



---

# **Towards Threat, Attack, and Vulnerability Taxonomies**

**Dennis Hollingworth**

**Network Associates laboratories**

**[dennis\\_hollingworth@nai.com](mailto:dennis_hollingworth@nai.com)**



## Measuring Assurance in Cyberspace (seedling call)

- ❖ Areas of interest included (but not limited to) the following:
  1. Concepts and terminology to succinctly express IA domain issues;
  - 2. Threat, attack and vulnerability taxonomies;**
  3. Security models and models of attacker intent, objectives, and strategies;
  4. Work factor metrics, survivability metrics, operational security metrics, cryptographic protocol metrics;
  5. Methods for testing and validating protection mechanisms; and
  6. Security and survivability requirements specifications.

- *Premise:* Models of attacker intent, objectives, and strategies will depend, at least in part, on useful threat, attack, and vulnerability taxonomies to parameterize those models.
- *Focus:* Taxonomies that relate to the needs of security model developers and provide a more useful information tool for security analysts.
- *Scope:* Modest seedling effort to explore...
  - Unifying/bridging taxonomy concepts, content, and organizational structure
  - Tools to support the rapid development of specialized or focused taxonomies

But...with metrics/measurability emphasis

- IT'S ALWAYS BEEN HARD (INTRACTABLE???)
  - Characterizing security events in a potentially measurable way
  - Dealing with intangibles such as software production methodology
  - Going beyond the *Framework Specification* stages to populating those frameworks
  - Identifying **useful** metrics and associated estimation or measurement techniques that are somewhat repeatable
  - Covering threat, attack, & vulnerability space such that error factors don't dominate measurability
  
- Security metrics & measures have been sought since mid 70s

Anyone can specify a taxonomy...“Is it useful for what you want to do?”

*Not an arbitrary assignment of information to categories:*

- *Mutually exclusive* - classifying in one category excludes all others because categories do not overlap,
  - *Exhaustive* - taken together, the categories include all possibilities,
  - *Unambiguous* - clear and precise so that classification is not uncertain, regardless of who is classifying,
  - *Repeatable* - repeated applications result in the same classification, regardless of who is classifying,
  - *Accepted* - logical and intuitive so that they could become generally approved,
  - *Useful* - can be used to gain insight into the field of inquiry.
- Approximation -- will fall short when the data being classified are imprecise and uncertain, as with security events



# Requirements for IAM Taxonomies



- Unification of different types of attack/vulnerability databases
  - incorporating organizational structure that unifies and bridges taxonomy concepts and content (e.g., virus and classical attack databases),
  - relating traditional attack information and malicious code descriptions, and
  - providing schema support for data slices at multiple levels of abstraction.
- Extended organizational structure
  - provide feature inheritance to support representation of evolutionary properties of attacks, for example, a genealogical approach to encoding the evolutionary nature of viruses and worms and
  - support temporal, causal and other content relationships.
- Encompass a larger development and maintenance environment that supports taxonomy derivation and customization tools
  - facilitate rapid creation of specialized taxonomies,
  - emphasize specific attacker objectives, and
  - capture the modeler's own organizing strategies or goals.

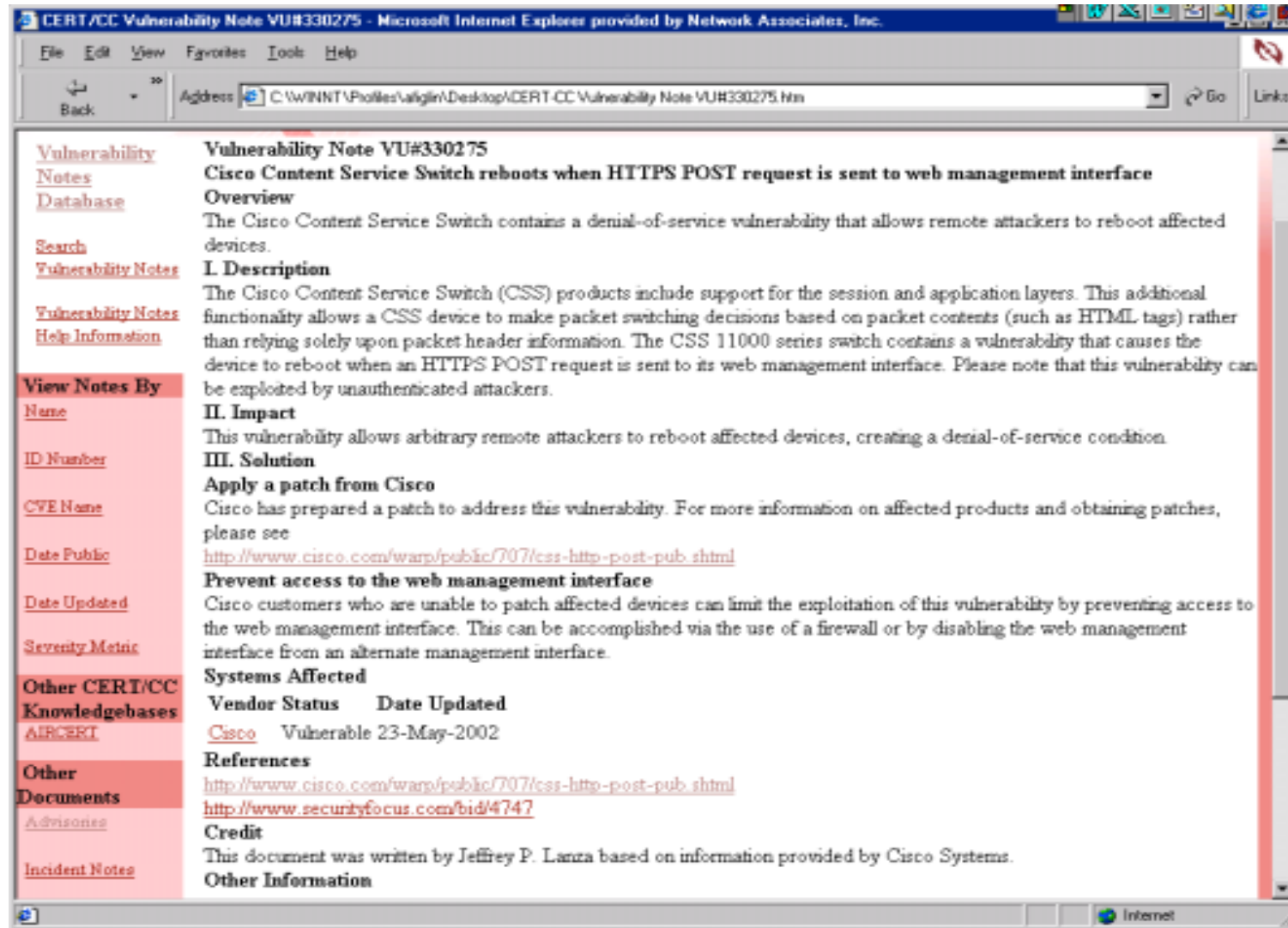
- ❖ *Measurement-based taxonomies must be **discovered** from the bottom up based upon understanding the details of a broad collection of threats, attacks, and vulnerabilities. They will not be developed from the top down. The latter approach leads to conceptual taxonomies, rather than those that capture elements amenable to measurement. They muddy the details of the relevant components of vulnerabilities that relate to measurement. Any generalization and abstraction must be predicated upon and preserve the results of deep error analysis and faithfully capture it in data organization and content and not introduce ill-defined concepts or inadvertently mix incommensurable error properties along the way.*

- **Illuminate and express measurable or estimable events or actions through:**
  - **Detail** - expose specific security components amenable to measurement or estimation.
  - **Structure** - relate events to one another, e.g., decompose progressive, multi-stage, and complex incidents such as worms into more useful representations. Represent evolution of attack strategies and other relationships through ordering or inheritance relationships.
  - **Derivation** - specialization: focusing on reduced class of events with specific properties. Reduction tools to help produce specialized taxonomies
  - **Consistency** - integration of a variety of sources: HIDS, NIDS, and BSM knowledge bases; content of major incident and vulnerability databases such as the CERT and NESSUS databases. Variability in content reporting formats and quality, substantial quantity of event reports, reliance upon natural language description, and existing database organizational deficiencies.
  - **Detection-based Classification** - relate taxonomies to detection and observation mechanisms. May provide opportunities for measurement or estimation that are unavailable to concept-based classification because of the latter's association of like events observed through unlike mechanisms.



- Literature review (taxonomies, attack and vulnerability categorizations)
  
- Raw Data Analysis:
  - HIDS (EMERALD/eXpertBSM, Trusted Solaris 8 BSM),
  - NIDS (TippingPoint, ISS Real Secure, BlackIce),
  - Network Associates Laboratories
    - 4 virus databases - (AVERT and McAfee)
    - 4 historical attack and vulnerability databases - COVERT and CyberCop Scanner and Monitor
  - Other Vendor/sources
    - 4 CISCO, SNORT, SYSLOG...
  - Public repositories: CERT, BugTraq, Security Focus, and CVE.
  
- Identify new directions/thrusts

- McAfee ASaP
  - 30 major categories grouping roughly 800 items
  - Somewhat arbitrary numerical assignment based on: exploited system services, attack strategy, specific operating system (e.g., Windows NT)
- AVERT:
  - 8 major categories grouping roughly 5800 different virus classifications
  - Categories broken down into subcategories of a more descriptive nature (based upon a broadly-specified exploitation avenue or mechanism, although some appear to be grouped according to the effect (e.g., Denial of Service) or in catch-all sub-categories (e.g., miscellaneous))
- McAfee Threat Scan (Anti-virus vulnerability scanner)
  - AV-centric vulnerability assessment tool that allows AV administrators to identify and patch security vulnerabilities that can be exploited by hybrid threats like Code Red and Nimda
  - Checks 90 conditions/vulnerabilities related to HTTP, WWW and CGI



**CERT/CC Vulnerability Note VU#330275**  
**Cisco Content Service Switch reboots when HTTPS POST request is sent to web management interface**

**Overview**  
 The Cisco Content Service Switch contains a denial-of-service vulnerability that allows remote attackers to reboot affected devices.

**I. Description**  
 The Cisco Content Service Switch (CSS) products include support for the session and application layers. This additional functionality allows a CSS device to make packet switching decisions based on packet contents (such as HTML tags) rather than relying solely upon packet header information. The CSS 11000 series switch contains a vulnerability that causes the device to reboot when an HTTPS POST request is sent to its web management interface. Please note that this vulnerability can be exploited by unauthenticated attackers.

**II. Impact**  
 This vulnerability allows arbitrary remote attackers to reboot affected devices, creating a denial-of-service condition.

**III. Solution**  
**Apply a patch from Cisco**  
 Cisco has prepared a patch to address this vulnerability. For more information on affected products and obtaining patches, please see <http://www.cisco.com/warp/public/707/css-http-post-pub.shtml>

**Prevent access to the web management interface**  
 Cisco customers who are unable to patch affected devices can limit the exploitation of this vulnerability by preventing access to the web management interface. This can be accomplished via the use of a firewall or by disabling the web management interface from an alternate management interface.

**Systems Affected**

Vendor Status	Date Updated
<a href="#">Cisco</a>	Vulnerable 23-May-2002

**References**  
<http://www.cisco.com/warp/public/707/css-http-post-pub.shtml>  
<http://www.securityfocus.com/bid/4747>

**Credit**  
 This document was written by Jeffrey P. Lanza based on information provided by Cisco Systems.

**Other Information**

**View Notes By**

- [Name](#)
- [ID Number](#)
- [CVE Name](#)
- [Date Public](#)
- [Date Updated](#)
- [Severity Metric](#)

**Other CERT/CC Knowledgebases**

- [AIRCERT](#)

**Other Documents**

- [Advisories](#)
- [Incident Notes](#)



# Nessus Vulnerability Scanner



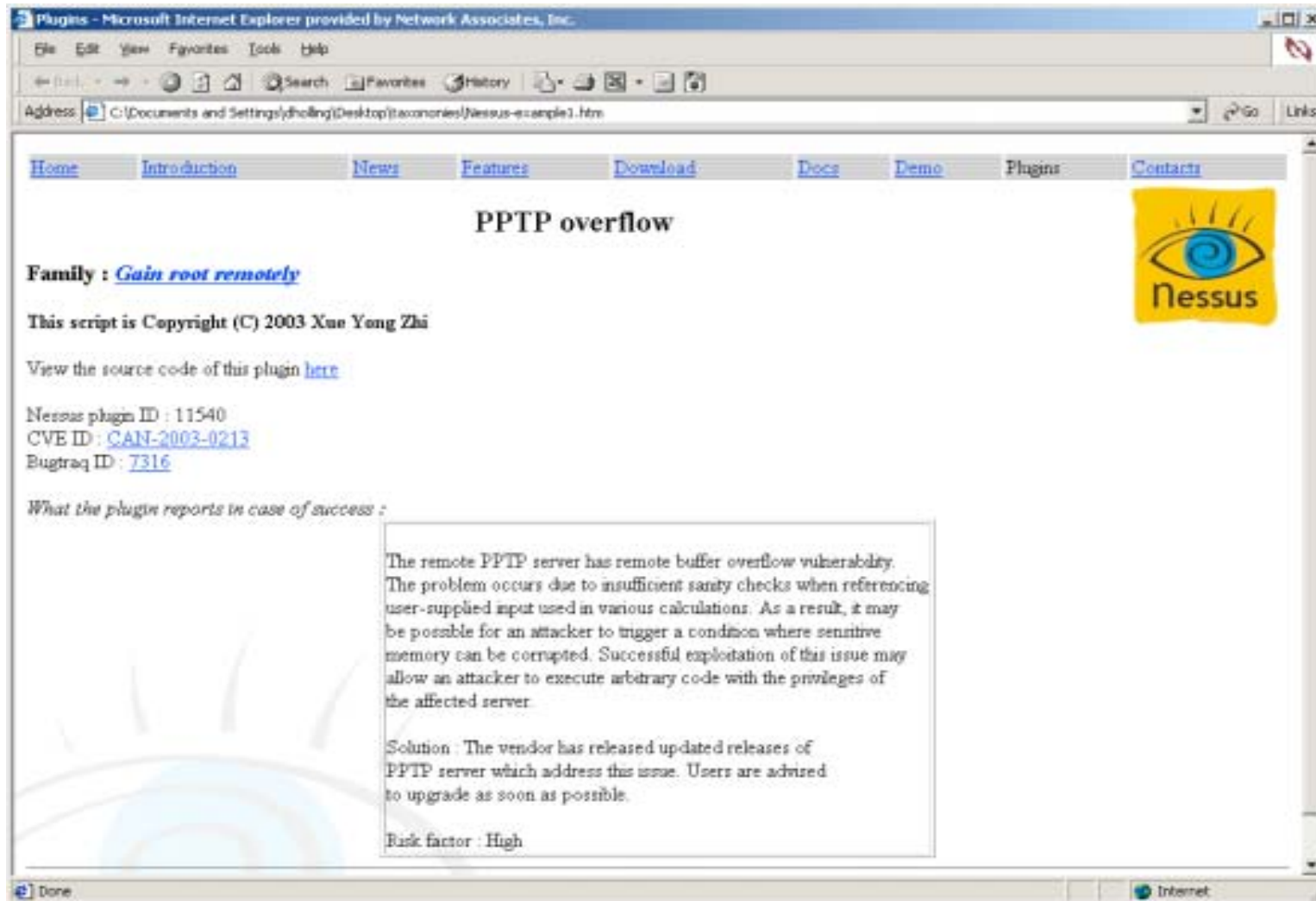
There are 1664 plugins in the database, covering 1057 unique CVE ids and 1133 unique Bugtraq IDs grouped into the following 24 categories:

- Backdoors 45
- CGI Abuses 535
- CISCO 48
- Default UNIX Accounts 27
- Denial of Service 147
- Finger Abuses 10
- Firewalls 24
- FTP 77
- Gain a Shell Remotely 63
- Gain Root Remotely 119
- General 99
- Miscellaneous 82
- Netware 2
- NIS 2
- Port Scanners 5
- Remote File Access 58
- RPC 45
- Settings 8
- SMTP Problems 51
- SNMP 12
- Untested 9
- Useless Services 23
- Windows 149
- Windows: (User Mgmt) 24

Plugins - Microsoft Internet Explorer provided by Network Associates, Inc.

Address: <http://cgi.nessus.org/plugins/dump.php3>

<a href="#">Unpassworded jack account</a>	<a href="#">Default Unix Accounts</a>	Logs into the remote host
<a href="#">Default password (manager) for system</a>	<a href="#">Default Unix Accounts</a>	Logs into the remote host
<a href="#">Unpassworded tutor account</a>	<a href="#">Default Unix Accounts</a>	Logs into the remote host
<a href="#">Default password (wb00tl) for root</a>	<a href="#">Default Unix Accounts</a>	Logs into the remote host
<a href="#">Unpassworded hax0r account</a>	<a href="#">Default Unix Accounts</a>	Logs into the remote host
<a href="#">Unpassworded 4Dgifts account</a>	<a href="#">Default Unix Accounts</a>	Logs into the remote host
<a href="#">Unpassworded root account</a>	<a href="#">Default Unix Accounts</a>	Logs into the remote host
<a href="#">Unpassworded date account</a>	<a href="#">Default Unix Accounts</a>	Logs into the remote host
<a href="#">Default password (root) for root</a>	<a href="#">Default Unix Accounts</a>	Logs into the remote host
<a href="#">Default password (D13hhD) for root</a>	<a href="#">Default Unix Accounts</a>	Logs into the remote host
<a href="#">Teardrop</a>	<a href="#">Denial of Service</a>	Crashes the remote host using the 'teardrop' attack
<a href="#">Crash SMC AP</a>	<a href="#">Denial of Service</a>	Crash SMC Access Point
<a href="#">Bonk</a>	<a href="#">Denial of Service</a>	Crashes the remote host using the 'bonk' attack.
<a href="#">Winuke</a>	<a href="#">Denial of Service</a>	MSG_OOB against the remote host
<a href="#">Notes MTA denial</a>	<a href="#">Denial of Service</a>	Crashes the remote SMTP server
<a href="#">Orange DoS</a>	<a href="#">Denial of Service</a>	Crashes the remote web server
<a href="#">mod_jk chunked encoding DoS</a>	<a href="#">Denial of Service</a>	Checks for version of mod_jk
<a href="#">Too long line</a>	<a href="#">Denial of Service</a>	Crashes a service by sending a too long line



The screenshot shows a Microsoft Internet Explorer browser window with the following content:

- Address bar: C:\Documents and Settings\jholing\Desktop\examples\Nessus-example1.htm
- Navigation menu: Home, Introduction, News, Features, Download, Docs, Demo, Plugins, Contacts
- Section title: PPTP overflow
- Family: [Gain root remotely](#)
- Copyright: This script is Copyright (C) 2003 Xue Yong Zhi
- Source code: View the source code of this plugin [here](#)
- Metadata:
  - Nessus plugin ID : 11540
  - CVE ID : [CAN-2003-0213](#)
  - Bugtraq ID : [7316](#)
- What the plugin reports in case of success :
  - The remote PPTP server has remote buffer overflow vulnerability. The problem occurs due to insufficient sanity checks when referencing user-supplied input used in various calculations. As a result, it may be possible for an attacker to trigger a condition where sensitive memory can be corrupted. Successful exploitation of this issue may allow an attacker to execute arbitrary code with the privileges of the affected server.
  - Solution : The vendor has released updated releases of PPTP server which address this issue. Users are advised to upgrade as soon as possible.
  - Risk factor : High

- Standardize the names for all publicly known vulnerabilities and security exposures.
- Grouped by year (1999 [~1500], 2000 [~1100], 2001 [~500])
- **CVE-1999-0002 Buffer overflow in NFS mountd gives root access to remote attackers, mostly in Linux systems.**
  - **Reference:** SGI:19981006-01-I
  - Reference:** CERT:CA-98.12.mountd
  - Reference:** CIAC:J-006
  - Reference:** BID:121
  - Reference:** XF:linux-mountd-bo





# IDS Example



## ISS/RealSecure

**Oracle connection failed:** (Oracle\_Failed\_Connection)

**About this signature or vulnerability:** This signature detects that an attempt to establish a connection to the Oracle database server has failed.

**Default risk level:** Medium

**Sensors with this signature:** RealSecure OS Sensor: 3.2, RealSecure Server Sensor: 5.5

**Systems affected:** Windows NT

**Type:** Host Sensor

**Vulnerability description:** A connection failed to be established to the Oracle database server.

**How to remove this vulnerability:** A single or sporadic connection failure may be normal, and could be caused by an errant connection attempt. Your audit history can help you determine if a pattern exists, and whether the pattern is an indicator of unauthorized access. Most connection activity is normal for accessing a database. Determine the role of this database and how critical it is in your business activities. This database may have been started for production, testing, troubleshooting, or development functions. The function of the database could help determine who should be connecting, how often, and at what times of the day. Verify that all activities are monitored and retained in an audit history. If there is no legitimate reason for a particular user to be connecting to this database, if database activity originates from unexpected user accounts, or if database accesses occur at unusual times, then this activity may be an indication of misuse.



- ***Blacklce***

- Bionet trojan horse activity**

- Summary: Bionet trojan horse sent data to an intruder.

- Class: Trojan

- Defense: Use anti-virus software to remove trojan horse from system.

- Risk: Intruder may gain control of the system.

- Issue ID: 2001535

- ***TippingPoint***

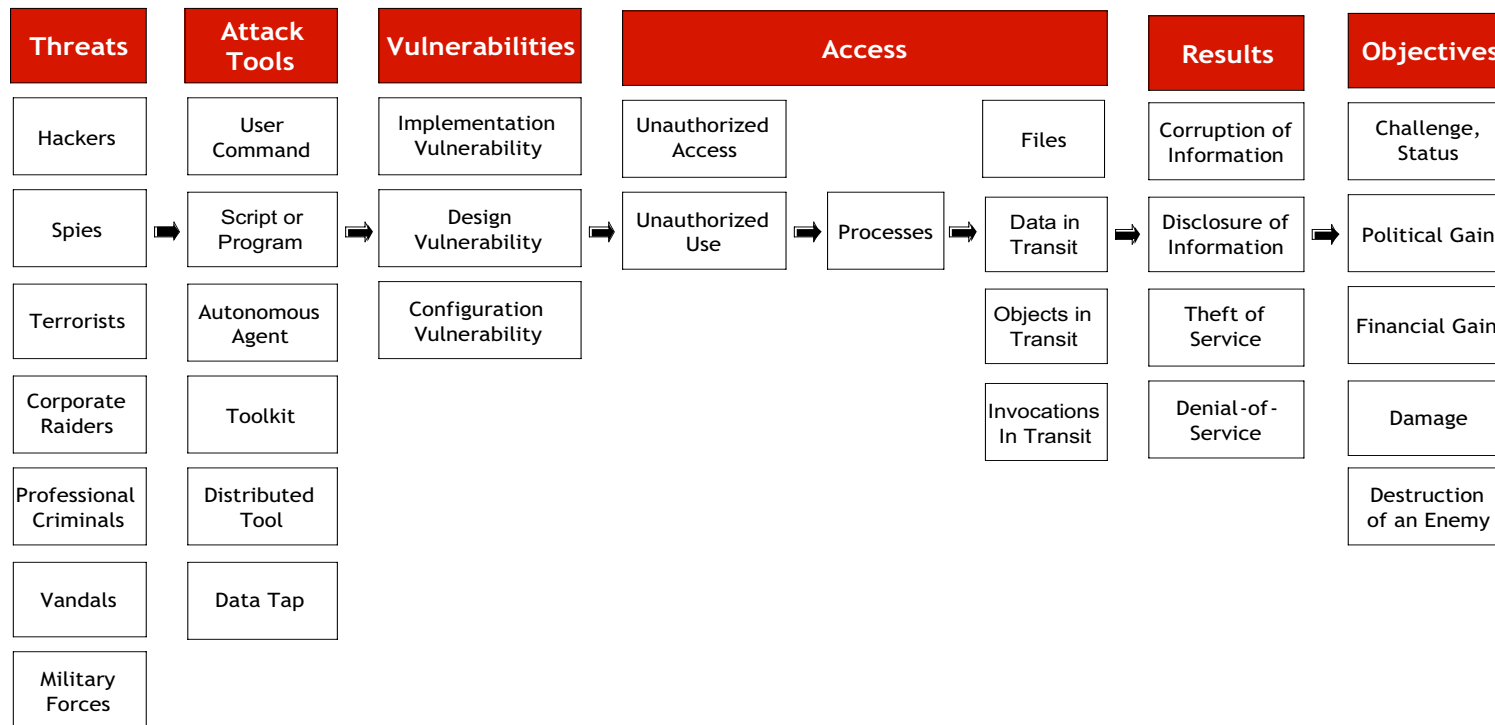
- Name: DNS: TSIG Buffer Overflow exploit 01**

- Id: 592

- Category: ip.udp.dns

- The alarm fires when an attempt to overflow a DNS Transaction Signature is detected. Successful buffer overflow would give a root access to the attacker. BIND, a widely used DNS server program, is susceptible to overflow attack while handling invalid TSIG {Transaction Signature Records}. It is possible to overwrite either the stack {UDP connection) or heap {TCP connection) and execute arbitrary code with the privileges of Bind daemon {usually root} . The Lion worm uses TSIG overflow attack to propagate itself.

## *Abstract and insufficiently detailed*



Adapted from *An Analysis Of Security Incidents On The Internet, 1989 - 1995*, Figure 6.9., Complete Computer and Network Attack Taxonomy, Copyright John D. Howard, 1997, all rights reserved, a doctoral thesis submitted to Carnegie Mellon University, as found at <http://www.cert.org/research/JHThesis/Start.html>

1. Vendor and public security databases are generally initialized and updated according to individual analyst intuition, knowledge and understanding. They lack uniformity, common vocabulary, semantic consistency, and organization. They tend to be internally inconsistent as well as inconsistent with one another. They also are in natural language.
2. Reconciliation of these inconsistencies during production of IAM taxonomies will require systematic incident analysis and description and may require machine-based reasoning as well as natural language understanding tools tuned to the security domain because of content quality, variability of reporting formats, reliance upon natural language description, and existing database organizational deficiencies.

3. IAM threat, attack, and vulnerability taxonomy categories must be more closely related to detection and observation mechanisms.
  - Existing taxonomies are instructional, informative, descriptive, procedural, capture problem terminology
  - Do not conform well with measurement or estimation opportunities that may exist with existing security tools
  - Based upon conceptual attack classification rather than detection-based classification
  - Group conceptually similar attacks and events into single categories regardless of detection methodology. For example, they ignore the fact that different “buffer overflow” events may be detected and potentially measured at different interfaces via separate, unrelated mechanisms.

4. Detection-based classification can provide opportunities for measurement or estimation that are unavailable to concept-based classification because of the latter's association of like events observed through unlike mechanisms.
5. Existing taxonomies are not amenable to representation of important event relationships or interdependencies. Have overly simplistic structure that do not capture temporal, causal, or evolutionary relationships among threats, attacks, and vulnerabilities. These are useful relationships for IAM taxonomy structure.

6. IAM taxonomies may be generated in significant part from the analysis of detection/response mechanism knowledge bases such as Host Intrusion Detection, Network Intrusion Detection, and BSM (basic security module) knowledge bases.
7. The content of IAM taxonomy databases may be populated in large part from the content of major incident and vulnerability databases such as CERT, NESSUS, and CVE, extended to include uncovered attacks and vulnerabilities because these are comprehensive and will likely be maintained.

**“A threat is an adversary motivated and capable of exploiting a vulnerability.”  
[Schneider, *Trust in Cyberspace*, 1998]**

- ✓ *Threat Characteristics* -- Different types of security threats/attackers exist from the perspective of IAM, both animate and inanimate.
- ✓ *Accessibility/Opportunity* – There may be specific periods when access controls are reduced or curtailed (such as preventative maintenance) or circumstances (such as insider threats) where an attacker has greater access to targets that otherwise.
- ✓ *Asset Financial Value* -- Helps establish attractiveness measures
- ✓ *Asset Security Value* – The exposure of other assets resulting from compromise of this asset.
- ✓ *Attacker Sophistication/knowledge* – e.g., script-kiddie following a canned attack vs sophisticated attacker with detailed knowledge in a particular area such as HTTP protocol.
- ✓ *Coordination/synchronization* – an attacker may be comprised of multiple elements such as zombie processes associated with a DDoS attack that have to work in concert to be effective.
- ✓ *Knowledge* - General or specific knowledge of target system properties, public knowledge of system characteristics, code exposure/publication, and vulnerability publication may provide attackers with different probability of success.
- ✓ *Progression* - Effectiveness depends upon specific target system properties (e.g., previously installed backdoors) that may be a prerequisite to successful attack execution.

**“An attack is the means [sequence of actions] of exploiting a vulnerability.”  
[Schneider, *Trust in Cyberspace*, 1998]**

- ✓ *Composition* - There are different conceptual levels of attack abstraction employed in existing databases.
- ✓ *Countermeasures* - Knowledge of a vulnerability and its detailed characteristics can suggest particular countermeasures that may be applied at an individual level or on a more global scale.
- ✓ *Temporal constraints* - A particular attack may only be possible during a particular time of day or day of the week, or other instance when specific countermeasures are relaxed.
- ✓ *Preconditions* – Enabling conditions that must be satisfied (e.g., available bandwidth)
- ✓ *References* – Links to external attack detail reports such as public databases
- ✓ *Cumulative effects* – An individual instance of an attack component may not be destructive (e.g., SYN) while a significant quantity of such events may constitute a serious attack (e.g., SYN flood)
- ✓ *Attack target*: host, operating system, service, application, file, user
- ✓ *Type of attack* - virus, buffer overflow, DoS, timing, password, desynchronization, resource exhaustion
- ✓ *Vehicle*: API, email, protocol
- ✓ *Medium*: wired, wireless
- ✓ *Composition*: different components of the total attack.
- ✓ *Breadth/scope*: host, LAN, WAN, Internet
- ✓ *Bandwidth*: communication resource consumption associated with or a manifestation of attacks.
- ✓ *Interdependency*: multiple components working together or progressively to achieve an attack goal.



**“A vulnerability is an error or weakness in design, implementation, or operation.” [Schneider, *Trust in Cyberspace*, 1998]**

- ✓ *Vulnerability Presence* - A given target platform or resource may or may not be vulnerable to a specific attack.
- ✓ *Protection Barriers* - Components along an attack path that may protect or isolate assets from particular types of attacks.
- ✓ *Temporal Constraints* - Periods of specific asset exposure.
- ✓ *Complexity* - Exploitation may depend upon related vulnerabilities for success.
- ✓ *Context* - May require contextual knowledge for successful exploitation, e.g., buffer overflow and knowledge of stack structure in order to successfully capture of control.
- ✓ *Temporal* - A given vulnerability might only materialize during specific times of the day or in concert with other events.
- ✓ *Social engineering* -- Necessary to convince the target of an attack to assist in attack execution through social engineering.
- ✓ *Access* - May require sufficient access time or particular type, level, or object access for execution
- ✓ *Exposure*: (temporal, breadth, depth)

- ❖ *A systematic method to characterize system security based on varying attacks. Each attack tree enumerates and elaborates the ways that an attacker could cause the event to occur. Each path through an attack tree represents a unique attack on the enterprise.*
- “Attack Trees: Modeling Security Threats,” *Dr. Dobb’s Journal*, December 1999. (Schneier)
- “Attack Modeling for Information Security and Survivability”, CMU/SEI-2001-TN-001 (Moore, Ellison, Linger)
- draft-convery-bgpattack-00 (Convery, Cook, Franz)  
(IETF draft presenting all known attack vectors into or using BGP, presented in "Attack Tree" format.)



# Partial BGP Attack Tree

(draft-convery-bgpattack-00)



## Originate Unauthorized Prefix/Attribute into Peer Route Table

Attack:

OR 1. Send from valid Router

OR 1. Misconfigured

2. Compromise router (Appendix A.1)

2. Send from Invalid Router

AND 1. Gag valid router

OR 1. Kill Router

OR 1. Power Off/Physical Layer

2. Crash and prevent reboot

3. Conduct denial of service against router

2. Steal IP Addr

OR 1. ARP Spoof

2. Steal MAC

2. Introduce rogue router (Assume IP)

OR 1. Steal IP Addr (section 2.1.3.1.2)

2. IGP More Specific Route Introduction

3. Establish unauthorized BGP session w/peer

3. Send spoofed BGP Update from Non-Router

OR 1. Conduct TCP Sequence Number Attack (Appendix A.4)

2. Conduct Man-in-the-Middle (Appendix A.3)

AND 4. Craft BGP Message

➤ **Concern: Majority still not quantifiable or even estimable!**



## ➤ Locate difficult to detect errors in large systems

- Claim: many abstract properties map directly to concrete code actions making it possible to find rule violations by checking the actions for errors.
  - Claim: many system restrictions can be automatically checked via light-weight compiler extensions
  - Employ system\_specific static analysis to locate security errors that violate general or specific implementation rules
    - data/pointer sanitization before use
    - event ordering requirements
    - idempotent operations
    - dead code
    - required or proscribed actions
    - legality of actions in a given context
    - assignments that were never read
    - unexercised conditional branches
  - Locate potentially dangerous code that strongly correlates with the presence of traditional hard errors including null pointer de-references and unreleased locks.
  - Automatically derive implicit program correctness rules directly from code (*no explicit statement of the underlying rule or programmer belief*).
  - Cross\_check rules against code base for coding contradictions that reflect software errors.
- ❖ *Approach tested against 4 complex systems: Linux, OpenBSD, the XoK exokernel, and the FLASH machine's embedded software - over one-hundred security errors in Linux and OpenBSD were uncovered.*

- **Locate more subtle programming errors that tend to show up after lengthy periods of program execution after atypical sequences of events occur**
  - Traditional model checking takes an abstract, simplified code description exhaustively tests it under all inputs
    - difficult and error prone,
    - hard to build
    - can take significantly more time to produce than the code being modeled
    - checking an abstraction of the code makes it easy to miss real implementation errors
  - Stanford automated model checking techniques - 2 major components:
    - an extensible compiler system - facilitates model checking for large scale systems
    - new form of model checker that works directly on C/C++ implementations
  - Automatically extracts model description from code description - combines it with a model of the hardware, a description of correctness, and an initial state and then checks the collection with a model checker
- ❖ Uncovered 8 errors in prior-exercised cache coherence protocols of the Stanford FLASH multiprocessor
- ❖ Applied to 3 implementations of the Ad-hoc On-demand Distance Vector networking protocol - 34 distinct errors + a bug in the AODV specification

- **Deduce global vulnerabilities by analyzing the effects of interactions of local vulnerabilities as well as the effects of the interconnection between hosts**
  - Automation of Red-team analysis (human-intensive, error prone, and non-scalable)
  - Depict ways in which a network can be forced into an unsafe state.
  - Each path represents a series of exploits which lead to a violation of a desirable security property, e.g., protection against unauthorized administrative access.
  - Model checking technology automatically generates the graphs (counter-examples indicate where violation of a desirable safety property can be induced) checking whether a formal model of the system satisfies a given property.
  - 4-step process:
    1. Specifying desired security property - violation corresponds to attacker goal
    2. Modeling the network (finite state machine - state transitions are to potential attacks)
    3. Producing attack graph from the model
    4. Analysis of the attack graph
  - Minimization: determine the minimum number of atomic attacks that must be prevented to guarantee that a particular attacker goal cannot be achieved.

- Take manageable bites out of the problem space
- No prior understanding as to how they fit
- No grand encompassing research plan
  
- ❖ Jeannette Wing re attack graphs: *“We’re all of us now, in the business of system security, searching for helpful solutions. They are not going to be the silver bullet; they are not going to be the one thing, but we have to provide some help...What we doing here is just systematizing things that are being done by hand today.”* [Jeannette M. Wing, CSE Colloquia 2003, Research Channel Programming]

## Foundation building:

- ❖ Refine/formalize the content of preeminent attack collection and vulnerability databases, the CERT and NISSUS databases, and CVE lexicon -- internal consistency, standardized repositories of security knowledge and example data that are uniform in their coverage.
- ❖ Additional research into error description & classification
- ❖ Security-domain-specific natural language processing tools (automated) that take natural language descriptions of attacks and vulnerabilities and reduces them into a standardized format as well as captures and encodes additional data relationships specified by the analyst



## Other Research Opportunities:

- ❖ Attack trees
  - Independent validation
  - Refinement of BGP attack tree (specificity)
  - Check convergence toward a single “correct” attack tree
- ❖ Automated vulnerability pattern detection (meta-compilation)
  - Independently validate formulation of static error detection patterns (old and new)
- ❖ Automated model checking
  - Continue and validate work re automatically generated models. Extend to different languages (JAVA?)
- ❖ Attack graphs
  - Scalability to larger collections of hosts and complex vulnerability data