: the product has two different sources of the same data or information, but it uses the source that has less support for verification, is less trusted, or is less resistant to attack.

CVE-2001-0860 - product uses IP address provided by a client, instead of obtaining it from the packet headers, allowing easier spoofing.

CAN-2004-1950 - web product uses the IP address in the X-Forwarded-For HTTP header instead of a server variable that uses the connecting IP address, allowing filter bypass.

BID:15326 - similar to CAN-2004-1950

CAN-2001-0908 - product logs IP address specified by the client instead of obtaining it from the packet headers, allowing information hiding.

CVE-2006-1126 - PHP application uses IP address from X-Forwarded-For HTTP header, instead of REMOTE_ADDR.

****

VER.UDAPP. Untrusted Data Appended with Trusted Data

Definition: The product, when processing trusted data, accepts any untrusted data that is also included with the trusted data.

CVE-2002-0018 - does not verify that trusted entity is authoritative for all entities in its response

****

VER.DNSREV. Improperly Trusted Reverse DNS

Definition: the product trusts the hostname that is provided when performing a reverse DNS resolution on an IP address, without also performing forward resolution.

CAN-2001-1488 - does not do double-reverse lookup to prevent DNS spoofing

CAN-2001-1500 - does not verify reverse-resolved hostnames in DNS

CAN-2000-1221, CVE-2002-0804 - authentication bypass using spoofed reverse-resolved DNS hostnames

CVE-2001-1155 - filter does not properly check the result of a reverse DNS lookup, which could allow remote attackers to bypass intended access restrictions via DNS spoofing.

CAN-2004-0892 - reverse DNS lookup used to spoof trusted content in intermediary

CAN-2003-0981 - product records the reverse DNS name of a visitor in the logs, allowing spoofing and resultant XSS.

****

VER.INSUFF-VERIFY.TYPE. Insufficient Type Distinction

Definition: the product does not properly distinguish between different types of elements in a way that leads to insecure behavior.

Note: overlaps others, e.g. Multiple Interpretation Errors.

CAN-2005-2260 - browser user interface does not distinguish between user-initiated and synthetic events

CVE-2005-2801 - Product does not compare all required data in two separate elements, causing it to think they are the same, leading to loss of ACLs. Similar to Same Name error.

****

VER.INTEG.MISS. Missing Integrity Check

Definition: the product does not perform an integrity check that is required by its design; or, the product's design does not include an integrity check for critical data or resources.

Note: overlaps origin validation error

CVE-2002-0671, CVE-2002-0676, CAN-2001-1125, CAN-2003-0237 - product downloads executables from a web site but does not verify integrity of the executables, allowing malicious injection using DNS spoofing

****

VER.INTEG.INC. Incomplete Integrity Check

Definition: the product does not perform all steps of an integrity check that is required by its design; or, the product's design does not provide sufficient steps in an integrity check for critical data or resources.

Note: overlaps origin validation error, Non-conformant API Usage

Note: examples are not currently available although such problems have been reported and may be covered by other PLOVER categories.

****

VER.WEB.CSRF. Cross-Site Request Forgery (CSRF)

Definition: the web product does not, or can not, sufficiently verify whether a well-formed, valid, consistent request was intentionally provided by the user who submitted the request.

Note: CSRF is multi-channel:

1. Attacker-to-victim (injection; external or internal channel)

2. Victim-to-server (activation; internal channel)

The associated WIFF is Insufficient Verification of Data.

Note: could be resultant from XSS, although XSS is not necessarily required.

Reference: [PeterW]

Examples: CAN-2004-1703, CAN-2004-1995, CAN-2004-1967, CAN-2004-1842, CAN-2005-1947, CAN-2005-2059

CAN-2005-1674 - CSRF

****

VER.PHP-UPLOAD. PHP Upload Verification

Note: this category needs work.

CAN-2002-1460 - PHP web forum does not properly verify whether a file was uploaded, allowing attackers to reference other files by modifying POST variables.

CAN-2002-1710 - product does not distinguish uploaded file from other files

CAN-2002-1759 - PHP script does not restrict access to uploaded files. Overlaps container error.

****

VER.OTHER. Other Insufficient Verification

CAN-2004-2163 - shared secret in a response is not verified, allowing authentication bypass using spoofing

CAN-2001-1568, CAN-2001-1569 - incomplete verification of certificates in WAP products allows SSL certifciate spoofing using man-in-the-middle attack

CAN-2002-1846 - product doesn't require user to provide correct old password when changing new password

CAN-2005-2145 - kernel driver doesn't verify the origin of certain messages, allowing attacker to disable certain warnings.

SECTION.9.24. [MAID] Modification of Assumed-Immutable Data

Definition: the product does not properly protect an assumed-immutable element from being modified by an attacker.

Factors: MAID issues can be primary to many other WIFFs, and they are a major factor in languages such as PHP.

Note: This happens when a particular input is critical enough to the functioning of the application that it should not be modifiable at all, but it is. A common programmer assumption is that certain variables are immutable; especially consider hidden form fields in web applications. So there are many examples where the MUTABILITY property is a major factor in a vuln.

Note: common data types that are attacked are environment variables, web application parameters, and HTTP headers.

CAN-2002-1757 - relies on $PHP_SELF variable for authentication

    CAN-2005-1905 - gain privileges by modifying assumed-immutable
    code addresses that are accessed by a driver

****

MAID.PTAMP. Web Parameter Tampering

  Definition: a web product does not properly protect
  assumed-immutable values from modification in hidden form fields,
  parameters, cookies, or URLs, which lead to modification of critical
  data.

  Alternate Term: Assumed-Immutable Parameter Tampering

  Note: this is a primary WIFF for many other WIFFs and functional
  consequences, including XSS, SQL injection, path disclosure, and
  file inclusion.

  Note: this is a technology-specific MAID problem.

  [SM2] can this be split into assumed-immutable vs. unverified?
  especially the case in direct request vulnerabilities in PHP
  applications - if the variable wasn't set from a $_GET or other
  variable, then it's probably assumed-immutable.

  CAN-2002-0108 - forum product allows spoofed messages of other
  users via hidden form fields for name and e-mail address.

  CVE-2000-0253, CVE-2000-0254, CVE-2000-0926, CAN-2000-0101,
  CAN-2000-0102 - shopping cart allows price modification via hidden
  form field

  CVE-2000-0758 - allows admin access by modifying value of form field

  CAN-2002-1880 - read messages by modifying message ID parameter

  CAN-2000-1234 - send email to arbitrary users by modifying email
  parameter

    CAN-2005-1652 - authentication bypass by setting a parameter

    CAN-2005-1784 - product does not check authorization for
    configuration change admnin script, leading to password theft via
    modified e-mail address field

CAN-2005-2314 - logic error leads to password disclosure

CAN-2005-1682 - modification of message number parameter allows attackers to read other people's messages

****

MAID.PHPVAR. PHP External Variable Modification

Definition: a PHP product does not properly protect against the modification of variables from external sources.

Note: this is a tech-specific instance of MAID.

Factors: this can be resultant from direct request (alternate path) issues. It can be primary to WIFFs such as PHP file inclusion, SQL injection, XSS, authentication bypass, and others.

CVE-2000-0860 - file upload allows arbitrary file read by setting hidden form variables to match internal variable names.

CAN-2001-0854 - mistakenly trusts $PHP_SELF variable to determine if include script was called by its parent

CAN-2002-0764 - PHP remote file inclusion by modified assumed-immutable variable.

CAN-2001-1025 - modify key variable when calling scripts that don't load a library that initializes it.

CAN-2003-0754 - authentication bypass by modifying array used for authentication

==========

SECTION.9.25. [MAL] Product-Embedded Malicious Code

==========

Definition: the product, as delivered to the consumer, contains undocumented, hidden functionality or configuration that is specifically intended to secretly obtain access, sensitive data, or cause a denial of service when certain conditions are met.

Note: this is distinct from default, documented configuration that happens to be insecure, or intentionally embedded vulnerabilities. In

some cases, the lines can be blurred, and the developer's intentions can not be known.

Terminology Note: a web search suggests that the phrase "embedded malicious code" is commonly used as a synonym for "payload" in the context of exploits.

****

MAL.BDOOR. Back Door

  Definition: a hidden, undocumented alternate channel, alternate path, or alternate name in the product, as delivered to the consumer, that is specifically intended for outside entities to interact with the product without successfully passing through all security mechanisms.

  Keywords: back door, backdoor

  Note: sometimes it is unclear whether a "back door" issue is intentionally malicious, or just the result of a design error.

****

MAL.BDOOR.ACC. Back Door

  Definition: a hidden, undocumented account, typically hard-coded, that allows an attacker to obtain access using either (1) a special name or (2) a special password.

  CVE-2002-1272 - back door intended for development accidentally left enabled in production

****

MAL.BDOOR.ACC.DEV. Developer-Introduced Back Door

  CAN-2000-1230 - developer back door

  CAN-2002-1936 - default and back door accounts

  CAN-2000-0248 - back door

  CVE-2004-1884 - back door account in FTP server

****

MAL.BDOOR.HPASS. Hard-Coded Password or Account

  Definition: the product contains a hard-coded password or account
  that cannot be changed by the user through the normal interface.

  Note: the password could be documented or undocumented.

  CAN-2005-1837 - hard-coded username with predictable password
  obtained from product serial number

  CAN-2005-2026 - developer hard-coded account/password

  CAN-2005-1867 - hard-coded admin password

****

MAL.BDOOR.ACC.OUT. Outsider-Introduced Back Door

  Definition: a back door that is introduced by a party other than the
  developer, e.g. by an attacker at the product's distribution source.

  CAN-2002-1840, CAN-2002-2049 - non-developer introduced back door

****

MAL.HFUNC. Hidden User-Triggered Functionality

  Definition: the product contains functionality that is "hidden" but
  cannot be triggered by a user, i.e. the functionality is not
  accessible during well-formed, valid, consistent interactions.

  Note: the "Trojan Horse" and "Easter Egg" concepts are covered by
  this WIFF.

****

MAL.LOGBOMB. Logic Bomb

  Needs definition.

  Note: overlaps business rule violation.

****

MAL.TIMBOMB. Time Bomb

Needs definition.

Note: overlaps business rule violation.

SECTION.9.26. [ATTMIT] Common Attack Mitigation Failures

This section covers failures of the product design in protecting against widely used attacks. Most of these attacks require conditions or WIFFs that are already covered elsewhere. However, they are mentioned so frequently that there should be some place for them in PLOVER.

Note that these attacks involve particular manipulations; the underlying WIFFs can vary widely.

****

ATTMIT.REPLAY. Insufficient Replay Protection

Definition: the product does not use sufficient measures to prevent replay attacks from succeeding, e.g. randomness, integrity checking, timeouts, and data verification.

CAN-2004-2243 - session hijacking via replay

CAN-2002-2046 - authentication bypass by stealing and replaying MD5'd password

CAN-2002-1746 - sniff and replay

CAN-2001-1545 - session ID stored in URL allows theft and replay by HTTP referer or sniffing. NOTE: many web session ID vulns are MFV since there's (1) intentional infoleak and (2) replay.

CAN-2005-2185 - cookies do not expire and simplify replay attacks

CAN-2001-1505 - modification of user sessions by replaying packets

CAN-2005-1664 - web middleware allows replay attacks using state identifiers

CVE-2005-3435 - password hash replay; don't need to know original

password.

****

ATTMIT.BRUTE. Susceptibility to Brute Force Attack

Functional Areas: cryptography, authentication

****

ATTMIT.SPOOF. Susceptibility to Spoofing

Terminology Note: the "spoofing" term is used to describe a wide variety of attacks. The "forgery" term is under-used but might be able to provide some distinction. More investigation is needed.

Research Gaps: while spoofing attacks are frequently reported, there is little research into the underlying WIFFs that enable spoofing to be successful.

Note: overlaps insufficient verification and misrepresentation problems. Insufficient randomness, or predictability, is often a critical factor in spoofing attacks.

Other WIFFs are heavily involved in allowing spoofing to be successful. For example:

[*] GUI does not notify user of original origin of a message or window

[*] product does not verify the origin of the message or window

[*] product does not check all required fields

[*] underlying protocol design does not include verification

Examples:

CVE-1999-0395 - race condition allows an attacker to spoof a server

CAN-1999-1254 - OS allows DoS by spoofed ICMP redirect messages, causing OS to change its routing tables

CAN-1999-0667 - ARP protocol allows ARP replies to be spoofed

CAN-2005-2145 - information hiding by spoofing

CAN-2003-0552 - kernel allows spoofing of routing information via forged packets whose source address is the same as the target

CAN-2001-0323 - ICMP PMTU discovery feature allows DoS by spoofing "ICMP Fragmentation needed but Don't Fragment (DF) set" packets between two target hosts, which could cause one host to lower its MTU when transmitting to the other host.

CAN-1999-0195 - Denial of service in portmapper allows attackers to modify or spoof RPC services with spoofed source IP such as 127.0.0.1.

CAN-2004-0527, CAN-2004-0528 - web browser allows remote attackers to spoof a legitimate URL in the status bar via A HREF tags with modified "alt" values that point to the legitimate site, combined with an image map whose href points to the malicious site, which facilitates a "phishing" attack.

CAN-2004-0763 - spoof of certificates in web browser via redirects and Javascript that uses the "onunload" method.

CAN-2005-1746 - spoofed cookies to force contact with systems outside a trusted group

CAN-1999-1291 - connection reset by forcing a reset (RST) via a PSH ACK or other means, obtaining the target's last sequence number from the resulting packet, then spoofing a reset to the target.

==========
SECTION.9.27. [CONT] Containment errors (container errors)
==========

This tries to cover various problems in which improper data is included within a "container."

Note: this overlaps many other WIFFs. Most vulnerabilities could be regarded as "container" problems.

****

CONT.ACC. Sensitive Entity in Accessible Container

Definition: the product stores sensitive data, objects, code, or other entities in a directory or other container that is accessible

to an attacker.

Alternate term: containment error, container error

****

CONT.WEB. Sensitive Data Under Web Root

Factors: can be resultant from insecure permissions.

CAN-2002-2029 - executable interpreter under web root

CAN-2002-1909, CAN-2005-1647, CAN-2005-2005, CAN-2005-2192, CAN-2005-2189, CAN-2005-2229 - data file with authentication information (usernames, passwords, keys) accessible under web root

CAN-2005-2075 - database backup file stored under web root with predictable filename

CAN-2005-0229 - temporary data file with credit card information under web root

CAN-2003-0841 - temporary file in guessable directory name

CAN-2005-1716, CAN-2005-1733 - database under web root

****

CONT.FTP. Sensitive Data Under FTP Root

Example: various Unix FTP servers require a password file that is under the FTP root, due to use of chroot.

==========

SECTION.9.28. [MISC] Miscellaneous WIFFs

==========

This section includes other WIFFs that do not fit cleanly into previously specified categories.

****

MISC.DFREE. Double-Free Vulnerability

Definition: the product performs a free() operation on a pointer that it has already previously freed.

Note: this is usually resultant from another WIFF, such as an unhandled error or race condition between threads. It could also be primary to WIFFs such as buffer overflows.

Note: also a Consequence.

CAN-2004-0642, CAN-2004-0772, CAN-2005-1689 - double-free resultant from certain error conditions

CAN-2003-0545 - double-free from invalid ASN.1 encoding

CAN-2003-1048, CAN-2005-0891 - double-free from malformed GIF

CVE-2002-0059 - double-free from malformed compressed data

****

MISC.INVFREE. Free of Invalid Pointer

Definition: the product attempts to free memory associated with an invalid pointer.

Note: usually resultant; an atomic consequence.

Note: overlaps double-free, others.

CAN-2003-1201 - free of uninitialized pointer when function return code does not indicate success

CVE-2004-2486 - free of uninitialized variable

CVE-2005-3249 - free of invalid pointer

CVE-2005-3806 - free of wrong pointer. typo in variable name causes wrong memory to be freed - possibly unallocated - leading to memory corruption or DoS

MISC.ASSERT. Reachable Assert Failure

Definition: the product allows an attacker to manipulate the product state, possibly via direct inputs, that reach an assert statement that fails.

Note: this is effectively an atomic consequence. It is frequently resultant from a number of low-level WIFFs.

Examples:

CAN-2004-0931 - database allows crash via request with "high ASCII" values (requiring 8 bits>) in the server field, triggering an assert error.

CVE-2004-0270 - anti-virus product has assert error when line length is non-numeric

CVE-2005-0446 - assert error in web proxy from various responses

****

MISC.EXTCONF. Externally Required Insecure Design Conformance

Definition: the product is required to support an externally specified design, e.g. to support interoperability, but that design has one or more inherent vulnerabilities.

Note: it could be argued that this is primary to most other WIFFs.

CAN-2005-1646 - FTP protocol design flaw (FTP bounce) allowed in product due to insecure requirements of FXP, which is forced to be supported by product

CAN-2002-1968 - attacker can download DOCSIS configuration file from TFTP server on internal network side, then modify. Possibly standard-required behavior.

****

MISC.WEAK. Selection of Weaker Scheme

Definition: the product can choose between two schemes, algorithms, or protocols that meet all applicability requirements for a task, but the product selects the "weaker" scheme that is less resistant to attack.

Note: conceptually similar to Use of Less Trusted Source

Note: can be primary, but probably resultant in most cases.

Research Gaps: under-studied

Note: other examples exist but have not been identified yet.

CAN-2005-2395 - web browser chooses weakest authentication scheme available instead of the strongest, enabling leak of credentials in plaintext

****

MISC.NONCONFORM.IMP. Non-Conformant Implementation

Definition: the product does not follow the required security-relevant conventions when implementing a design-required algorithm, scheme, or protocol.

Resarch Gaps: under-studied.

Note: overlaps interaction errors.

Examples: many, spread throughtout other WIFFs.

****

MISC.NONCONFORM.API. Non-Conformant API Usage

[King] "API abuse"

Definition: the product does not follow the required conventions when using a specific API, in a way that leads to a vulnerability.

Alternate names: "API abuse" or "Convention Violation"

Terminology note: currently, there is not a good term to capture the concepts being described.

Note: conceptually, this can overlap a large number of other issues. It can be primary to many WIFFs, e.g. a "non-conformant API usage" of strcpy() leads to buffer overflows.

CAN-2003-0653 - kernel module does not use a required structure when sending certain error responses, leading to kernel panic.

CVE-2005-3181 - kernel uses wrong specialized function to free a structure, leading to memory leak

CVE-2003-0986 - kernel does not use required function when copying

data from user space to kernel space, allowing DoS.

****

MISC.CDEP. Client-Dependent Security Enforcement

Definition: the product trusts a client or other user-controlled resource to enforce security restrictions but does not protect against the use of a modified client that bypasses those restrictions.

Note: can be primary in many web application vulnerabilities, although the resultant issues are normally emphasized.

Note: overlaps parameter tampering, MAID.

****

MISC.SDIST. Incomplete Internal State Distinction

Definition: the product does not properly determine which state it is in, causing it to assume it is in state X when in fact it is in state Y, causing it to perform incorrect operations in a security-relevant manner.

Note: this conceptually overlaps other categories such as insufficient verification, but this refers to the product's "self-perception."

Note: probably resultant from other WIFFs such as unhandled error conditions, inability to handle out-of-order steps, multiple interpretation errors, etc.

****

MISC.INCACT. Incomplete Action

Definition: the product does not perform all steps of a particular task, or act on all relevant objects, or examine all relevant data

Alternate term: Partial Action

Note: overlaps other categories such as incomplete verification, missing steps, etc.

CVE-2004-0715 - product does not clear all member relationships when

a group is deleted, which can cause those members to be part of a new group that has the same name as the old group.

CVE-2004-2305 - AV product only scans password-protected file in ZIP file, skipping the other files in the ZIP

BID:6787 - anti-virus product only checks first 15K of a message, so virus can avoid detection by inserting malicious code after that.

****

MISC.TRUNC. Other Types of Truncation Errors

Buffer consumption (buffer truncation?)

CAN-2003-0748 - fill a filename with spaces so that a ".html" can't be added to the end

CVE-2004-2597 - long buffer from client causes server additions to be truncated, preventing key/value pair from being modified and leading to ACL bypass and spoofing.

Others:

CAN-2005-0983 - long message is not properly truncated, causing the remainder of the message to be interpreted as different data.

****

MISC.SIGNAL. Signal Errors

Definition: the product does not properly handle or manage a signal.

Note: several sub-categories could exist, but this needs more study. Some sub-categories are:

[*] unhandled signals
[*] untrusted signals
[*] sending wrong signals

Note: Signal Handler Race Conditions are covered elsewhere.

Examples:

CAN-2002-2039 - unhandled SIGSERV signal allows core dump

CAN-1999-1224 - SIGABRT (abort) signal not properly handled, causing core dump

CAN-2002-2039 - SIGSERV (invalid memory reference) signal causes core dump

CAN-2004-1014 - remote attackers cause a crash using early connection termination, which generates SIGPIPE signal

CAN-2005-2377 - library does not handle a SIGPIPE signal when a server becomes available during a search query. Overlaps unchecked error condition?

CAN-2002-0839 - SIGUSR1 can be sent as root from non-root process

CAN-1999-1441 - kernel does not prevent users from sending SIGIO signal, which causes crash in applications that do not handle it. Overlaps privileges.

CVE-2000-0747 - script sends wrong signal to a process and kills it.

CVE-1999-1326 - interruption of operation causes signal to be handled incorrectly, leading to crash

CVE-2001-1180 - shared signal handlers not cleared when executing a process. Overlaps initialization error.

CAN-2004-2069 - privileged process does not properly signal unprivileged process after session termination, leading to connection consumption

CAN-2004-2259 - SIGCHLD signal to FTP server can cause crash under heavy load while executing non-re-entrant functions like malloc/free. Possibly signal handler race condition?

CAN-2005-0893 - certain signals implemented with unsafe library calls

****

MISC.STDCHK. Improperly Implemented Security Check for Standard

Definition: the software does not properly implement one or more security-relevant checks as specified by the design of a standardized algorithm, protocol, or technique.

Note: this is a "missing step" error on the product side, which can overlap WIFFs such as insufficient verification and spoofing. It is frequently found in cryptographic and authentication errors. It is sometimes resultant.

Note: this is an implementation error, in which the algorithm/technique requires certain security-related behaviors or conditions that are not implemented or checked properly, thus causing a vulnerability.

CAN-2002-0862, CVE-2002-0970, CVE-2002-1407 - browser does not verify Basic Constraints of a certificate, even though it is required, allowing spoofing of trusted certificates.

CAN-2005-0198 - logic error prevents some required conditions from being enforced during Challenge-Response Authentication Mechanism with MD5 (CRAM-MD5)

CAN-2004-2163 - shared secret not verified in a RADIUS response packet, allowing authentication bypass by spoofing server replies.

CAN-2005-2181, CAN-2005-2182 - insufficient verification in VoIP implementation, in violation of standard, allows spoofed messages.

CAN-2005-2298 - security check not applied to all components, allowing bypass

****

MISC.MISINT. Misinterpretation Error

Definition: the product misinterprets an input, whether from an attacker or another product, in a security-relevant fashion.

Note: this concept needs further study. It is likely a factor in several WIFFs, possibly resultant as well. Overlaps MIE.

CAN-2005-2225 - product sees dangerous file extension in free text of a group discussion, disconnects all users

CVE-2001-0003 - product does not correctly import and process security settings from another product

****

MISC.BUSRULE. Business Rule Violations or Logic Errors

Definition: the product performs as expected with respect to documented WIFFs, manipulations, and attack vectors, but it behaves in certain ways that can only be regarded as vulnerabilities within the context of the "business rules" that the product implements.

Note: it is hoped that most business rule violations can already be captured by other PLOVER categories, e.g. "users should not have privilege X" or "customers should not be able to modify prices."

Research Gaps: under-studied as a concept. Since business rule violations are most likely to appear in custom software, third party code auditors may have insights regarding this type of problem.

****

MISC.SAMENAME. Same Name Error

Definition: the product relies on the name or identifier of a resource for security-relevant decisions, but it does not properly protect against the use of other resources that have the same name.

Alternate term: Same Identifier Error

Terminology Note: "name" is a slight misnomer in that a "name" does not have to be an alphabetic identifier or word.

Note: overlaps untrusted path, macro/function redefinition, spoof, bypass, origin validation error, insufficient verification, others.

Note: similar to equivalence errors.

Examples:

CAN-2005-1933 - attacker can override system behavior using a resource with the same name

CAN-2004-1051 - product allows the attacker to define functions that are executed in place of programs with the same function name

CVE-2002-2063 - malicious program can bypass firewall if it has the same filename as an otherwise "trusted" file (though this is also an "insufficient verification" problem)

CAN-2004-0708 - attacker gains privileges by creating username that has the same name as a privileged group.

CAN-2005-1791 - near-equivalence causes crash (domain name looks like an IP address)

CAN-2002-0572 - Unix-based OS allows local users to access restricted files by closing the file descriptors for stdin, stdout, and stderr, which might then be reused by a called setuid process.

CAN-2002-2053 - protocol implementation allows DoS by router with same IP address as the product router.

****

MISC.ALTNAME. Other Alternate Name Error

Path traversal vulnerabilities, which have already been covered, involve the manipulation of the "alternate name" property to reach the same resource. Alternate name problems exist elsewhere, although they are not documented as heavily.

Examples:

CAN-2004-2083 - browser allows misrepresentation of "safe" file type in download box using CLSID in filename

CAN-2004-0420 - application allows attackers to spoof the type of a file with a CLSID specifier in the filename.

****

MISC.NEIGHBORNAME. Neighbor Name Error

Definition: the name of a resource ("neighbor name") can be inferred or guessed from a known name. The most visible examples are in filenames.

Note: related to predictability.

Examples:

file.ext~ (tilde) - commonly used for backups

file.ext.bak - backup

CAN-2002-1928 - view directory using "~" or ".bak"

****

MISC.UNDOC. Undocumented Functionality

  Definition: the product contains functionality that is not documented but is not otherwise malicious in nature.

  Note: Undocumented functionality is conceptually similar to product-embedded malicious code. The fundamental difference is that undocumented functionality is not necessarily malicious in nature.

****

MISC.UNDOC.ACCOUNT. Undocumented Account

  Examples:

  CVE-2006-0181 - undocumented admin account has default password, allowing privilege escalation

****

MISC.UNDOC.EGG. Easter Egg

  Definition: the product contains "easter egg" functionality that does not compromise security or privacy of the user, but is unrelated to the proper functioning of the product, e.g. an embedded game or animated credit list of the developers.

  Note: always primary. Easter egg itself may contain other WIFFs.

****

MISC.CONF. Configuration Error

  Configuration errors are under-studied from a vulnerability classification perspective. This category identifies common configuration problems that are not covered in other sections.

****

MISC.CONF.DEFPASS. Default Password

  Definition: the product is installed with a password that is common across many installations, but the product does not require that the password be changed before access is granted.

Note: typically primary, but sometimes resultant from unexpected installation actions or interruptions.

Examples:

CVE-2005-3717 - WIFI phone has default password

CVE-2005-3595, CVE-2005-3344, CVE-2005-0865 - admin account installed with a blank password

CVE-2005-3280 - product installs database with default password for database admin account

CVE-2005-0601 - default password used if a setup dialog has not been run. Resultant.

CVE-2004-1591 - router restes password to default if the router is shut off. Resultant.

****

MISC.CONF.ACCESS. Broad Access Configfuration

Definition: the product's default configuration allows access to a broad set of users, systems, etc., instead of a restricted set.

Examples:

CAN-2002-1782 - default configuration allows a user without shell access to read files as that user

CAN-2002-1921 - default configuration of database, when running on Windows, does not set the bind address to loopback, allowing remote connections

CAN-2005-1748 - remote anonymous connections allowed, leading to infoleak or DoS

****

MISC.CONF.AUDIT. Insufficient Audit Configuration

Definition: the product's auditing or logging features are not properly configured to capture the information that is requied for the product's operating environment.

Note: this is frequently dependent on the particular enterprise.

CAN-2002-1923 - logging not enabled by default

==========
SECTION.10. Additional Examples
==========

Many additional examples are provided in this section. They can serve different functions.

1) The manipulation, alternate elements, and consequence examples further demonstrate these important concepts.

2) The categorized examples highlight additional subtleties or unusual manifestations of the associated WIFFs.

3) The uncategorized examples include outliers that are not currently describable by PLOVER, complex examples, or other examples that simply have not been inserted into PLOVER yet.

==========
SECTION.10.1. [ALT] Alternate Elements Examples
==========

These are additional examples for demonstrating the concepts of alternate names, channels, and paths.

Note that many examples are already scattered throughout the WIFF list and other sections with examples.

****

EX.ALT.NAME. Alternate Name Examples

****

CVE-2001-0846 - obtain database by requesting it using its ID instead of its name.

CVE-2001-0873 - inconsistency between "-option" and "--option" (long option name)
- alternate name

- there must be some more examples out there...

CVE-2001-0664, CVE-2001-0724 - browser allows security restriction bypass using URLs with dotless IP addresses.

CVE-2001-0664, CVE-2001-0724 - dotless IP address

CVE-2002-1961, BID:7456 - trailing dot in a fully qualified domain name (FQDN), e.g. "www.example.com."

CAN-2003-0896 - attacker provides a Java class name that uses an internal representation instead of the expected one (example of "alternate name")

CAN-2001-1026, CAN-2002-1877, CAN-2002-1962 - filter bypass using IP address instead of hostname in a URL

CAN-2002-1790 - filter bypass using encapsulated SMTP addresses

CAN-2003-0976 - component does not support aliases, which prevents restrictions from being applied

****

EX.ALT.CHAN. Alternate Channel / Alternate Path Examples

CAN-2002-1883 - product opens unprotected alternate port

CAN-2005-2150 - alternate named pipes accessible by null sessions

CAN-2005-2261 - incomplete disabling of scripts allows execution via alternate channel. Probably MFV.

CAN-2005-2144 - permission bypass using alternate channel - using memory mapping to access files

CAN-2005-1970 - bypass using feature (alternate channel)

CAN-2004-2176 - firewall trusts an application that can be used as a proxy for other processes; result is bypass via alternate channel

CAN-2002-2083 - authentication bypass via alternate path (help feature launched from login window)

- CAN-2002-1722 - physical access screen lock bypass by pressing

user-assigned buttons (alternate path)

- CAN-2001-1520 - alternate channel - user with physical access can obtain PIN using a serial monitor. Also overlaps "sends sensitive information to entity over untrusted channel" - the read-only version of "allows external input for critical internal variables"

- CAN-2005-2148 - insufficient filtering allows SQL injection via alternate channel. An MFV.

- CAN-2002-1826 - permission bypass using alternate channel (using mmap to access memory devices)

SECTION.10.2. [MAN] Manipulations Examples

Note: This list of manipulations is incomplete. The most basic manipulations are frequently covered elsewhere, e.g. "inject special characters" or "provide long input."

****

MANIP.LENGTH. Lengthening manipulation

This manipulation involves providing more data than is expected, making it "longer" or "larger." Many buffer overflow attacks (but not all) require an extender manipulation to provide a long or large argument.

CVE-2002-0462 - product with very large parameter either causes external infoleak in one configuration, or resource consumption in another

****

MANIP.SHORTEN. Shortening manipulation

This manipulation involves providing less data than is expected, making it "shorter" or "smaller."

****

MANIP.COMPRESSOR. Compressor manipulations

Definition: A compressor manipulation provides an input that, when transformed by the product, produces an output that is larger than the original input.

For example, the attacker could provide a string "&&" which could be expanded to "&&" in an HTML context, possibly leading to a buffer overflow.

Or, the attacker could create a very small ZIP file that, when unzipped, expands to an extremely large result.

CVE-2002-0068 - FTP URL with many special characters causes a core dump when client escapes the characters.

CAN-2001-0247 - buffer overflow using wildcard characters to expand string

****

MANIP.INFLATOR. Inflator manipulations

Definition: An inflator manipulation is the opposite of a compressor. The attacker provides an input that, when transformed by the product, produces an output that is smaller than the original input. Such manipulations theoretically exist, but no public reports are known.

One effect of an inflator manipulation might be to expose unused portions of a buffer that were expected to be filled by the product.

****

MANIP.SPOOFING. Spoofing manipulations

Note: The spoofing concept is covered elsewhere, but "inserting a false identifier," or a false reference, is an important manipulation in many vulns.

****

MANIP.INCONSISTENCY. Multiple Value Inconsistency

Definition: the attacker manipulates multiple values so that they are inconsistent.

Note: this manipulation is frequently successful in exploiting multiple interpretation errors. However, it is also a factor in other WIFFs such as buffer overflows, e.g. when using a length parameter manipulation so that the length field for a buffer does not reflect the actual length of the buffer.

Functional Area: non-specific

Examples: CAN-2004-0244 (difficult to search for examples of this type, although they are known to exist)

****

MANIP.REFLOOP. Reference loop

Definition: A "reference loop" exists when Object A refers to B, which refers back to A. Reference loops can include more than two objects, such as A->B->C->A.

If a reference loop violates a consistency property, then it could have security-relevant consequences, typically an infinite loop, amplification, or invalid pointer dereferencing.

CAN-2005-1829 - infinite loop/crash via 2 embedded objects that call each other ("reference loop")

****

MANIP.DOUBLE. Double or duplicate elements.

****

MANIP.FILEXT.MULT. Multiple File Extensions

Definition: the attacker uses a filename with multiple extensions.

Note: this manipulation can be useful in multiple interpretation errors, containment errors, remote code injection, invalid handler deployment, and others. It is frequently found in PHP applications, e.g. "test.php.jpg".

It also overlaps misrepresentation errors in the user interface, e.g. spoofed icons or truncated long filenames with dangerous extensions.

Functional Area: file processing

Examples: CAN-2004-1404, CAN-2004-1405, CAN-2002-0223, CAN-2005-0565

****

MANIP.MIXTYPE. Mixed Data Types

Definition: use a data type that is not appropriate for the associated input.

This manipulation can be used in a couple different cases. For example, the attacker could provide an alphabetic argument for a numeric field, in order to manipulate the validity property. Or, the attacker could inject HTML/script into text files that are automatically processed as if they were HTML, or use URL encoding when communicating with an FTP server that tries to be "friendly" to web clients that don't remove URL encoding before making FTP requests.

****

MANIP.DATA.UNC. Uncontrolled Data Manipulations

Uncontrolled manipulations are manipulations that are not performed with repeatable or well-designed data. The resulting data is likely to be interpreted as a more specific manipulation by the product, but that manipulation is not known at the time of generation.

From the product's perspective, this data is going to be either well-formed or malformed, valid or invalid, consistent or inconsistent. However, the attacker does not necessarily know which of these properties is held by a particular manipulation.

This is a legitimate and common manipulation, although it can make diagnosis more difficult.

****

MANIP.DATA.UNC.RAND. Random Data

The attacker uses randomly generated data in an attack.

****

MANIP.DATA.UNC.CONT. Well-formed Data in Wrong Context

Definition: The attacker uses data that is well-formed for one context, but not in the format or structure that is expected by the product. For example, the attacker could provide a JPEG image to an audio tool that expects an MP3 sound file. Data could also be generated on the attacker side using rules that do not follow the product's expected structure or format. For example, a rule might be "send all possible sub-strings between 1 and 3 characters long."

Note: overlaps Mixed Data Types.

****

MANIP.STEP. Step Manipulations

****

MANIP.STEP.MISSING. Missing step

MANIP.STEP.MISSING.FIRST. Missing first step

MANIP.STEP.MISSING.LAST. Missing last step

Examples:

CAN-2000-1227 - resource consumption by sending requests but not reading the responses

CAN-2005-1911 - product hangs while waiting for input that never arrives

CAN-2004-0829 - attacker performs step without previous required step

****

MANIP.STEP.ORDER. Out-of-order step

CAN-2002-2082 - step ordering error - resource is locked before authentication succeeds, allowing attackers to lock other resources

CVE-2000-1022 - step-based manipulation on out-of-order operations

CAN-2001-1560 - step order violation? leads to crash

CVE-2005-3296 - FTP server directory listing by using LIST before logging in

****

MANIP.STEP.REPEAT. Repeated step

   Definition: perform the same step multiple times.

   Note: depending on the associated WIFF, repeated step manipulations can trigger buffer boundary violations, not just resource consumption.

   CAN-2002-1763 - step-based fault (repeatedly pressing certain keys) causes screensaver crash and resultant authentication bypass

****

MANIP.STEP.REPEAT.FLOOD. Flooding Step

   Definition: perform a repeated step rapidly.

   Note: this is used to exploit resource exhaustion problems, including asymmetric consumption

    CAN-2002-1876 - resource exhaustion (licenses) via large number of rapid requests

   CAN-1999-1569 - flood of spoofed UDP packets exceeds server's user limit

   CAN-2002-1850 - hang/memory consumption by writing a large amount of data to stderr

****

MANIP.STEP.INTERRUPT.   Interrupted step (early termination)

   Definition: the attacker terminates session, procedure, algorithm, etc. before normally expected.

   Note: the results of this manipulation can be variable, but it can result in infinite loop, null dereference, memory leaks, and others.

   Note: this overlaps incomplete data manipulations.

   CAN-2002-1862 - step-based vuln by closing connection before all

data has been sent

CAN-2005-2170 - DoS by connecting then disconnecting without sending any data

CVE-2004-2356 - null deref by connecting then disconnecting without sending any data

CAN-2004-0437 - disconnect from FTP server while performing "LIST -L," which causes an invalid socket to be accessed.

CAN-2002-1942 - certain Keep-Alive connections are not properly handled if terminated early

****

MANIP.STEP.INCOMPLETE. Incomplete step

Definition: perform only a portion of a particular step.

Note: overlaps data manipulations.

CAN-2002-1906 - CPU consumption via incomplete HTTP requests and leaving those connections open.

****

MANIP.STEP.DELAY. Delayed Execution of Next Expected Step

Definition: the attacker delays, or does not perform, the next expected step.

Note: overlaps missing step

CAN-2003-0744 - product hangs while waiting for expected input. step-based manipulation - "stop executing steps"

CVE-2001-0513 - product opens up a separate port and redirects user to that port, allowing port consumption when user does not connect to the separate port.

****

MANIP.STEP.EX. Miscellaneous step-based examples

CAN-2004-0829 - crash by sending a "find next" request without the

required initial "find first" request

CAN-2000-0647 - FTP server crash with MLST before user logs in

CAN-2000-0648 - FTP server crash with a "RENAME TO" command before a "RENAME FROM" command.

(unexpected abort causes infinite loop)

****

MANIP.DATA. Data manipulations

****

MANIP.VALID.ZEROLEN. Zero Length Issues

Examples: CAN-2004-0218, CAN-2004-0367, CAN-2004-0627 (overlaps authentication), CVE-1999-0905, CVE-2001-0825 (overlaps overflow), BID:4804

****

MANIP.NESTING. nesting manipulations

CAN-2001-0519 - nested SCRIPT tags bypass script filter

- CAN-2005-2161
  - by nesting [url] tags, XSS is possible
- same with CAN-2005-2327

CAN-2005-2161 - nested structure manipulation allows XSS

CAN-2005-1935 - manipulation involving nested structures

CAN-2005-2327 - nested structure enables XSS

CAN-2005-1665 - deep nesting causes CPU consumption

****

MANIP.GEN.SPOOF. Spoofing

CAN-2001-1519 - create a spoofed named pipe

CAN-2002-2063 - bypass by spoofing trusted filenames

CAN-2005-1942 - bypass security using spoofed messages

CAN-2005-2268, CAN-2005-2271, CAN-2005-2272, CAN-2005-2273, CAN-2005-2274 - GUI does not clearly identify the origin of a dialog box; overlaps spoofing

==========

MANIP.EX. Manipulation examples

****

MANIP.EX.ALT

CAN-2005-1994 - bypass access restrictions (imposed by intermediary) using hex-encoded characters such as "%2e". Overlaps multiple interpretation error and alternate name.

****

MANIP.EX.LONG - long input manipulation

CAN-2004-2165 - long input manipulation leads to unallocated memory write

CVE-2002-2081: long input causes service abort, no big deal normally except this happens during mid-transfer, so the file being uploaded is never cleared. result: disk consumption.

CAN-2005-2105 - long username manipulation leads to authentication bypass

CAN-2002-2081 - disk consumption (resultant) via long input manipulation, which causes crash without deleting file being uploaded

****

MANIP.EX.NUMERIC

CAN-2004-2179 - manipulation using maximum allowable numeric values

****

MANIP.EX.NUMERIC.SIZE. manipulation of size or length field to introduce inconsistency

CAN-2002-1828 - negative length value causes crash

CAN-2002-1768 - random manipulation (packet size) causes DoS

CAN-2004-2223 - crash caused by large size manipulation

****

MANIP.EX.VALIDITY

CAN-2002-1969 - DoS (crash) via invalid username

CAN-2002-1801 - error message infoleak via category that does not exist

CAN-2000-1226 - unsupported protocol (non-IP) packets cause crash

CAN-2005-2265 - crash by invalid argument (wrong type - object instead of string)

CAN-2005-1885 - error message infoleak via invalid (non-integer) value

****

MANIP.EX.UNSTRUCT - non-random, unstructured manipulation

CAN-2002-1881 - non-random, unstructured manipulation of content (ROT13 encoding) causes DoS

CVE-2001-0080 - DoS by connecting to SSH using non-SSH client causes "protocol mismatch error"

****

MANIP.EX.MULT. misc. multi-manipulations

CAN-2005-2239 - multiple manipulation - long string + special (null) characters er with a large number of / characters

****

MANIP.EX.MISC misc/unclassified manipulations

CAN-2004-2147 - DoS via email message without a body

CAN-2001-1489, CAN-2001-1490, CAN-2001-1491 - CPU consumption and memory leak (?) via web page with large number of images

CAN-2005-2006 - unusual manipulation produces resultant infoleak

CAN-2002-1994 - manipulation of multiple HTTP requests with a single CRLF instead of the normal 2

CAN-2002-2003 - DoS by structured manipulations using nmap, actual fault/manipulation undiagnosed

CAN-2001-1552 - multiple newlines cause DoS

CAN-2005-1931 - crash from malformed (or invalid?) argument

CAN-2005-1793 - large width and height value manipulations cause a crash

CAN-2005-1808 - large size value causes memory allocation failure and triggers exception

CAN-2005-1643 - large size value leads to failed memory allocation or out-of-bounds read

****

MANIP.EX.FRAG. Fragmentation

This manipulation involves breaking a data element into multiple smaller fragments, which are later combined by the product into the original element.

Fragmentation manipulations can lead to consequences such as denial of service, bypass, and buffer boundary violations.

Note: This concept involves any sort of element that can be subdivided, not just packets.

Types of fragmentation:

MANIP.EX.FRAG.COMPLETE. Complete fragmentation

All fragments, when combined, produce an entire data element.

CAN-2001-1572 - filter bypass using small packets

MANIP.EX.FRAG.INCOMPLETE. Incomplete fragmentation

All fragments, when combined, leave "gaps" in the data.

MANIP.EX.FRAG.OVERLAP. Overlapping fragmentation

Some fragments, when combined, can overlap and overwrite other fragments.

Examples:

CAN-2004-0744 - DoS using "Rose Attack" by sending small packet fragments that don't produce a full packet

CAN-2001-1540 - DoS by fragmented IP packets that split the TCP header

BID:6245 - bypass URL blocking by fragmenting the URL into separate packets

****

MANIP.EX.NONSTD. Inject Non-standard Data.

- CAN-2002-1775 - non-RFC MIME header
- CAN-2002-1778 - firewall bypassing using certain invalid TCP flag combinations e.g. SYN/FIN.
- CAN-2002-2072 - DoS via invalid null argument
- CAN-2002-2075 - large number manipulation causes DoS; fault unknown

****

MANIP.IMMUTABLE. Modify Assumed-Immutable Data.

SECTION.10.3. [ACON] Atomic Consequences - Examples

Most atomic consequences have already been covered in other sections. However, some atomic consequences are frequently reported as if they were problems on their own, so they are highlighted here.

****

EX.ACON. Invalid Pointer Dereference

  Note: this overlaps null dereferences, but is intended to be
  slightly different.

  CAN-2004-1748, CAN-2004-1718, CAN-2005-0114, CAN-2004-0767,
  CAN-2004-0766 - invalid pointer to hook function leads to crash.

    CAN-2005-1830 - invalid pointer causes crash

****

EX.ACON.NULLDEREF. Null Dereference (Null Pointer Dereference)

   Functional Area: non-specific

   Note: resultant from various issues, including unchecked error
   condition and race condition.

   Note: most vulnerability reports only list the null dereference and
   not the underlying trigger.

   Examples: CAN-2004-0079, CAN-2004-0365, CAN-2003-1013,
             CAN-2003-1000, CAN-2004-0389 (overlaps malformed inputs),
             CAN-2004-0119, CAN-2004-0458 (overlaps missing argument),
             CVE-2002-0401

   CVE-2005-3274 - race condition causes a table to be corrupted if a
   timer activates while it is being modified, leading to resultant
   null dereference; also involves locking.

   CAN-2002-1912 - large number of packets leads to null dereference

   CAN-2005-0772 - packet with invalid error status value triggers
   null dereference

****

EX.ACON.DIVZERO. Divide-by-zero

  Note: this can be resultant from various issues, including unchecked
  error condition. Examples are difficult to find, although it is
  likely that many issues have been reported at higher levels,
  e.g. "crash".

CAN-2005-2134 - multiple operations at the same time cause divide-by-zero, fault unknown

CAN-2005-0306, CAN-2004-0804 - numeric parameter of "0" cause divide-by-zero

CAN-2004-0245 - large or negative Content-Length causes divide-by-zero

CAN-1999-1448 - dates before a minimum, or well after the current, lead to divide by zero

CAN-2004-0804 - image processor generates divide-by-zero when the number of row bytes is zero.

****

EX.ACON.ACCFREED. Access of Previously Freed Memory

Note: this is the result of other faults

- CAN-2004-1141, CAN-2004-1093, CVE-2004-0080 (overlaps infoleak), CAN-2001-1397, CAN-2003-0813 (overlaps race), maybe CAN-2004-1057

****

EX.ACON.DOUBLEFREE. Double-free

Note: this is a special instance of Access of Previously Freed Memory.

Functional Area: memory management

Examples: CVE-2002-0004, CVE-2000-0550, CVE-2002-0847, CVE-2002-0059, CAN-2004-0416, CVE-2003-0015, CVE-2003-0073

****

EX.ACON.ACCUNINIT. Access of Uninitialized Memory

Note: resultant from an initialization error.

Note: other flavors probably exist

CAN-2004-0082 - copy of uninitialized buffer into a password field might make a less secure password

****

EX.ACON.INFLOOP. Infinite loop

  Functional Area: control flow, non-specific

  Note: this is more the result of a programming error. Multiple
  sub-categories likely. More study is needed.

  Factors: can be primary to amplification or flooding, can be
  resultant from integer handling errors and probably many others.

    Examples: CVE-2000-0620, CVE-2000-1203, CVE-2000-0738,
              CAN-2002-1355 (overlaps integer signedness),
              CVE-2002-0403

    CAN-2005-2295 - infinite loop via zero size value

    CVE-2001-0194 - infinite loop due to long input manipulation

    CAN-2005-1899 - infinite loop via zero-length data (packet)

    CAN-2005-1923 - infinite loop caused by maximum alowable value
    manipulation leading to zero-length read

    CAN-2005-1807 - long header field leads to infinite loop (memory
    and CPU consumption)

    CAN-2005-1739 - 0 value leads to infinite loop

    CAN-2005-1741 - infinite loop by malformed data

****

EX.ACON.LONGLOOP. Long Loop

  Definition: the product enters a loop that is not infinite, but
  performs many more iterations than intended.

  Note: often resultant from integer signedness errors, e.g. a loop
  from 1 to -1.

****

EX.ACON.DEADLOCK. Deadlock

Note: this is under-studied relative to vulnerability research.

Examples: CAN-2001-1400

****

EX.ACON.INFREC. Infinite Recursion

Definition: the product enters a series of recursive calls that do not have any terminating condition that could be met.

Note: this is a special type of infinite loop.

CAN-2001-1539 - DoS by causing stack recursion

CAN-2002-1714 - infinite recursion via an object with a field that references the document that contains the object

CAN-2002-1902 - infinite recursion by creating a child of an outdated parent

****

EX.ACON.DEEPREC. Deep Recursion

Definition: the product enters a series of recursive calls that are not infinite, but go more deeply than intended.

****

EX.ACON.STALE. Stale Identifier, Pointer, or Handle Access

Definition: the product accesses an identifier, pointer, or handle that is "stale," i.e. the associated object has been moved or delted.

Note: sometimes a consequence of a race condition, resource management error, or unhandled error condition.

Note: bugs that lead to stale handles can overlap with infoleaks, e.g. when data is read from previously freed memory.

CAN-2004-0689 - doesn't handle when symlinks point to stale locations

BID:6305 - stale process ID for a privileged process may allow a

later process with the same PID to access network traffic.

CAN-2002-1674 attacker causes DoS by removing a file that another function is referencing (stale identifier bug). overlaps race condition.

SECTION.10.4. [CAT] Additional Categorized Examples

This section contains a large number of additional examples for previously described categories.

****

EX.MFV. Examples - Multi-Factor Vulnerabilities

While many items in PLOVER are multi-factor, these examples help to illustrate the variety of MFVs that have been reported. For most taxonomies, these could be outliers or classifiable under more than one class.

CAN-2003-0981 - product records the reverse DNS name of a visitor in the logs, allowing spoofing and resultant XSS.

CVE-2004-2351 - resultant XSS from incomplete blacklist (only <script> and <style> are checked)

CAN-2005-2819 - XSS using "Conditional Comments" in Internet Explorer (overlaps multiple interpretation error, incomplete blacklist)

CVE-2005-4454 - MFV. Filter-before-canonicalize, incomplete blacklist, XSS, interaction error. Product searches for "javascript" in style attributes before stripping "\", allowing a "javas\cript" to yield a valid pseudo-URI that is rendered by some web browsers.

CVE-2004-2398 - MFV. Interaction error, permissions, predictability. One product installs a directory with world-readable permissions, another product uses that directory and uses filenames that contain valid usernames, leading to infoleak.

CAN-2003-0721 - Integer signedness error causes an out-of-bounds

array access using a negative number.

CVE-2004-2352 - XSS in PHP script via an alternate path using cookies instead of POST data

CVE-2001-0054 - MFV. directory traversal and other issues in FTP server using Web encodings such as "%20"; certain manipulations have unusual side effects

CVE-2002-1982 - MFV. directory traversal sequences and a discrepancy information leak lead to disclosure of the existence of files.

CVE-2004-1354 - MFV. directory traversal sequences and information leak by inconsistent responses lead to disclosure of the existence of files.

CAN-2005-2319 - PHP file include under complex, atypical conditions, bypassing local file existence check

CVE-2005-3288 - "lazy" race condition combined with direct request. User can upload file with dangerous extension while composing a message, then access that file before the message is completed and the product renames the file to a safer extension.

CAN-2003-0161 - MFV. email address parser does not properly handle certain conversions from char and int types, causing a length check to be disabled when an input value is interpreted as a special control value, leading to resultant buffer overflow.

CAN-2002-0253 - obtain physical path via trailing slash, which modifies a base path and causes an include directive to fail, leading to error message infoleak

CAN-2005-0708 - memory disclosure that occurs when a file is truncated while it's being transferred. Involves a step-based attack and a non-standard race condition.

CAN-2001-1534 - predictable session ID's allows authentication bypass (primary insufficient randomness with resultant authentication)

[long input manipulation, early termination step manipulation]
CVE-2002-0741 - DoS by sending command with long argument, then immediately terminating connection

CAN-2002-2057, CAN-2002-2058 - MFV. weak encryption and sensitive

file under web root. A common vulnerability.

CVE-2003-0124 - MFV. malformed file with improper quotes causes a static string to be returned, which is then used to find a program to execute. Variant of untrusted search path.

CAN-2002-2025 - MFV. resource exhaustion via flood of requests using MS-DOS device names

CAN-2005-1768 - race condition in concurrent threads with resultant overflow

CVE-2004-2354 - XSS manipulation triggers SQL injection problem, which is reflected to user when MySQL generates errors

CAN-2002-1676 - cleartext passwords in config file while product is running. "lazy" race condition.

CAN-2003-0972 - Integer signedness error via a large number of special characters in escape sequences, leading to resultant buffer overflow.

SECUNIA:18223 - argument injection, incomplete blacklist, interaction error. Program filters dangerous "-S" arguments but does not filter getopt-style "-vS" arguments.

CVE-2002-0121 - session IDs stored in temporary files whose name contains the session ID, allowing local users to hijack web connections. Overlaps containment error.

****

EX.CAT.BUFF.OVER. Examples - Unbounded Transfer ("classic overflow")

CVE-1999-0006 - buffer overflow using long password

CVE-1999-0021 - buffer overflow in CGI program using long query string, referer, or user agent string.

CVE-1999-0879 - buffer overflow using macro variables

CAN-2005-2120 - large number of consecutive special characters (("\" in registry key name) leads to overflow

CVE-1999-0368 - buffer overflow in FTP server by creating large directory names

CAN-2001-0247 - buffer overflow using wildcard characters to expand string

CVE-2001-0236, CVE-2001-0500 - buffer overflow via long argument

CVE-2001-0836 - buffer overflow via long URL

CVE-2002-0801 - buffer overflow via long HTTP Host header field

CAN-2002-0031 - multiple buffer overflows using long arguments in a URI

CAN-2002-0154 - multiple buffer overflows in long arguments to database extended stored procedures

CAN-2003-0533 - buffer overflow that causes long debugging entries to be created

CAN-2004-0460 - buffer overflow in logging utility using multiple options

CAN-2002-1692, CAN-2005-1826 - buffer overflow via long file extension

CAN-2002-1754 - buffer overflow by DNS resolution to long hostname

****

EX.CAT.BUFF.INDEX. Array index overflow

CAN-2003-0072 - request causes out-of-bounds read

CVE-2004-0093 - out-of-bounds array index in window manager

CVE-2001-1036 - out-of-range offset

CAN-2002-1066 - large message index value in POP RETR/DELE command

CAN-2005-2115 - large ID value used as array index

CAN-1999-0798 - numeric "type" argument used as an array index

CAN-2002-1387 - "number of operations" argument used as index

****

EX.CAT.BUFF.LEN. Length parameter manipulation ("length tampering")

****

EX.CAT.BUFF.MISC. More boundary violation examples

  CAN-2002-1687 - buffer overflow via environment variable

  CAN-2002-1792 - buffer overflow involves manipulation of long input
  by splitting into multiple packets

  CAN-2002-1973 - buffer overflow triggered by a query string that
  causes a parsing error

    CAN-2005-1766 - heap overflow without multi-field manipulations

  CAN-2005-1873 - buffer overflow involving wildcard

  CAN-2005-2081 - buffer overflow in parser using special characters

  CAN-2005-2213 - buffer overflow via large number of entities

  CAN-2004-0444 - multiple interesting examples - length tampering,
  overflow during expansion/transformation, and an overflow that
  occurs by NOT providing certain expected fields.

  CAN-2004-0891 - unusual circumstances cause an unbounded write to
  the wrong buffer

   CAN-2003-0057 - buffer overflow by connecting from IP address that
   DNS resolves to a long hostname

  CVE-2005-1268 - off-by-one error leads to overwrite of one null byte

  CAN-2005-1770 - buffer overflow requires certain signals to trigger

****

EX.CAT.FORMSTR. Format string vulnerability

   CAN-2000-0574 - format strings used when creating title for a
   process

   CAN-2004-0354 - format strings in logging and error functions

CVE-2000-0594 - format string in IRC client via channel name

CVE-2000-0763, CVE-2001-0111 - local format string via command line argument

CVE-2000-0844 - format string in internationalization product

CVE-2000-0967 - format string by triggering errors

CVE-2000-1000 - format string in filename

CVE-2000-1004 - format string in directory name

CVE-2000-0573 - format string in SITE EXEC command in FTP server

CVE-2004-0159, CVE-2004-0159 - format strings in file names not handled by "ls" command

CVE-2001-0060, CAN-2001-0609 - format string in malicious IDENT server response

CVE-2001-0740 - lagre number of "%s" *might* trigger format string (undiagnosed)

CVE-2001-1081 - format strings into log messages

CVE-2002-0374 - format string in ICQ client

CVE-2002-0251 - buffer overflow via large number of format strings - possibly involving expansion?

CAN-2001-0281, CAN-2003-0697, CAN-2004-0733 - format string in debugging commands

CAN-2004-0800 - format string in name of invoking program

CAN-2005-2390 - product-specific format string specifiers such as "%C", "%R", and "%U" lead to information leak via a shutdown message

CVE-2002-0716, CAN-2004-0536 - local format string via filename

CVE-2002-0916, CVE-2002-1244, CVE-2002-1519, CAN-2002-0930, CAN-2003-0391, CAN-2004-2074 - format strings in username or password

CAN-1999-1417 - MFV. format strings using encoded "%" characters

CAN-2005-1122 - MFV. format strings using double-encoded "%" characters.

CAN-2000-0918, CAN-2000-1207 - format strings in environment variable

CAN-2001-1078 - format strings in arguments to common SMTP and POP3 commands

CAN-2002-0586, CAN-2002-0587 - same vector has both format string and overflow

CAN-2002-0702, CAN-2002-0913 - format strings in DNS server response

CAN-2005-1738 - format string allows access to files outside restricted directory

CVE-2002-0598 - format string in security product via server banner

****

EX.CAT.ELT.MISS. Missing parameter/field/argument

CVE-2002-1169, CVE-2002-1169 - missing version number in HTTP request triggers crash

CVE-2000-0521 - missing HTTP version number triggers source code disclosure

CVE-2001-0590 - missing HTTP protocol specification triggers source code disclosure

CAN-2002-1023 - crash in HTTP request without the URI

CAN-2002-1488 - crash in IRC client via PART message without channel name

CAN-2003-0239 - malformed GIF does not have a color table after an image descriptor

CAN-2002-0566 crash via an HTTP Authorization header without an authentication type.

CAN-2005-2399 - trigger SQL errors in web app with missing parameters

****

EX.CAT.SPEC.DELIM.LINE. Delimiter between lines

CVE-2001-0902 - MFV. spoof web log entries with URL encoded carriage returns/line feeds

CVE-2002-1405 - CRLF injection in web browser allows injection of false HTTP headers

CAN-2002-1575 - spam proxy via URL-encoded newlines in mail-related parameters such as subject line

CAN-2003-0336 - MFV. carriage return in spoofed special string allows arbitrary file read

CAN-2004-2140 - bulletin board allows modification of text file via CRLF in subject

CVE-2000-0610 - bypass mail server authentication and spam proxy via username with carriage return

CAN-2004-2146 - HTTP response splitting in bulletin board using CRLF in CGI script parameter

CVE-2001-0902 - Microsoft IIS 5.0 allows remote attackers to spoof web log entries via an HTTP request that includes hex-encoded newline or form-feed characters.

****

EX.CAT.SPEC.WILDCARD. Wildcard, matching, or "completion" character

CVE-2000-0587 - bypass directory permissions using filename completion

CAN-2001-1501 - CPU/memory consumption using many wildcards

CAN-2004-0930, CAN-2005-0256 - CPU consumption using wildcards

CAN-2002-0558 - MFV. directory traversal using ".." and wildcards

CAN-2003-1137 - read files or execute scripts using wildcard

character

CAN-2002-0433 - bypass access restrictions in HTTP server using HTTP request with a "*"

CAN-2002-0558 - list arbitrary directories in FTP server via ".." and "*.*" sequences in a LIST command.

CAN-2003-1137 - read files or execute CGI scripts in web server via a GET with a "*"

CAN-2003-1207 - FTP server crash via dir with large amount of "." followed by "/*" string.

CAN-2004-0696 - CGI script directory list using "*" (asterisk) character

CAN-2004-0736 - search engine error infoleak via "**"

CAN-2005-0483 - directory traversal in shell scripts implementing special FTP SITE commands, using ".." and "*" characters.

****

EX.CAT.PTRAV.ABS.1. /absolute/pathname/here

CAN-2000-0614 - absolute pathnames specified for output of compressed files

CAN-2004-1277 - FTP server allows file writing using arguments with "/"

CAN-2003-0753 - PHP local file inclusion using full pathname

CVE-2000-1196 - file read using parameter to error page generator script

****

EX.CAT.PTRAV.REL Other relative path directory traversal examples

CAN-2002-1982 - directory traversal ".." allows file existence disclosure from infoleak (inconsistent error messages)

CAN-2002-1837 - ".." directory traversal used to determine existence of filenames by resultant infoleak (inconsistent error

messages)

CAN-2002-1813 - directory traversal in web-friendly client using the href attribute of a link

****

EX.CAT.FILEEQ.1. filedir. (trailing dot)

CAN-2005-0622 - read PHP source code via trailing dot or trailing space

CAN-2002-1997 - filter bypass using trailing dot after file extension

CAN-2002-1855, CAN-2002-1856, CAN-2002-1857, CAN-2002-1858, CAN-2002-1860, CAN-2002-1861 - trailing dot in directory name allows retrieval of protected files

CVE-2002-1986 - read CGI script source code using trailing dot

****

EX.CAT.FILEEQ.5. filedir[SPACE] (trailing space)

CAN-2005-1656 - source code disclosure by trailing hex-encoded space (manipulation of equivalence property by alternate encoding leading to improper handler deployment)

****

EX.CAT.FILEEQ.10. //multiple/leading/slash ("multiple leading slash")

CVE-1999-1456 - web server read files via multiple leading slashes

CVE-2002-0275, CAN-2002-1238 - multiple leading slashes in web server

CAN-2004-1878 - access administrative scripts via double leading /

CAN-2005-1908 - bypass access restrictions using extra leading / or \

CAN-2005-1365 - Product tries to remove ".." sequences by incrementing and decrementing a counter, but multiple leading slashes prevent the counter from reaching the expected value.

CAN-2004-1032 - MFV. multiple leading slashes fill a buffer so

that a static filename can't be appended to the buffer, leading to file deletion.

****

EX.CAT.INFO.LEAK.ERR. Error Message Infoleak Examples

CAN-2001-1437 - error message infoleak from invalid (non-numeric) value

CAN-2002-1677 - infoleak by error message from invalid value

CAN-2002-2008 infoleak in error message using invalid (non-existent) identifier

CAN-2002-2045 - path disclosure via error message infoleak using invalid parameter

CAN-2002-1822 - web server error message infoleak via request to JSP page that does not exist.

CAN-2002-1723 - path disclosure infoleak via error message from invalid (non-existent) user name

CAN-2002-1728 - path disclosure infoleak via error message from invalid (non-existent) file name

CAN-2002-1801 - error message infoleak from invalid (nonexistent) value

****

EX.CAT.INFO.LEAK.OTHER. Other Information Leak Examples

CAN-2002-1943 - infoleak - internal IP address (alternate name) leaked to external entity

CAN-2002-0284 - client leaks absolute path to server

CAN-2002-1934 - leaks sensitive password information to the screen during boot, requiring physical access to exploit

CAN-2002-2006 infoleak by example code

CAN-2001-1499 - user enumeration by infoleak in response

discrepancy, also authentication method disclosure via intentional infoleak

CAN-2002-1940 - infoleak by writing extraneous sensitive data into unused portion of a compiled program

CAN-2000-1237 - user enumeration from infoleak of early error reporting (behavioral infoleak or response infoleak?)

CAN-2001-1532 - authentication infoleak in URLs, allowing user session hijacking e.g. by obtaining HTTP referer URLs

CAN-2003-0105 - intermediary does not obfuscate certain responses, leading to infoleak that identifies the type of web server running

CVE-2001-1382 - behavioral infoleak in security countermeasure of enrypted communications product

CVE-1999-1099 - malformed UDP packet causes error string that inadvertently includes sensitive information

CAN-2001-1571 - most recently logged in user is sent in cleartext

CAN-2005-2226 - inadvertent infoleak

CAN-2005-2285 - information leak - info stored in externally accessible resources (URLs, web pages, config files)
--> a containment error?

CAN-2002-1888 - privacy leak.

CAN-2004-2226 - infoleak by causing victim to connect to attacker server to download a CSS file (alternate channel?). Similar to CSRF? Kind of a behavioral infoleak, not on product but on user. Other associated vulns can allow user to be forced into doing something.

CAN-2005-1760 - infoleak of password from intermediate report

CAN-2005-1728 - includes credentials in log file

****

EX.CAT.INFOLOSS.OMIT. Omission of Security-relevant Information

CVE-2000-0937 - does not log failed logins if username is correct but password is wrong

CAN-2001-0471 - repeated login attempts not recorded

CAN-2001-0471 - SSH daemon does not log repeated login attempts

CVE-2001-0056 - does not log invalid logins

CVE-2001-0978 - does not record failed login attempts

CAN-2000-0118 - does not log failed password guesses if process is killed before timeout

CAN-2004-1357 - does not properly log IP addresses as result of other error

CAN-2002-1839 - sender's IP address not recorded in message headers, allowing information hiding

****

EX.HANDLER.WRONG. Improper handler deployment

CVE-2000-0682 - source code disclosure by inserting string into URL that invokes a servlet

CVE-2000-0778 - soruce code disclosure via a specific header in an HTTP request

CVE-2001-0126 - arbitrary Java execution via a style sheet that redirects to another source

CVE-2005-1112 - source code disclosure when an invalid HTTP header causes the server to process the page instead of the proper engine

****

EX.CAT.MULTINT. Multiple Interpretation Error (MIE)

CAN-2004-0935, CAN-2004-0937 - compressed file with headers set to 0 cause the file to be ignored an anti-virus product, but the compression software still handles it.

CAN-2001-1542 - multiple interpretation error: intermediary allows improperly MIME-encoded email attachments that can be processed by certain clients.

CAN-2002-0440 - Content-Length of 0 causes HTTP proxy scan to be skipped, but web clients may ignore the Content-Length.

CAN-2003-1015 - multiple virus products allow content restriction bypass using unusual whitespace manipulations.

CAN-2002-1776 - AV product does not scan files with .nch and .dbx extensions, which are automatically recognized and processed by another product (incomplete blacklist)

CVE-2002-0714 - FTP proxy does not compare the IP addresses of control and data connections with the FTP server, allowing firewall rules to be bypassed.

CAN-2002-0285 - MIE. mail client treats carriage return in mail headers as if they are CRLF, allowing filter bypass

CAN-1999-1053 - interaction error/MIE. Product cleanses SSI commands between "<!--" and "--> separators, but underlying web server allows other closing sequences.

CAN-2001-1548, CAN-2001-1549 - bypass firewall on Windows via non-standard TCP packets using non-Windows protocol adapters

****

EX.CAT.RESLEAK.FILEDESC UNIX file descriptor leak

CAN-2002-0677 - file descriptor argument is used as an array index

CAN-2004-1033 - file descriptor leak allows read of restricted files

CVE-2000-0094 - access to restricted resource using modified file descriptor for stderr

CVE-2002-0638 - open file descriptor used as alternate channel in complex race condition

CVE-2002-0766 - MFV. attacker fills file descriptor table then closes a descriptor e.g. for stderr, which leads to unhandled error condition when privileged program can't assign an alternate descriptor.

CVE-2000-1108 - does not verify that file descriptor is a TTY, allowing file corruption via symlink

CAN-2001-1047 - race condition allows one thread to set a file descriptor to NULL, creating stale handle in the other thread.

CAN-2003-0489 - program does not fully drop privileges after creating a file descriptor, which allows access to the descriptor via a separate vulnerability

CAN-2003-0937 - user bypasses restrictions by obtaining a file descriptor then calling setuid program, which does not close the descriptor.

CAN-2002-1866 - file descriptors not closed for HTTP 404 messages, leading to resource consumption

CAN-2005-1922 - file descriptor consumption after input triggers errors

CVE-2004-2215 - terminal manager does not properly close file descriptors, allowing attackers to access terminals of other users

****

EX.CAT.RESOURCE.AMP.ALG. Algorithmic Complexity

CAN-2005-2505 - certain Gregorian dates cause CPU consumption due to algorithmic complexity

CVE-1999-1537 - web server allows SSL requests to HTTPS port for normally unencrypted files, requiring extra work for the server.

CVE-2000-1184 - attackers can specify a large file in the TERMCAP variable, causing the server to consume resources while processing the file.

CAN-2001-1244, CAN-2004-0002 - both amplification and complexity

****

EX.RESOURCE.POOL. Insufficient Resource Pool

CAN-2002-0234 - product does not impose maximum limit on connections, allowing port scan to consume all available connections

CAN-2002-1063 - large number of FTP PASV requests consumes all available FTP ports.

****

EX.RESOURCE.LOCK. Unrestricted critical resource lock

   CAN-2002-1963 - resource exhaustion by local users due to low
   resource limit

   CAN-2005-2283 - no input size restriction allows resource
   consumption

   CAN-2001-1518 - product in multi-user environment only supports one
   session at a time, leading to resource exhaustion by creating a
   named pipe session

   CAN-2004-2164 - database connection not closed

   CAN-2002-1866 - product does not close file descriptors for 404
   error messages, leading to resource exhaustion (resource leak?)

   CAN-2005-2241 - "resource leak" by not quickly "timing out"
   inactive sockets

  ** Resource hijacking

   CAN-2002-1827 - DoS by obtaining an exclusive lock

   CAN-2002-1869 - local user prevents log files from being opened for
   writing

   CAN-2005-2070 - keeping connection open prevents product from
   reloading

   CAN-2002-1914, CAN-2002-1915 - attacker uses a file lock to prevent
   a program from executing

****

EX.CAT.INT.SIGN. Integer Signedness Error

   CVE-2003-0075 - negative offset value used - overlaps array index?

   CAN-2002-0973 - large negative values to OS system calls allow
   kernel memory access

   CAN-2003-0619 - negative size value

CAN-2003-0721 - negative number used as array index

CAN-2003-0972 - large number of special characters in escape sequences trigger signedness error and lead to buffer overflow

CAN-2005-0340 - negative string length

CAN-2004-1035 - signedness errors cause crash or memory infoleak

CAN-2004-0493 - large number of special characters leads to signedness error and overflow on 64-bit systems

CAN-2002-1355 - signedness error leads to infinite loop

CVE-2002-1373 - large negative integers used in memory copy call

CAN-2003-0372 - negative value to interpreted language function leads to signedness error underneath

CAN-2005-1263 - negative length passes signed integer comparison and leads to buffer overflow

CAN-2002-1062 - signedness error triggered by long inputs?

CVE-2002-0036 - integer signedness error when large unsigned data element length is later used as a negative value.

CVE-2002-1420 - integer signedness error in OS system call via a negative size value, which passes check as a signed integer but is later used as an unsigned integer when copying data.

CAN-2002-0973 - integer signedness error in several system calls via large negative values allows sensitive memory access

CVE-2001-1279 - invalid lengths trigger signedness error and lead to buffer overflow

CVE-2002-0036 - large unsigned data length is later used as negative value

CVE-2002-1420 - negative size value satisfies maximum value check as signed integer, but is later used as unsigned value.

****

EX.CAT.INT.OVERFLOW. Integer Overflows

CVE-2005-3278 - [code excerpt available] integer overflow in malloc calculation causes malloc of less memory than expected, leading to buffer overflow.

CVE-2001-0144 - integer overflow in security patch

CVE-2002-0639 - integer overflow during challenge/response authentication

CAN-2005-1704 - integer overflow via file that claims large number of headers, leading to heap overflow

CAN-2005-1693 - name length of -1 leads to heap overflow

CAN-2005-1545 - integer overflow in parser for executable file leads to heap overflow

CAN-2005-1521 - integer overflow via large value in parameter

CAN-2004-0990 - integer overflow in image files with large image rows value

CAN-2005-0736 - integer overflow by creating a large number of events

CAN-2004-1503 - large number of requests causes variable to wrap around (step-based manipulation)

CAN-2004-1311 - content length field of -1 leads to heap overflow

CAN-2004-1049 - integer overflow in image with large size field

CAN-2004-0657 - integer overflow in NNTP server when a client requests a time 34 years in the future

CAN-2003-0357, CAN-2004-0633 - integer overflow in network sniffer

CAN-2004-0417 - integer overflow causes server crash and resultant un-deleted temporary resources

CAN-2004-0216 - long file name triggers integer overflow when calculating a buffer length

CAN-2004-0184 - integer overflow in ISAKMP Identification payload

leads to small byte count during conversion, then out-of-bounds read

CAN-2005-1693, CAN-2005-1704 - integer overflow leading to heap overflow

CAN-2004-0431 - large "number of elements" field triggers integer overflow then resultant heap overflow

CAN-2005-1513 - integer overflow only on 64-bit platforms with large amounts of virtual memory

CAN-2004-1308 - integer overflow with "-1" count, leading to resultant heap overflow

****

EX.CAT.ERR. Unhandled Error Condition / Unchecked Error Condition

CAN-2005-2617 - return value from function not checked, leading to resultant memory leak

CAN-2004-1070 - return values from certain calls not properly checked

CAN-2001-1324 - return value to function call not properly checked, allowing setuid to user-specified UID.

CAN-2005-0078 - unchecked value from a function call allows attackers with physical access to cause a crash

CVE-2004-0077 - return value not checked when maximum number of descriptors is exceeded

CVE-2002-1372 - return values of file/socket operations not checked, allowing resultant consumption of file descriptors

CAN-2002-0717 - HTTP POST request with unspecified manipulations leads to unhandled error condition, then a free of improper memory.

CAN-1999-1434 - login program does not check for an error when a key file is missing, preventing privileges from being dropped

CAN-2005-0255 - string handling functions do not check return values of other functions, causing a reallocation to fail when memory is exhausted, leading to the wrong pointer being returned.

CAN-2005-2151 - failure in DNS not properly handled

CAN-2003-0690 - product does not verify whether a function call succeeds, allowing privilege escalation by trigging certain error conditions

CAN-2005-1795 - MFV. shell metacharacters if permissions prohibit a delete action from being successful.

****

EX.CAT.RAND. Randomness and Predictability

CVE-2002-1107 - product does not generate sufficiently random numbers, which may make it vulnerable to attacks like spoofing.

CAN-2002-1935 - bypass registration using predictable IDs

CAN-2000-0916 - insufficient random number generator used to generate initial TCP sequence numbers, allowing spoofing

CAN-2005-1631 - view private bookmarks by guessing IDs

****

EX.CAT.DOCROOT.WEB. Sensitive Data Under Web Root

CAN-2005-1835, CAN-2005-2217 - data file under web root

CAN-2002-1449 - username/password in data file under web root

CAN-2002-0943, CAN-2005-1645 - database file under web root

SECTION.10.5. [UNCAT] Additional Uncategorized Examples

The following examples have not been categorized yet, for any of several reasons:

- they might require more diagnosis
- they might be multi-factor, and it might not be certain which WIFF would demonstrate them most effectively

- they do not cleanly fit into the current PLOVER classes, i.e. are outliers

These uncategorized examples can also be used as a "stress test" for other classification efforts.

****

EX.UNCAT. Miscellaneous Uncategorized Examples

CVE-2001-0268 - user can access restricted kernel memory using an address that is out of the expected range.

CVE-2005-4412 - attacker with access to user's session can read plaintext passwords from window/GUI, which "hides" the passwords using asterisks although the plaintext version is still accessible from a tool. Also an unprotected alternate channel problem.

CVE-2005-2923 - invalid memory reference, triggered by long input manipulation - NOT a buffer overflow.

CVE-2005-3784 - product accidentally removes entities that are still in use by other processes, leading to resultant "dangling reference"

OSVDB:11002 - MFV. GUI race condition, certificate handling problem. Product allows MITM until user finishes interacting with a dialog box.

CVE-2005-3783 - functional change causes a routine to fail to properly determine if an action is being performed against itself

CVE-2005-3505 - XSS specific to a single web browser only.

CVE-2005-3432 - list protected files using wildcard character

CVE-2005-3494 - missing authorization check, as a subgroup of authorization error - but overlap w/privs

CAN-2001-1570 - step manipulation leads to lockout

CAN-2005-1671 - single common log file writable/readable by multiple users, leading to infoleak. General design issue: single-user app on a multi-user system

CAN-2005-1743 - doesn't handle a particular exception that is thrown, leading to audit information loss or incorrect identity

CAN-2005-2969 - an option in a product disables a step that is needed to prevent certain man-in-the-middle attacks that cause a weaker protocol to be selected.

==========

EX.UNCLASS. Unclassifiable Examples

These examples contain sufficient detail to understand the problem(s), but there is not a WIFF in PLOVER that sufficiently captures the problem.

CAN-2001-1547 - does not block attachments from forwarded messages; alternate path issue?

CAN-2002-2028 - does not check if an account is locked before accepting a valid password, allowing bypass of policy

CAN-2004-2182 - session fixation vuln. One factor: making a mutable variable immutable.

CVE-2005-3323 - a file inclusion issue, but not PHP?

CVE-2004-2306 - certain OS configuration, when one package has been removed, disables a security setting and allows detection avoidance.

CAN-2002-1932 - full event log does not trigger an alert in certain configs.

CAN-2002-1937 - MAC address hard-coded within configuration, allowing ARP spoofing.

CVE-2005-3275 - product declares a variable to be static, when it could be modified at the same time by two different entities.

CAN-2005-3254 - program checks for minimum value for UIDs that it should not seteuid to, but the minimum value is too small, thus allowing seteuid to some system accounts with UIDs that exceed the minimum. Generally speaking, involves an incorrect/incomplete specification of all the members of a generic group; conceptually it is very similar to a permissive blacklist.

CAN-2005-2176, CAN-2005-2177 - mail client automatically processes HTML in an attachment instead of prompting the user first. Could be

unprompted dangerous action, maybe resultant.

CAN-2004-0461 - product, when compiled in certain environments, uses insecure versions of library functions for handling strings, enabling resultant buffer overflows.

CVE-2001-0850 - configuration error uses insecure versions of safe library functions, enabling buffer overflows

CVE-2004-0652 - obtain username and password by directly accessing internal methods.

CVE-2005-4345 - access to administrator password hash can be obtained by using an API call.

CAN-2002-2013 - cookie theft from other domains using null character followed by target domain

==========

EX.RESEARCH. Interesting Examples requiring further research

These examples identify manipulations or WIFFs that require additional research or investigation to classify.

CAN-2002-1747, CAN-2002-1755 - cut-and-paste attacks in crypto/authentication

CAN-2002-1849 - resource consumption (users) by not performing "standard" logout - step-based manipulation

CVE-2005-2798 - product can delegate GSSAPI credentials to clients who don't use GSSAPI methods, leading to exposure of those credentials (information leak/redirection)

CAN-2005-2069 - Interaction Error? Product uses TLS when connecting to a slave, but does not use TLS when referred to the master.

CAN-2003-0496 - attacker gains privileges by providing a named pipe as an argument to a function, instead of a normal file

CAN-2005-0051 - anonymous login to named pipe leads to information disclosure

CVE-2005-3280 - could be a problem with a default password, an unrestricted channel, or a combination of both.

CVE-2004-2415 - XML entity expansion attack

CAN-2005-1723 - an odd issue

CVE-2004-2338 - product does not properly parse certain access rules on big-endian 64-bit platforms, leading to filter bypass

CAN-2005-0268 - PHP script executes code specified in a particular parameter. (not clear whether dynamic or static code injection)

CAN-2005-0593 - GUI spoof. Web browsers allow remote attackers to spoof the "security" icon via several different methods

CAN-2003-0592, CAN-2003-0593, CAN-2003-0594 - web browser sends cookies outside domain with "%2e%2e"

CAN-2002-1667 - OS does not check the existence of a particular object, which can be triggered by unexpected user actions

CAN-2001-1535 - iDEFENSE "Brute-Forcing Web Application Session IDs"

CAN-2002-1983 - DoS (hang) via multiple timers with a 1-ms tick, possibly involving asymmetric consumption or resource management error

CAN-2005-2068 - OS allows modifying TCP options of existing session with TCP packet with SYN flag for already-existing session

CAN-2005-2114 - CPU consumption, possibly involving infinite recursion

CAN-2005-2134 - unusual manipulation leads to divide-by-zero, possibly due to race condition or unhandled error

CAN-2005-1640 - missing or erroneous authorization check - doesn't check privileges

CAN-2005-1992 - default security-critical configuration value is invalid

CAN-2002-1712 - Flood of empty packets with ACK/FIN bit set. Note: might not be MFV if the packets don't have to be empty.

CAN-2005-1762 - unknown vuln with "non-canonical" manipulation

CAN-2005-1703 - incomplete packet triggers null dereference

CAN-2005-1740 - insecure tempfile creation allows writing by symlink *or* execution by modification of file (bad permissions)

CVE-2000-0353, CAN-2002-0317, CAN-2001-1192 - automatic download and execution by various means, not direct dynamic code execution

CAN-2005-1638 - special characters, possibly by unusual manipulation, which enables XSS

CAN-2003-0336 - mail client allows file read via an email message with a carriage return character in a spoofed special string that is associated with the message. Special sequence/element injection, not special char.

==========

EX.UNDIAG. Interesting Examples that were not diagnosed

These examples identify interesting manipulations or resultant WIFFs, but the primary WIFFs are unknown due to the lack of diagnosis by the researcher and/or vendor.

CAN-2002-1871 - product installs files with setuid/setgid privileges if unusual entries are found in certain fields.

CAN-2005-1720 - ACL handling error during file copy prevents permissions from being properly applied to the copied file

CVE-2005-1992 - a change between software versions causes dangerous methods to be allowed when they weren't in earlier versions.

CAN-2001-0617 - attacker can gain access to services even if individual services have been disabled

CVE-2000-0915 - finger allows read files by specifying filename instead of username

CAN-2005-2301 - LDAP injection?

CVE-2002-1132 - diagnostic error? malformed argument to script causes error message when the file cannot be included in the script; reported as error message infoleak.

CAN-2002-1925 - certain portscans cause crash when admin selects a

particular log tab in the GUI

CAN-2002-1988 - long variables for non-existent resources trigger memory consumption and hang

CAN-1999-1265 - special character "(" in SMTP arguments causes CPU consumption.

CAN-1999-0347 - web browser allows file reading and spoofing of pages using "%01" character in URL, causing browser to use the domain specified after the "%01"

CAN-2000-1138 - mail program does not notify user when S/MIME message has a broken signature. Could be resultant or primary UI, or primary CRYPTO.

CAN-2002-2080 - memory consumption via large number of RCPT TO headers

CAN-2004-2151 - memory allocation via large data size leads to memory consumption or crash

CAN-2002-2009 - path disclosure via certain characters before .jsp extension including "+" (space equivalent), ">/", "</", and "%20" (hex-encoded space)

CAN-2002-1397 - API function for language interpreter allows DoS via negative argument, underlying WIFF unknown but possibly signedness error.

CAN-2003-0418 - IP stack does not properly calculate the size of a response, causing memory infoleak in ICMP error response

CAN-2005-2053 - diagnostic error? wildcard "*" injection causes infoleak and possibly directory traversal

CAN-2005-2076 - "@" character not properly handled in a password, leaking part of password to the screen; possibly special char

CAN-2004-0470 - inadvertent removal of security-relevant tags when a key datum is missing

CAN-2005-1932 - multiple examples of critical variable modification

CAN-2004-2197 - does not check ownership of files

CAN-2004-2225 - crafted data: URI has unexpected consequences (delete files)

CAN-2005-1659 - triple dot and XSS

CAN-2005-1663 - unusual manipulation "://" in an HTTP request

CVE-2004-2516 - directory traversal bug with unusual syntactic manipulations; underlying bug unclear

CAN-2005-1774 - unknown issue prevents enforcement of permissions

CAN-2005-1791 - near-equivalence causes crash (domain name looks like an IP address)

CVE-2001-0969 - keyword in firewall rules not honored when certain interfaces are used

CVE-2001-0866 - outbound ACL not handled if an inbound ACL is not configured on all interfaces

CAN-2004-1001 - password check function does not properly handle error from a function call

CAN-2005-1956 - unknown WIFF allows bypass of file extension check using unusual "~~~~~~" data manipulation

SECTION.11. References

[Paget] "Exploiting design flaws in the Win32 API for privilege escalation. Or... Shatter Attacks - How to break Windows." August, 2002 http://security.tombom.co.uk/shatter.html

[Segal] "HTTP Request Smuggling" Ory Segal June 2005 http://www.watchfire.com/resources/HTTP-Request-Smuggling.pdf

[SPI] "Web Applications and LDAP Injection" SPI Dynamics

[PeterW] "Cross-Site Request Forgeries (Re: The Dangers of Allowing
Users to Post Images)"
Peter W
Bugtraq
June 15, 2001
http://cert.uni-stuttgart.de/archive/bugtraq/2001/06/msg00216.html

[Moore] "0x00 vs ASP file upload scripts"
Brett Moore
July 13, 2004
http://www.security-assessment.com/Whitepapers/
0x00_vs_ASP_File_Uploads.pdf

[RFP] "Poison NULL byte"
Rain Forest Puppy
Phrack

[LitchfieldBU] "Buffer Underruns, DEP, ASLR and improving the
Exploitation Prevention Mechanisms (XPMs) on the Windows platform"
September 30, 2005
http://www.ngssoftware.com/papers/xpms.pdf

[PtacekNewsham] "Insertion, Evasion, and Denial of Service: Eluding
Network Intrusion Detection"
Thomas H. Ptacek, Timothy N. Newsham
January 1998
http://www.insecure.org/stf/secnet_ids/secnet_ids.pdf

[Zalewski2001] "Strange Attractors and TCP/IP Sequence Number Analysis"
Michal Zalewski
2001
http://www.bindview.com/Services/Razor/Papers/2001/tcpseq.cfm

[Watts] "Discovering and Exploiting Named Pipe Security Flaws for Fun
and Profit," Blake Watts
http://www.blakewatts.com/namedpipepaper.html

[Howard2002] "When scrubbing secrets in memory doesn't work"
Michael Howard
Bugtraq
Nov 5, 2002
http://cert.uni-stuttgart.de/archive/bugtraq/2002/11/msg00046.html

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dncode/html/secure10102002.asp

[Wagner] "GNU GCC: Optimizer Removes Code Necessary for Security"
Joseph Wagner
Bugtraq
November 16, 2002
http://www.derkeiler.com/Mailing-Lists/securityfocus/bugtraq/2002-11/0257.html

"A Critical Analysis of Vulnerability Taxonomies
Matt Bishop and David Bailey
CSE-96-11
September 1996"

http://seclab.cs.ucdavis.edu/projects/vulnerabilities/scriv/ucd-ecs-96-11.pdf

http://www.laas.fr/IFIPWG/Workshops&Meetings/44/W1/09-Iyer.pdf
** "FSM" model
** exploit involves multiple operations on several objects; exploits must pass through
"elementary activities" where each one is an opportunity for security check

[SanctumX] Sanctum, "Blind XPath injection", May 19, 2004
http://www.sanctuminc.com/pdfc/WhitePaper_Blind_XPath_Injection_20040518.pdf

[Clowes] A Study in Scarlet

[Flake2001] Halvar Flake, "Third Generation Exploits" presentation at
Black Hat Europe 2001.
http://www.blackhat.com/presentations/bh-europe-01/halvar-flake/bh-europe-01-halvarflake.ppt

[Christey2004a] "Off-by-one errors: a brief explanation"
Steve Christey
Secprog and SC-L mailing list posts
May 5, 2004

Secprog:
http://marc.theaimsgroup.com/?

l=secprog&m=108379742110553&w=2
http://marc.theaimsgroup.com/?
l=secprog&m=108379754014251&w=2

SC-L:
[need to get URL]

[Christey2005] "Second-Order Symlink Vulnerabilities"
Steve Christey
Bugtraq
June 7, 2005
http://www.securityfocus.com/archive/1/401682

[Christey2005b] "On Interpretation Conflict Vulnerabilities"
Steve Christey
Bugtraq
November 3, 2005

[Colley2004a] "Crafting Symlinks for Fun and Profit"
Infosec Writers Text Library
April 12, 2004
http://www.infosecwriters.com/texts.php?op=display&id=159

[Crosby] "Algorithmic Complexity Attacks" (Crosby, Wallach)
http://www.cs.rice.edu/~scrosby/hash/CrosbyWallach_UsenixSec2003/
index.html

[klog1999] klog, "The Frame Pointer Overwrite" September 9, 1999, in
Phrack Issue 55, Chapter 8
http://kaizo.org/mirrors/phrack/phrack55/P55-08

[McHog] "Exploiting Sotware" Gary McGraw, Greg Hoglund

[Newsham] Format String Attacks
Tim Newsham, Guardent
September 2000
http://www.lava.net/~newsham/format-string-attacks.pdf

[Skoll] "Re: Corsaire Security Advisory - Multiple vendor MIME RFC2047
encoding"
David F. Skoll
Bugtraq
September 15, 2004
http://marc.theaimsgroup.com/?l=bugtraq&m=109525864717484&w=2

[Corsaire] "Re:[2] Corsaire Security Advisory - Multiple vendor MIME

RFC2047 encoding issue"
Martin O'Neal
Bugtraq
September 15, 2004
http://marc.theaimsgroup.com/?l=bugtraq&m=109551582712011&w=2

[Younan2003] "An overview of common programming security
vulnerabilities and possible solutions"
Yves Younan
Student thesis
August 2003
http://fort-knox.org/thesis.pdf

[blexim] "Basic Integer Overflows"
blexim
Phrack Issue 60, Chapter 10
http://www.phrack.org/phrack/60/p60-0x0a.txt

[Harnhammar] "CRLF Injection"
Ulf Harnhammar
Bugtraq
May 7, 2002
http://cert.uni-stuttgart.de/archive/bugtraq/2002/05/msg00079.html

SECTION.12. Contributors / Acknowledgements

This document identifies over 1400 specific vulnerabilities in real-world products. This would not be possible without detailed, public vulnerability reports by security researchers everywhere, from bored teenagers to seasoned professionals. Some variants or manipulations were first discovered by previously unknown researchers in rarely-used products.

Antonomasia and Mads Rasmussen suggested that the original vulnerability auditing checklist should be annotated with specific examples.

Pascal Meunier was a good sounding board for early versions of some of the concepts that evolved into PLOVER.

In 2005, many people started discussing vulnerability classification

again, especially the webappsec list. This inspired a final effort to provide a workable update to previous documents. Bob Martin was especially encouraging in getting this version out.

SECTION.13. Change Log

0.1 - 2005/07/20
  [*] created PLOVER acronym
  [*] developed outline
  [*] copied over vulnerability auditing checklist (0.000026)
  [*] reviewed and added many new examples
  [*] refined manipulations, properties, etc. in separate document

0.2 - 2005/07/21
  [*] refined categories
  [*] categorized most CVE's

0.3 - 2005/07/22
  [*] wrote intro
  [*] reorganized many categories; began merging checklist gaps with categories
  [*] new concepts - multi-channel attack, facilitator manipulation
  [*] added theories and observations

0.4 - 2005/07/23
  [*] more work on categories
  [*] started "bad practices" section
  [*] wrote up short descriptions for some CVE's, collected more examples

0.5 - 2005/07/24
  [*] more short descriptions for some CVE's, added more examples
  [*] continued categorical refinement and merging

0.6 - 2005/07/25
  [*] cleaned up document formatting
  [*] fleshed out some more examples

0.7 - 2005/07/26
  [*] wrote Core Definitions
  [*] merged data properties/manipulations from elsewhere, then refined
  [*] started Problems with Existing Terminology
  [*] added Multi-Factor Vulnerabilities

0.8 - 2005/07/27
  [*] more definitions
  [*] wrote Alternate Elements
  [*] wrote sections for channels and endpoints
  [*] functional consequences
  [*] more organization

0.9 - 2005/07/28
  [*] more cleanup, a little more formatting
  [*] more writing
  [*] started moving extra examples to appendix

0.10 - 2005/07/29
  [*] more cleanup, more examples
  [*] wrote various definitions, research gaps, terminology notes

0.11 - 2005/07/30
  [*] still more cleanup, more WIFF definitions

0.12 - 2005/07/31
  [*] finished up WIFF definitions and examples, reorganized
  [*] more formatting
  [*] added Unprotected Alternate Channel
  [*] summarized WIFF list
  [*] cleaned up Hypotheses, Diagnostic Errors, Genesis,

0.13 - 2005/08/01
  [*] cleaned up chapter/section names
  [*] added more intro text
  [*] reorganized sections
  [*] added Direction and Location of Channels, expanded multi-channel
      attacks accordingly

0.14 - 2005/08/02
  [*] finished categorizing WIFFs
  [*] renumbered entire document
  [*] reordered more sections

0.15 - 2005/08/03
  [*] fixed typos

0.16 - 2005/08/19
  [*] more examples
  [*] minor modifications over past couple weeks
  [*] renamed "eval injection" issue

0.17 - 2005/10/03
  [*] minor modifications
  [*] generated HTML version

0.18 - 2005/10/11
  [*] added Litchfield paper on buffer underruns
  [*] more examples

0.19 - 2005/10/20
  [*] more notes on intended information leak
  [*] merged "Mutable Search Path" into "Untrusted Search Path" - not
      sure why there were two in the first place.

0.20 - 2005/10/27
  [*] began mapping CLASP root causes to PLOVER
  [*] began replacing BID's with CVE's
  [*] began mapping Pernicious Kingdoms to PLOVER
  [*] worked on categorizing un-categorized examples
  [*] added MFV examples section

0.21 - 2005/11/05
  [*] added descriptions to remaining CVE's in examples section
  [*] moved more uncategorized examples to proper categories

0.22 - 2005/11/06
  [*] moved more uncategorized examples to proper categories
  [*] added some more categories

0.23 - 2006/02/22
  [*] finished up examples
  [*] made more MISC categories
  [*] froze existing set of categories

0.24 - 2006/03/15
  [*] Major changes since 0.17: REMOVED categories SPECTS.XSS.NULL,
      CP.UPATH.MUTABLE
  [*] Major changes since 0.17: ADDED categories SPECTS.XSS.INVTAGS,
      INFO.LEAK.PRIVACY, RES.AMP.EARLY, RAND.STATIC, ERS.UNREP,
      VER.OVE, VER.INTEG.MISS, VER.INTEG.INC, VER.PHP-UPLOAD,
      MAL.BDOOR.HPASS, MISC.INVFREE, MISC.ASSERT, MISC.EXTCONF,
      MISC.WEAK, MISC.NONCONFORM.IMP, MISC.NONCONFORM.API, MISC.CDEP,
      MISC.INCACT, MISC.SAMENAME, MISC.ALTNAME, MISC.NEIGHBORNAME,
      MISC.UNDOC, MISC.UNDOC.ACCOUNT, MISC.UNDOC.EGG, MISC.CONF,
      MISC.CONF.DEFPASS, MISC.CONF.ACCESS, MISC.CONF.AUDIT
  [*] Major changes since 0.17: RENAMED categories: SPECTS.NULLTERM,
      PATH.LINK.UNIX.SYM, HAND.UPLOAD, RES.LOCK.INSUFF,

CODE.STAT.PHPINC, ERS.UNCH