Vulnerability Scanner Tools

ACIS Professional Center

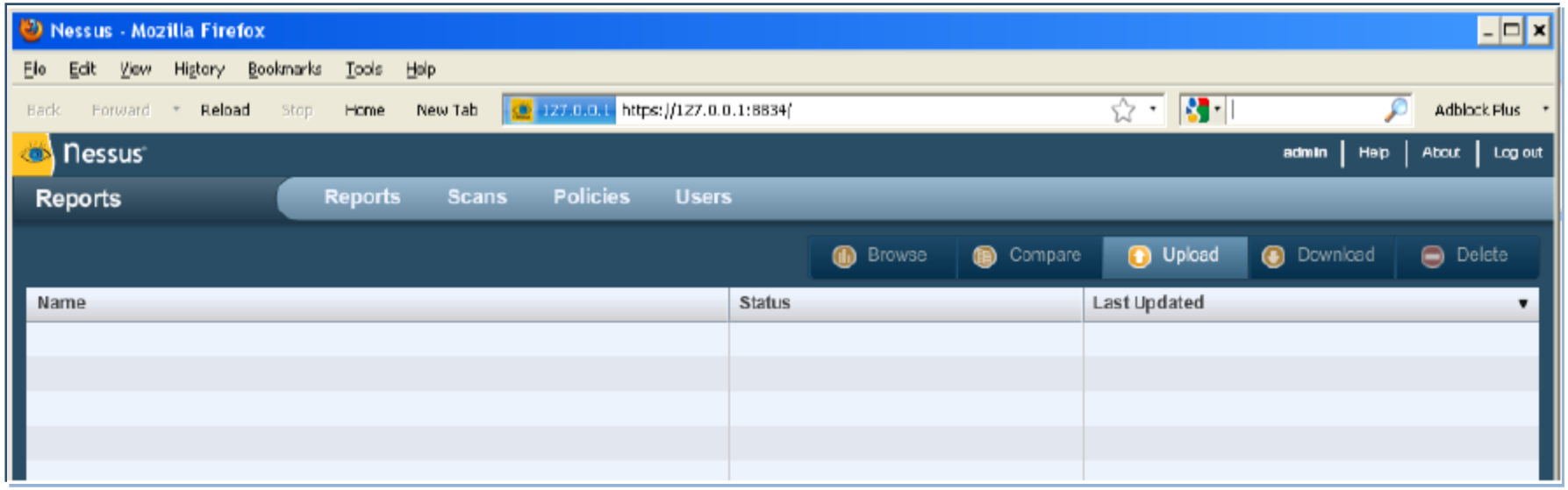
Difference between Penetration Testing and Vulnerability Assessment?

- Vulnerability Assessment:
  - Typically is general in scope and includes a large assessment.
  - Predictable. (I know when those darn Security guys scan us.)
  - Unreliable at times and high rate of false positives. (I've got a banner)
  - Vulnerability assessment invites debate among System Admins.
  - Produces a report with mitigation guidelines and action items.

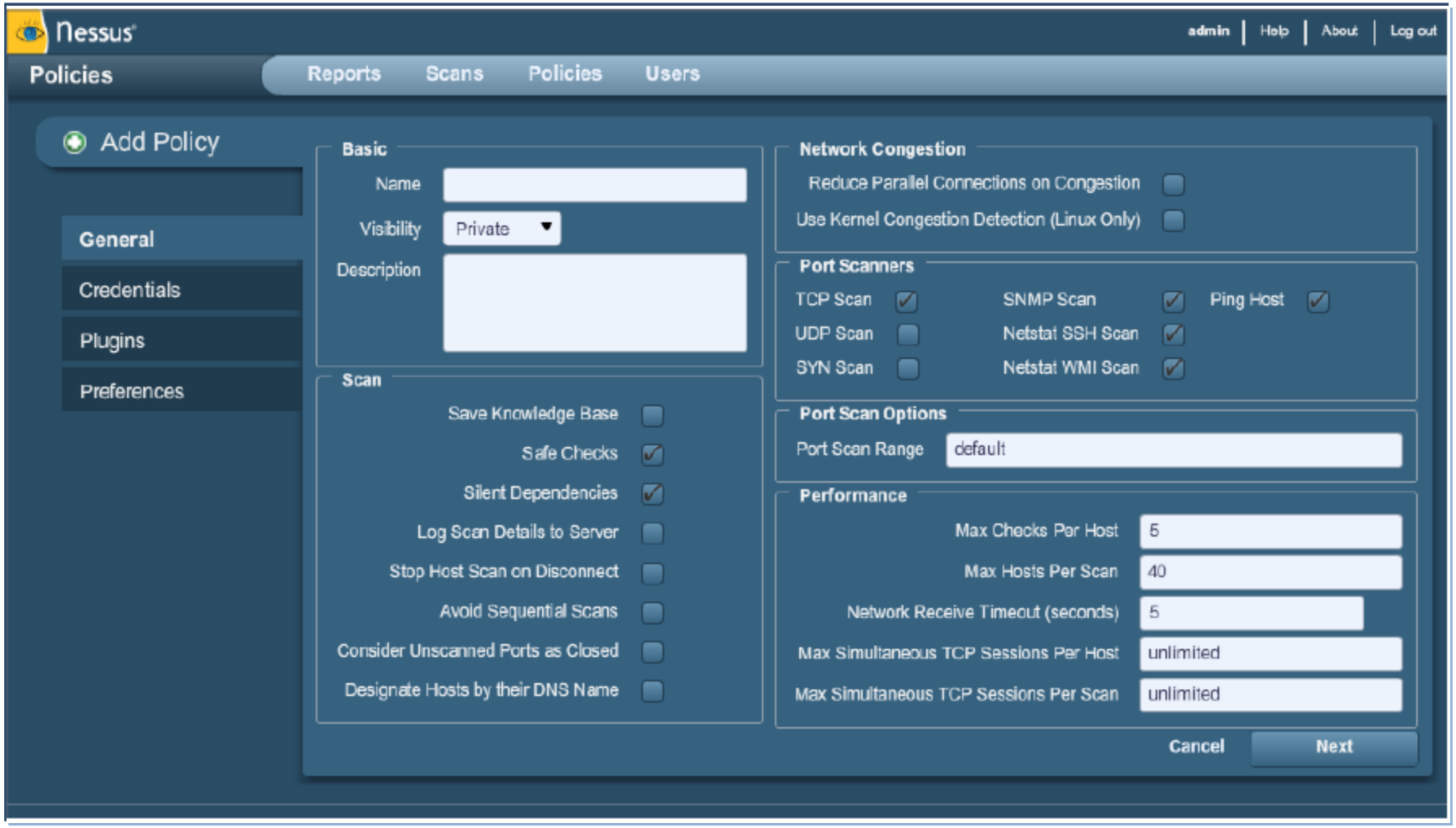
- Penetration Testing:
  - Focused in scope and may include targeted attempts to exploit specific vectors (Both IT and Physical)
  - Unpredictable by the recipient. (Don't know the "how?" and "when?")
  - Highly accurate and reliable. (I've got root!)
  - Penetration Testing = Proof of Concept against vulnerabilities.
  - Produces a binary result: Either the team owned you, or they didn't.

Vulnerability Assessment

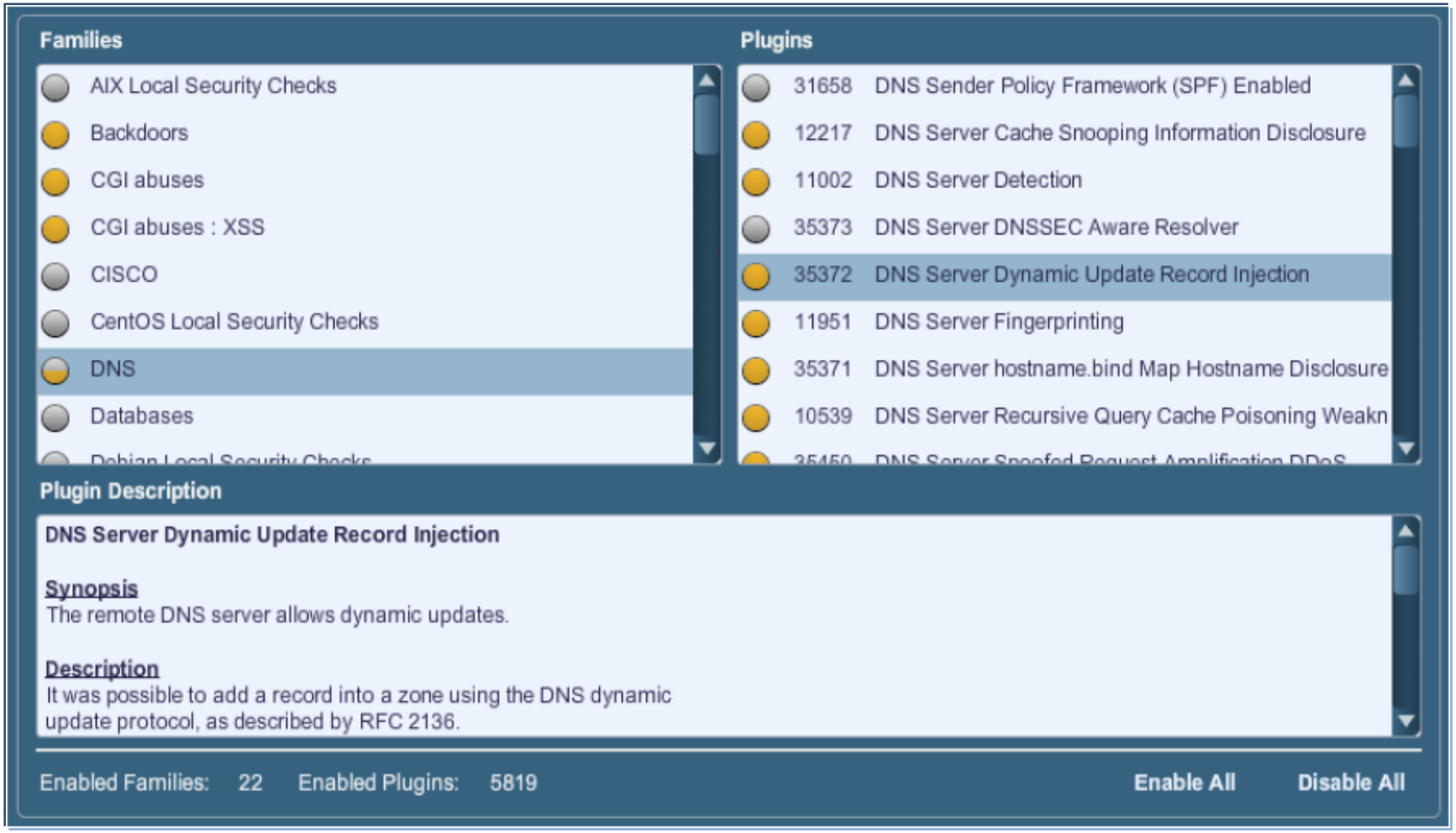
Nessus


Nessus - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Back Forward Reload Stop Home New Tab
https://127.0.0.1:8834/
Adblock Plus
Nessus
admin | Help | About | Log out
Reports
Reports Scans Policies Users
Browse Compare Upload Download Delete
Name
Status
Last Updated

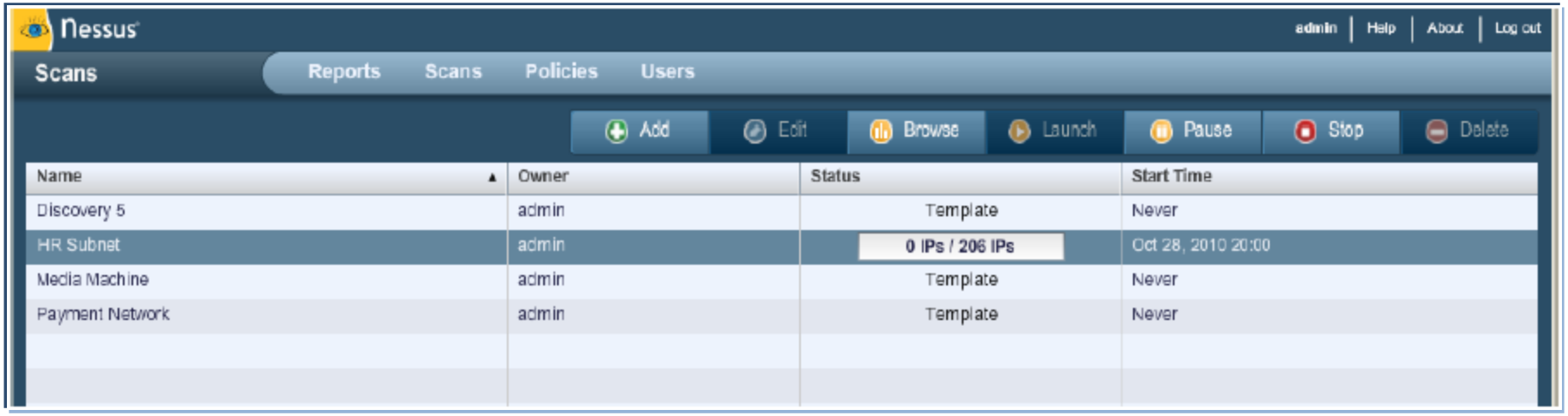
Nessus


Nessus
admin | Help | About | Log out
Policies
Reports
Scans
Policies
Users
Add Policy
General
Credentials
Plugins
Preferences
Basic
Name
Visibility
Private
Description
Scan
Save Knowledge Base
Safe Checks
Silent Dependencies
Log Scan Details to Server
Stop Host Scan on Disconnect
Avoid Sequential Scans
Consider Unscanned Ports as Closed
Designate Hosts by their DNS Name
Network Congestion
Reduce Parallel Connections on Congestion
Use Kernel Congestion Detection (Linux Only)
Port Scanners
TCP Scan
SNMP Scan
Ping Host
UDP Scan
Netstat SSH Scan
SYN Scan
Netstat WMI Scan
Port Scan Options
Port Scan Range
default
Performance
Max Checks Per Host
5
Max Hosts Per Scan
40
Network Receive Timeout (seconds)
5
Max Simultaneous TCP Sessions Per Host
unlimited
Max Simultaneous TCP Sessions Per Scan
unlimited
Cancel
Next

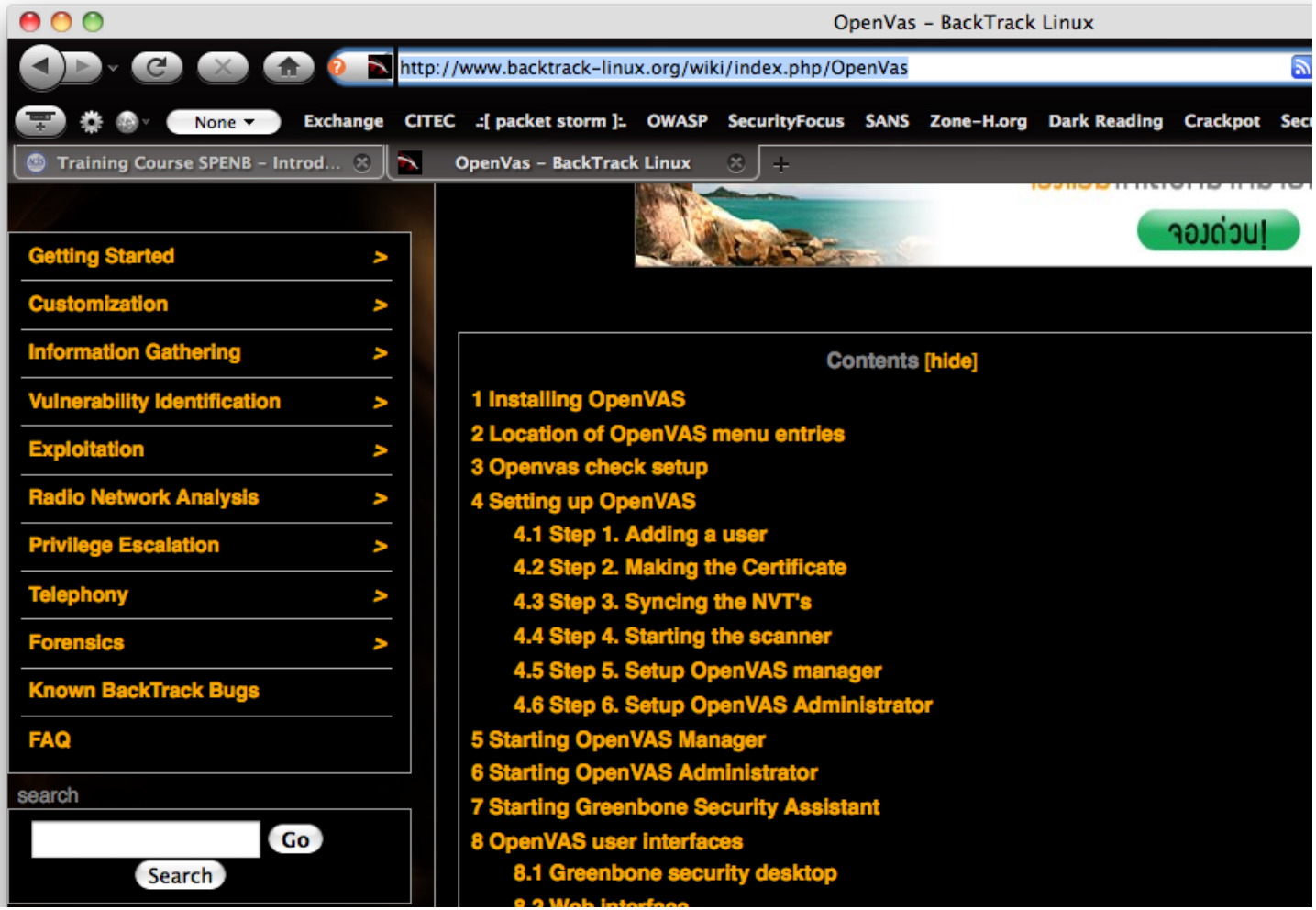
Nessus


Families
AIX Local Security Checks
Backdoors
CGI abuses
CGI abuses : XSS
CISCO
CentOS Local Security Checks
DNS
Databases
Plugins
31658 DNS Sender Policy Framework (SPF) Enabled
12217 DNS Server Cache Snooping Information Disclosure
11002 DNS Server Detection
35373 DNS Server DNSSEC Aware Resolver
35372 DNS Server Dynamic Update Record Injection
11951 DNS Server Fingerprinting
35371 DNS Server hostname.bind Map Hostname Disclosure
10539 DNS Server Recursive Query Cache Poisoning Weakn
Plugin Description
DNS Server Dynamic Update Record Injection
Synopsis
The remote DNS server allows dynamic updates.
Description
It was possible to add a record into a zone using the DNS dynamic update protocol, as described by RFC 2136.
Enabled Families: 22 Enabled Plugins: 5819
Enable All
Disable All

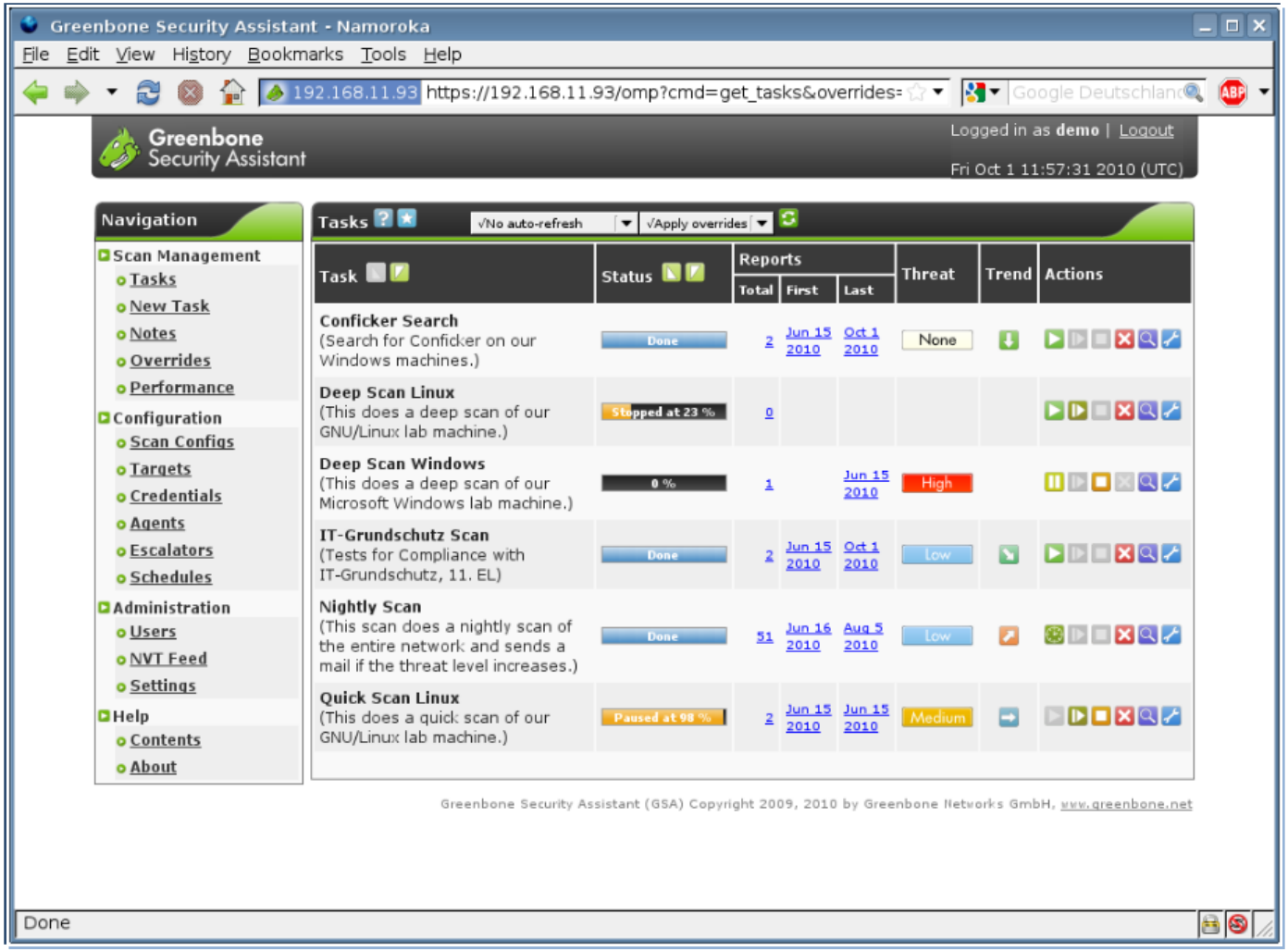
Nessus


Nessus
admin | Help | About | Log out
Scans
Reports
Scans
Policies
Users
Add
Edit
Browse
Launch
Pause
Stop
Delete
Name
Owner
Status
Start Time
Discovery 5
admin
Template
Never
HR Subnet
admin
0 IPs / 206 IPs
Oct 28, 2010 20:00
Media Machine
admin
Template
Never
Payment Network
admin
Template
Never

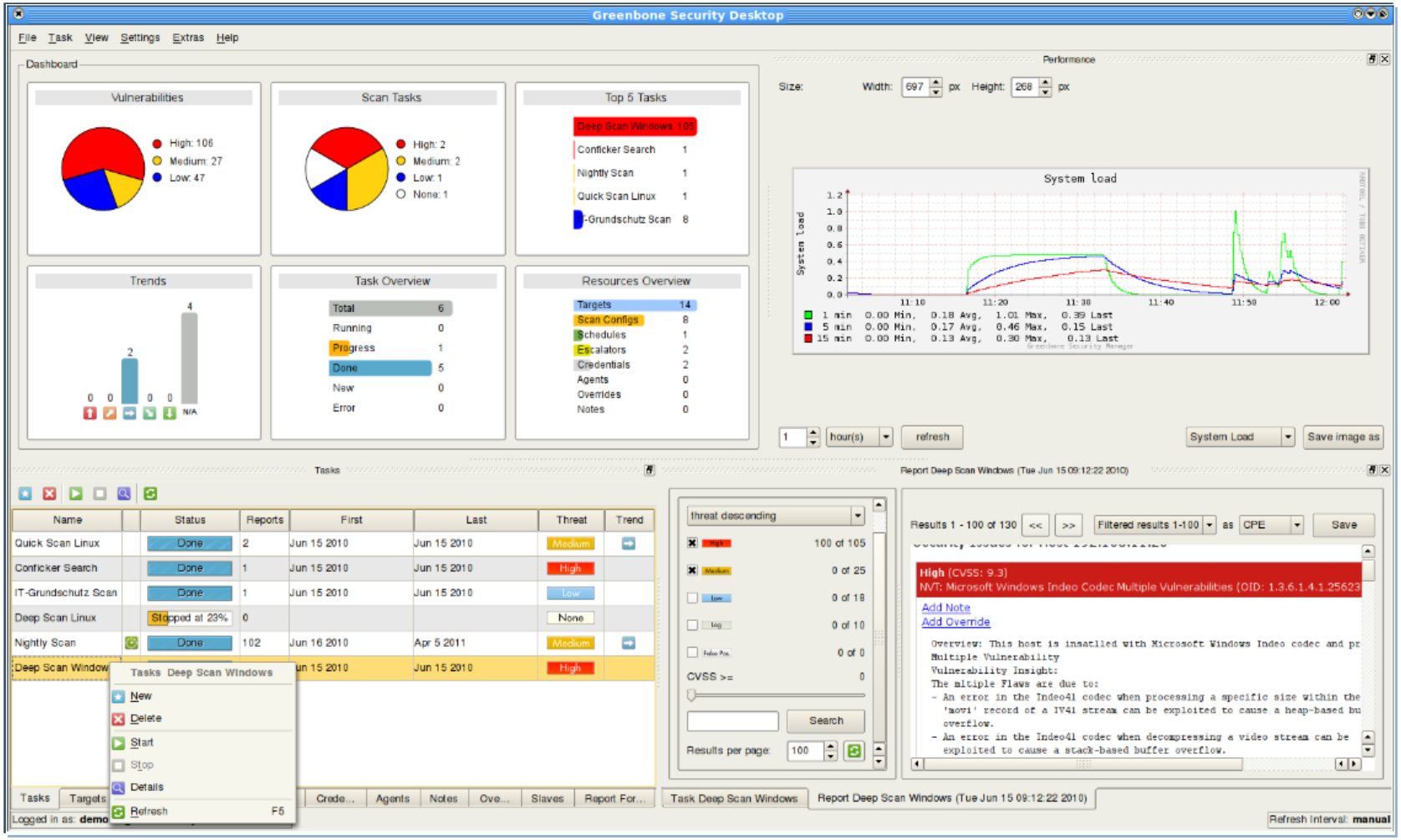
OpenVas: http://www.backtrack-linux.org/wiki/index.php/OpenVas



OpenVas


Greenbone Security Assistant - Namoroka
File Edit View History Bookmarks Tools Help
192.168.11.93 https://192.168.11.93/omp?cmd=get_tasks&overrides=
Greenbone Security Assistant
Logged in as demo | Logout
Fri Oct 1 11:57:31 2010 (UTC)
Navigation
Scan Management
Tasks
New Task
Notes
Overrides
Performance
Configuration
Scan Configs
Targets
Credentials
Agents
Escalators
Schedules
Administration
Users
NVT Feed
Settings
Help
Contents
About
Tasks
√No auto-refresh
√Apply overrides
Task
Status
Reports
Total
First
Last
Threat
Trend
Actions
Conficker Search
(Search for Conficker on our Windows machines.)
Done
2
Jun 15 2010
Oct 1 2010
None
Deep Scan Linux
(This does a deep scan of our GNU/Linux lab machine.)
Stopped at 23 %
0
Deep Scan Windows
(This does a deep scan of our Microsoft Windows lab machine.)
0 %
1
Jun 15 2010
High
IT-Grundschutz Scan
(Tests for Compliance with IT-Grundschutz, 11. EL)
Done
2
Jun 15 2010
Oct 1 2010
Low
Nightly Scan
(This scan does a nightly scan of the entire network and sends a mail if the threat level increases.)
Done
51
Jun 16 2010
Aug 5 2010
Low
Quick Scan Linux
(This does a quick scan of our GNU/Linux lab machine.)
Paused at 98 %
2
Jun 15 2010
Jun 15 2010
Medium
Greenbone Security Assistant (GSA) Copyright 2009, 2010 by Greenbone Networks GmbH, www.greenbone.net
Done

OpenVas


Greenbone Security Desktop
File Task View Settings Extras Help
Dashboard
Vulnerabilities
High: 106
Medium: 27
Low: 47
Scan Tasks
High: 2
Medium: 2
Low: 1
None: 1
Top 5 Tasks
Deep Scan Windows 105
Conficker Search 1
Nightly Scan 1
Quick Scan Linux 1
-Grundschutz Scan 8
Trends
Task Overview
Total 6
Running 0
Progress 1
Done 5
New 0
Error 0
Resources Overview
Targets 14
Scan Configs 8
Schedules 1
Escalators 2
Credentials 2
Agents 0
Overrides 0
Notes 0
Performance
Size: Width: 697 px Height: 268 px
System load
1 min 0.00 Min, 0.18 Avg, 1.01 Max, 0.39 Last
5 min 0.00 Min, 0.17 Avg, 0.46 Max, 0.15 Last
15 min 0.00 Min, 0.13 Avg, 0.30 Max, 0.13 Last
refresh
System Load
Save image as
Tasks
Name Status Reports First Last Threat Trend
Quick Scan Linux Done 2 Jun 15 2010 Jun 15 2010 Medium
Conficker Search Done 1 Jun 15 2010 Jun 15 2010 High
IT-Grundschutz Scan Done 1 Jun 15 2010 Jun 15 2010 Low
Deep Scan Linux Stopped at 23% 0 None
Nightly Scan Done 102 Jun 16 2010 Apr 5 2011 Medium
Deep Scan Window Jun 15 2010 Jun 15 2010 High
Tasks Deep Scan Windows
New
Delete
Start
Stop
Details
Refresh F5
Report Deep Scan Windows (Tue Jun 15 09:12:22 2010)
threat descending
100 of 105
0 of 25
0 of 18
0 of 10
0 of 0
CVSS >= 0
Search
Results per page: 100
Results 1 - 100 of 130
Filtered results 1-100
as CPE
Save
High (CVSS: 9.3)
NVT: Microsoft Windows Indeo Codec Multiple Vulnerabilities (OID: 1.3.6.1.4.1.25623
Add Note
Add Override
Overview: This host is installed with Microsoft Windows Indeo codec and pr
Multiple Vulnerability
Vulnerability Insight:
The multiple Flaws are due to:
- An error in the Indeo41 codec when processing a specific size within the
'movi' record of a IV41 stream can be exploited to cause a heap-based bu
overflow.
- An error in the Indeo41 codec when decompressing a video stream can be
exploited to cause a stack-based buffer overflow.
Tasks Targets Agents Notes Slaves
Task Deep Scan Windows
Report Deep Scan Windows (Tue Jun 15 09:12:22 2010)
Logged in as: demo
Refresh Interval: manual

Enumeration

ACIS Professional Center

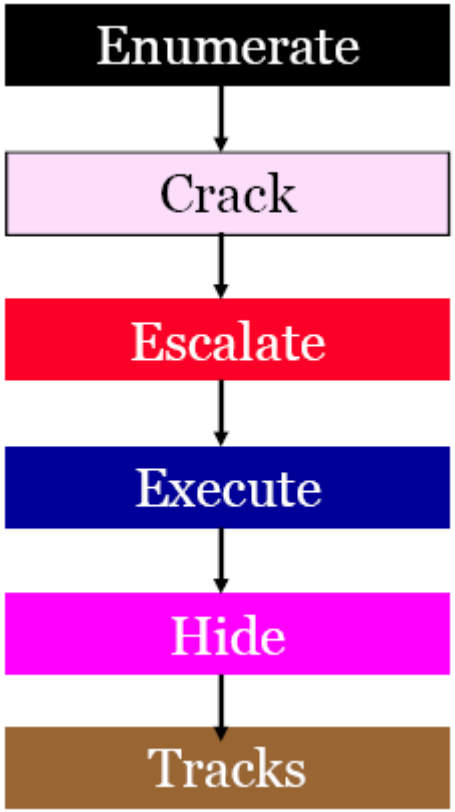
Objective

- Overview of system Hacking Cycle
- Enumeration
- Techniques for Enumeration
- Establishing Null Session
- Enumerating User Accounts
- Null user Countermeasures
- SNMP Scan

Objective (cont'd)

- MIB
- SNMP Util Example
- SNMP Enumeration Countermeasures
- Active Directory Enumeration
- AD Enumeration Countermeasures

Overview of System Hacking Cycle



What is Enumeration

- Enumeration is defined as extraction of user names, machine names, network resources, shares, and services
- Enumeration techniques are conducted in an intranet environment
- Enumeration involves active connections to systems and directed queries
- The type of information enumerated by
  - Network resources and shares
  - Users and groups
  - Applications and banners
  - Auditing settings

Netbios Null Sessions

- The null session is often refereed to as the Holy Grail of Windows hacking. Null sessions take advantage of flaws in the CIFS/SMB (Common Internet File System/Server Messaging Block)
- You can establish a null session with a Windows (NT/200/XP) host by logging on with a null user name and password
- Using these null connections, you can gather the following information from the host:
  - List of users and groups
  - List of machines
  - List of shares
  - Users and host SIDs (Security Identifiers)

So What's the Big Deal

Anyone with a NetBIOS connection to your computer can easily get a full dump of all your user names, groups, shares, permissions, policies, services, and more using the null user

The following syntax connects to the hidden Inter Process Communication 'share' (IPC$) at IP address 192.34.34.2 with the built-in anonymous user (/u:"") with a ("") null password

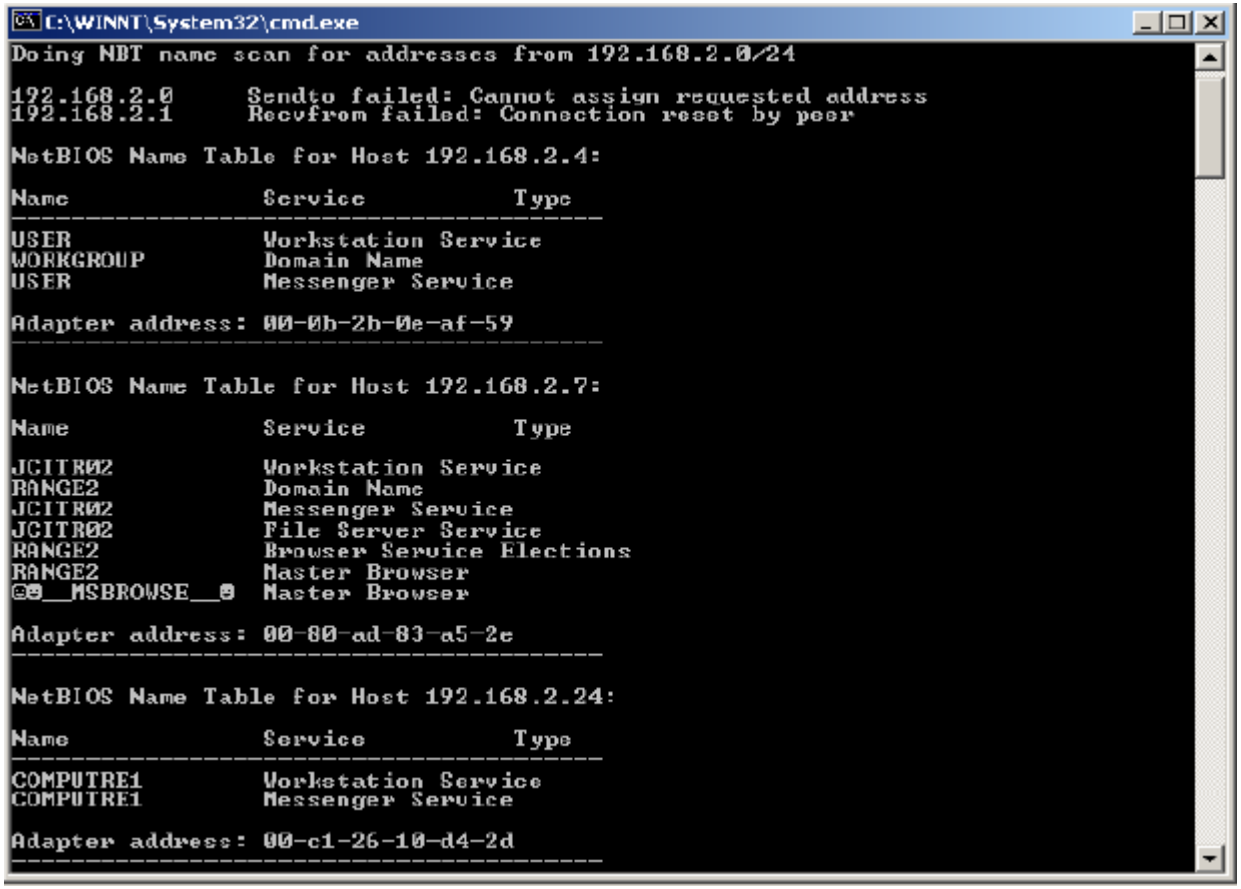
The attacker now has a channel over which to attempt various techniques

The CIFS/SMB and NetBIOS standards in Windows 2000 include APIs that return rich information about a machine via TCP port 139—even to the unauthenticated users

This works on Windows 2000/XP systems, but not on Win 2003

```
Windows: C:\>net use \\192.34.34.2\IPC$ "" /u:""
Linux: $ smbclient \\\\target\\ipc\$ ""  -U ""
```

NetBIOS Enumeration Using Netview (cont'd)



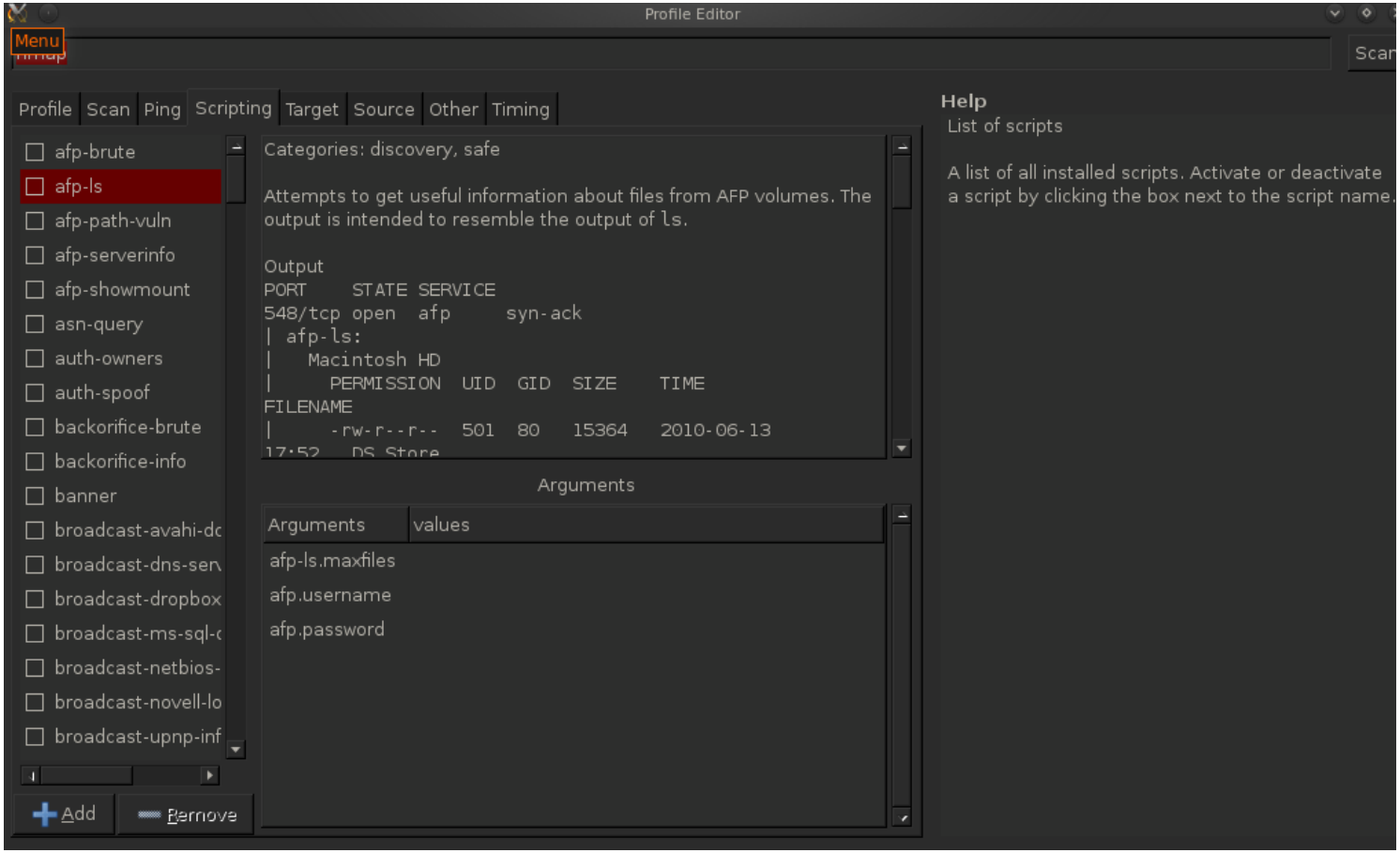
Nbtstat Enumeration Tool

- Nbtstat is a Windows command-line tool that can be used to display information about a computer's NetBIOS connections and name tables
  - Run: nbtstat –A <some ip address>
- C:\nbtstat
  - Displays protocol statistics and current TCP/IP connections using NBT(NetBIOS over TCP/IP).
  - NBTSTAT [-a RemoteName] [-A IP address] [-c] [-n] [-r] [-R] [-s] [S][interval]]
-

Null Session Countermeasures

- Null sessions require access to TCP 139 and/or TCP 445 ports
- Null sessions do not work with Windows 2003
- You could also disable SMB services entirely on individual hosts by unbinding the WINS Client TCP/IP from the interface
- Edit the registry to restrict the anonymous user:
- Step1: Open regedt32 and navigate to
  - `HKLM\SYSTEM\CurrentControlSet\LSA`
- Step2: Choose edit | add value
  - value name: Restrict Anonymous
  - Data Type: `REG_WORD`
  - Value: 2

Nmap Script (NSE)



Zenmap (NSE Mode)

```
nmap -sS -A -Pn --script banner,http-enum,http-headers,http-vmware-path-vuln,ms-sql-brute,ms-sql-config,ms-s...
Network Distance: 1 hop
Service Info: OS: Windows

Host script results:
| ms-sql-info:
|   Windows server name: SERVER
|   [192.168.1.111\MSSQLSERVER]
|     Instance name: MSSQLSERVER
|     Version: Microsoft SQL Server 2000 RTM
|       Version number: 8.00.194.00
|       Product: Microsoft SQL Server 2000
|       Service pack level: RTM
|       Post-SP patches applied: No
|     TCP port: 1433
|     Named pipe: \\192.168.1.111\pipe\sql\query
|_    Clustered: No
| smb-brute:
|   backup:pukcab => Login was successful
|   epp:password => Login was successful
|   guest:guest => Login was successful
|   john:money => Login was successful
|_  molly:money => Login was successful
| smb-enum-users:
|_  Domain: SERVER; Users: Administrator, backup, epp, epp_contractor, Guest, IUSR_SERVER,
IWAM_SERVER, Jim, John, mary, molly, prathan, TsInternetUser
| smb-check-vulns:
|_  MS08-067: VULNERABLE
```

System Hacking

ACIS Professional Center

CRACKING PASSWORD

Types of Password Attacks

- Passive online attacks
- Active online attacks
- Offline attacks
- Non-electronic attacks

Passive Online Attack: Wire Sniffing

- Access and record the raw network traffic
- Wait until the authentication sequence
- Brute force credentials
- Considerations:
  - Relatively hard to perpetrate
  - Usually computationally complex
  - Tools widely available

Passive Online Attack:Man-in-the- Middle and Replay Attacks

- Somehow get access to the communications channel
- Wait until the authentication sequence
- Proxy authentication-traffic
- No need to brute force

Active Online Attack: Password Guessing

- Try different passwords until one works
- Succeeds with:
  - Bad passwords
  - Open authentication points
- Considerations:
  - Takes a long time
  - Requires huge amounts of network bandwidth
  - Easily detected
  - Core problem: bad passwords

Offline Attacks

- Offline attacks are time consuming
- LM Hashes are much more vulnerable due to smaller key space and shorter length
- Web services are available
- Distributed password cracking techniques are available
- Mitigations:
  - Use good passwords
  - Remove LM Hashes
  - Attacker has password database

Offline Attacks (cont'd)

Dictionary Attack

Try different passwords from a list

Succeeds only with poor passwords

Considerations:

- Very fast
- Core problem: bad passwords

Hybrid Attack

Start with the dictionary

Insert entropy:

- Append a symbol
- Append a number

Considerations:

- Relatively fast
- Succeeds when entropy is poorly used

Offline Attack: Brute-force Attack

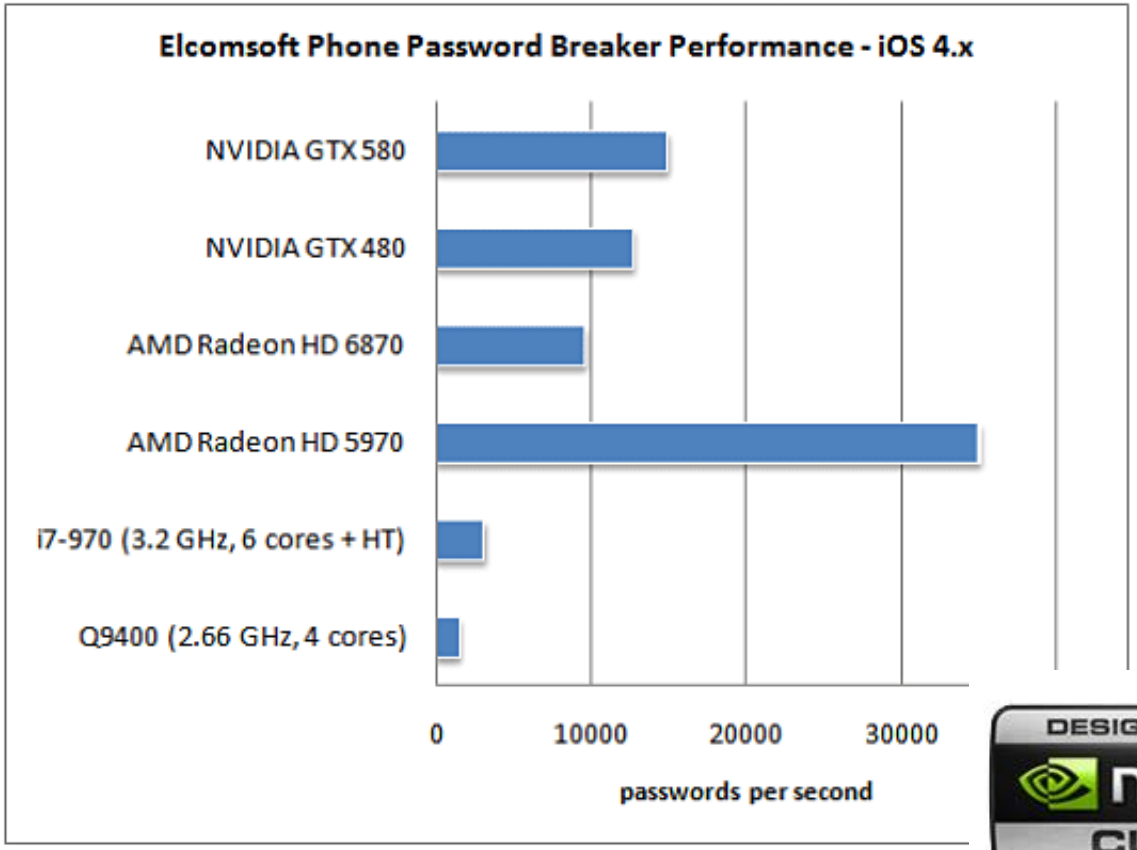
- Try all possible passwords:
  - More commonly, try a subset thereof
- Usually implemented with progressive complexity
- Typically, LM "hash" is attacked first
- Considerations:
  - Very slow
  - All passwords will eventually be found
  - Attack against NT hash is much harder than LM hash

Rainbow Attack

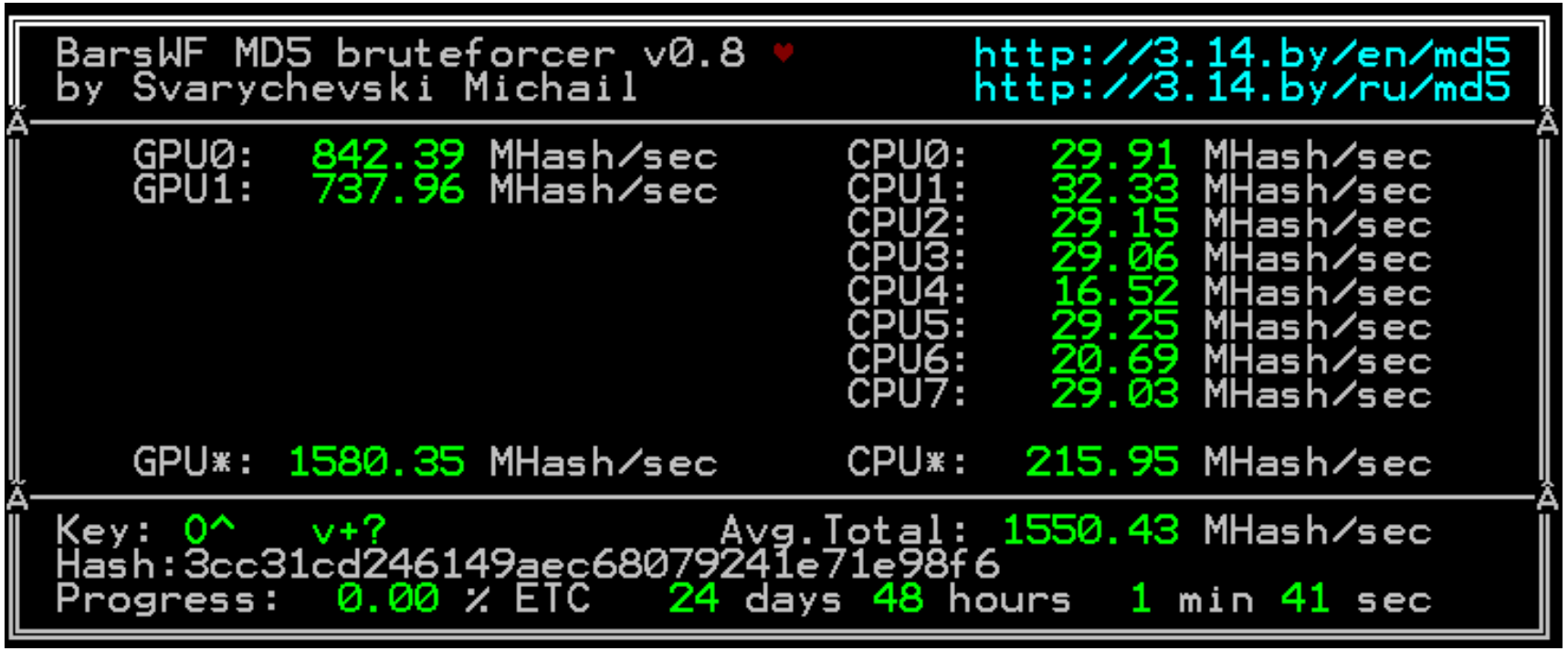
- In rule-based attack, password hash table is generated in advance (only once) and during the recovery process, cracker simply looks up the hash in these pre-computed tables
- A rainbow table is a lookup table specially used in recovering the plaintext password from a ciphertext
- This attack reduces the auditing time for complex passwords

Elcomsoft Phone Password Breaker

- Brute-Force backup password with GPU



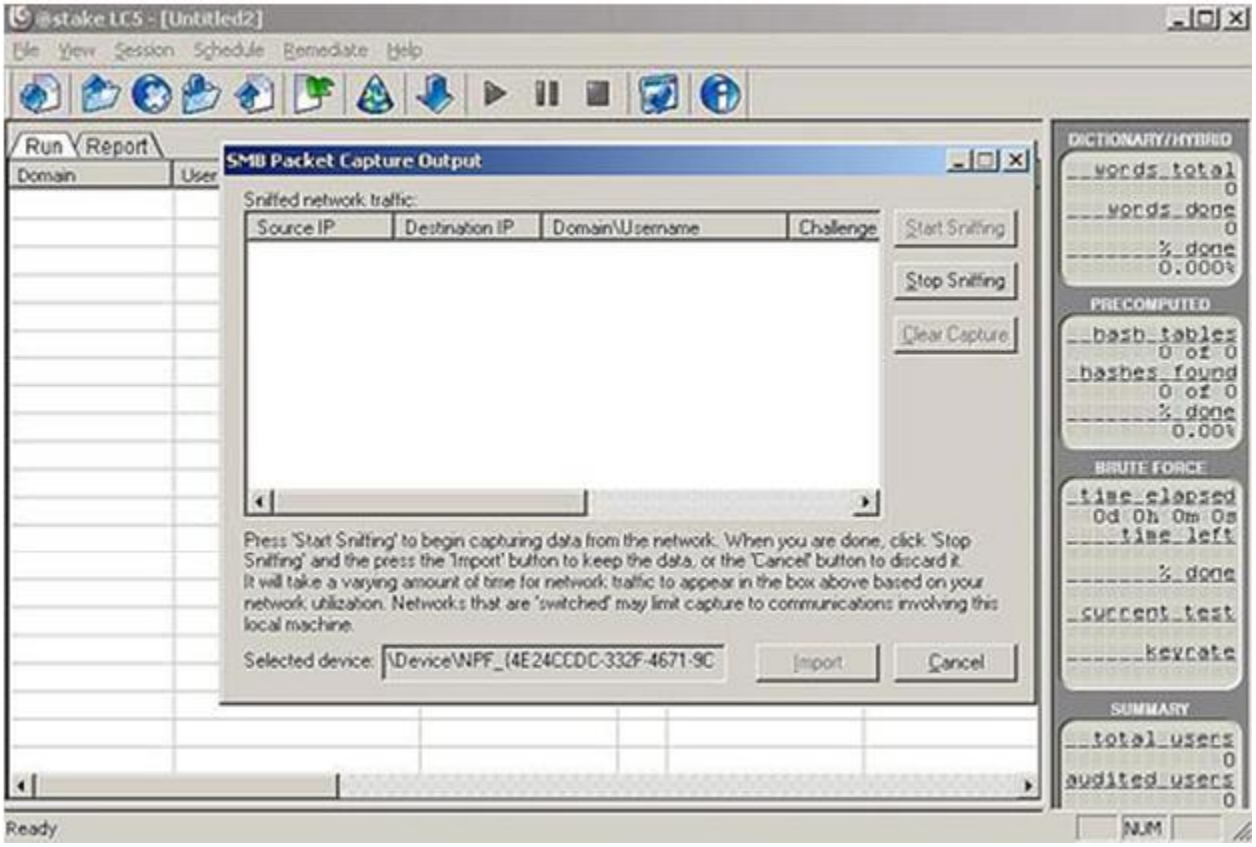
NVIDIA® Tesla™ C1060 + Intel® Core™ i7



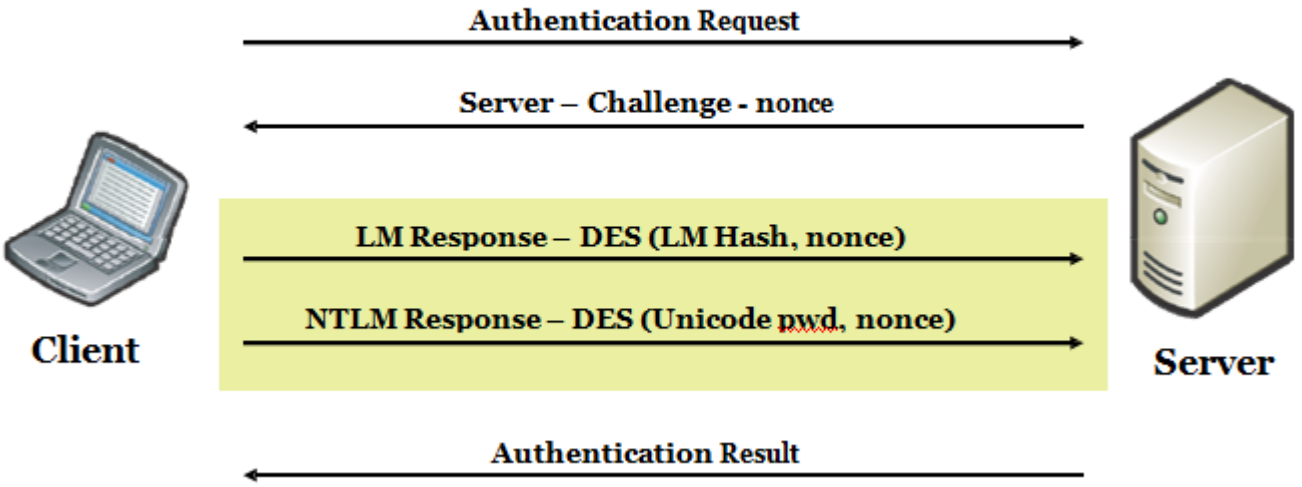
Password Mitigation

- Use the following in place of passwords:
  - Smart cards
    - Two-factor authentication
    - Difficult to thwart
    - High cost of initial deployment
  - Biometric
    - Two- or three-factor authentication
    - Usually defeated with non-technical attacks
    - Very expensive

Hacking Tool: LOphtcrack



NTLM and LM Authentication on the Wire



PASSWORD CRACKING TOOLS

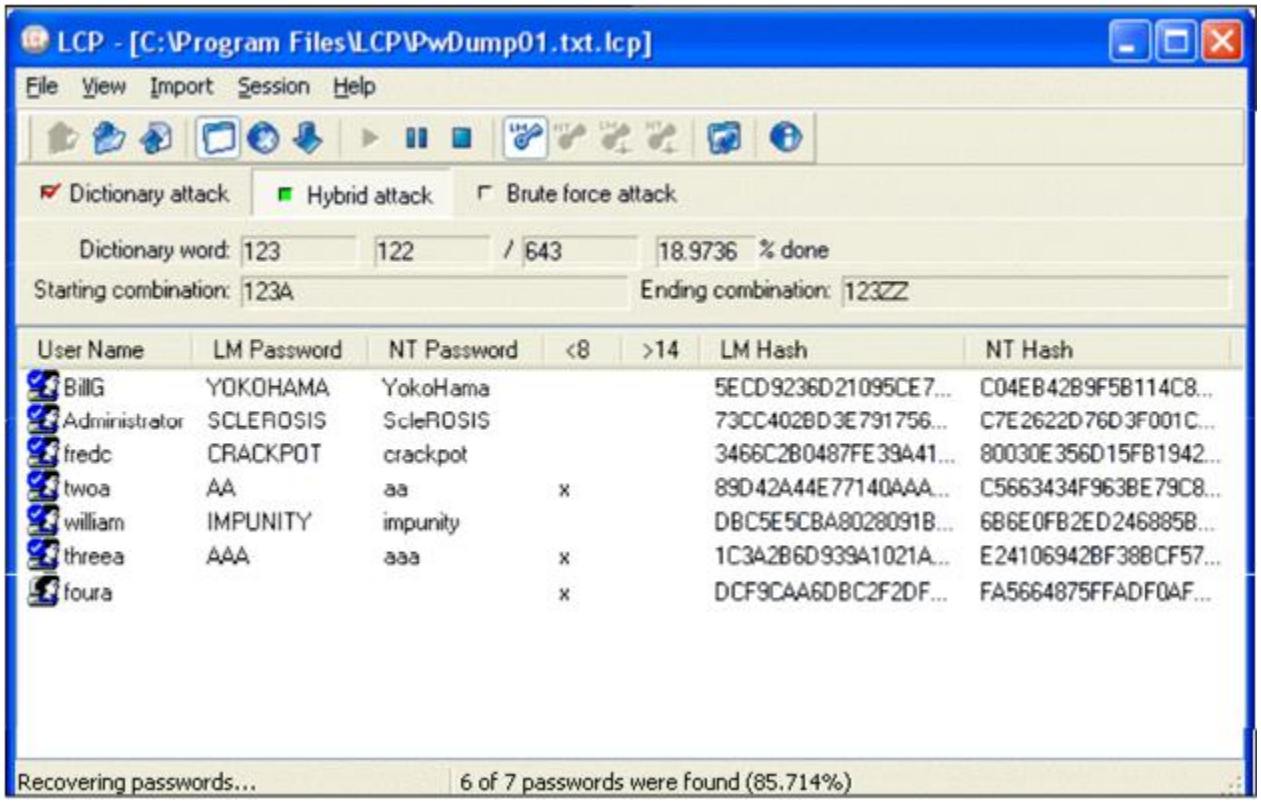
Hacking Tool: John the Ripper

- It is a command-line tool designed to crack both Unix and NT passwords
- The resulting passwords are case insensitive and may not represent the real mixed- case password

```
John the Ripper  Version 1.6  Copyright (c) 1996-98 by Solar Designer

Usage: john [OPTIONS] [PASSWORD-FILES]
-single                  "single crack" mode
-wordfile:FILE -stdin    wordlist mode, read words from FILE or stdin
-rules                   enable rules for wordlist mode
-incremental[:MODE]      incremental mode [using section MODE]
-external:MODE           external mode or word filter
-stdout[:LENGTH]         no cracking, just write words to stdout
-restore[:FILE]          restore an interrupted session [from FILE]
-session:FILE            set session file name to FILE
-status[:FILE]           print status of a session [from FILE]
-makechars:FILE          make a charset, FILE will be overwritten
-show                    show cracked passwords
-test                    perform a benchmark
-users:[-]LOGIN|UID[,..] load this (these) user(s) only
-groups:[-]GID[,..]      load users of this (these) group(s) only
-shells:[-]SHELL[,..]    load users with this (these) shell(s) only
-salts:[-]COUNT          load salts with at least COUNT passwords only
-format:NAME             force ciphertext format NAME (DES/BSDI/MD5/BF/AFS/LM)
-savemem:LEVEL           enable memory saving, at LEVEL 1..3
```

LCP: Screenshot



Password Cracking Countermeasures

- Enforce 8-12 character alphanumeric passwords
- Set the password change policy to 30 day
- Physically isolate and protect the server
- Use SYSKEY utility to store hashes on disk
- Monitor the server logs for brute force attacks on user accounts

LM Hash Backward Compatibility

- LAN Manager (LM) authentication
- Windows NT (NTLM) authentication
- NTLM version 2 (NTLMv2) authentication

How to Disable LM HASH

Method 1: Implement the NoLMHash Policy by Using Group Policy

- Disable "Network security: Do not store LAN Manager hash value on next password change" in Local Security Policy ⇗ Security Options

Method 2: Implement the NoLMHash Policy by Editing the Registry

- Locate the following key:
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa
- Add Key, type NoLMHash

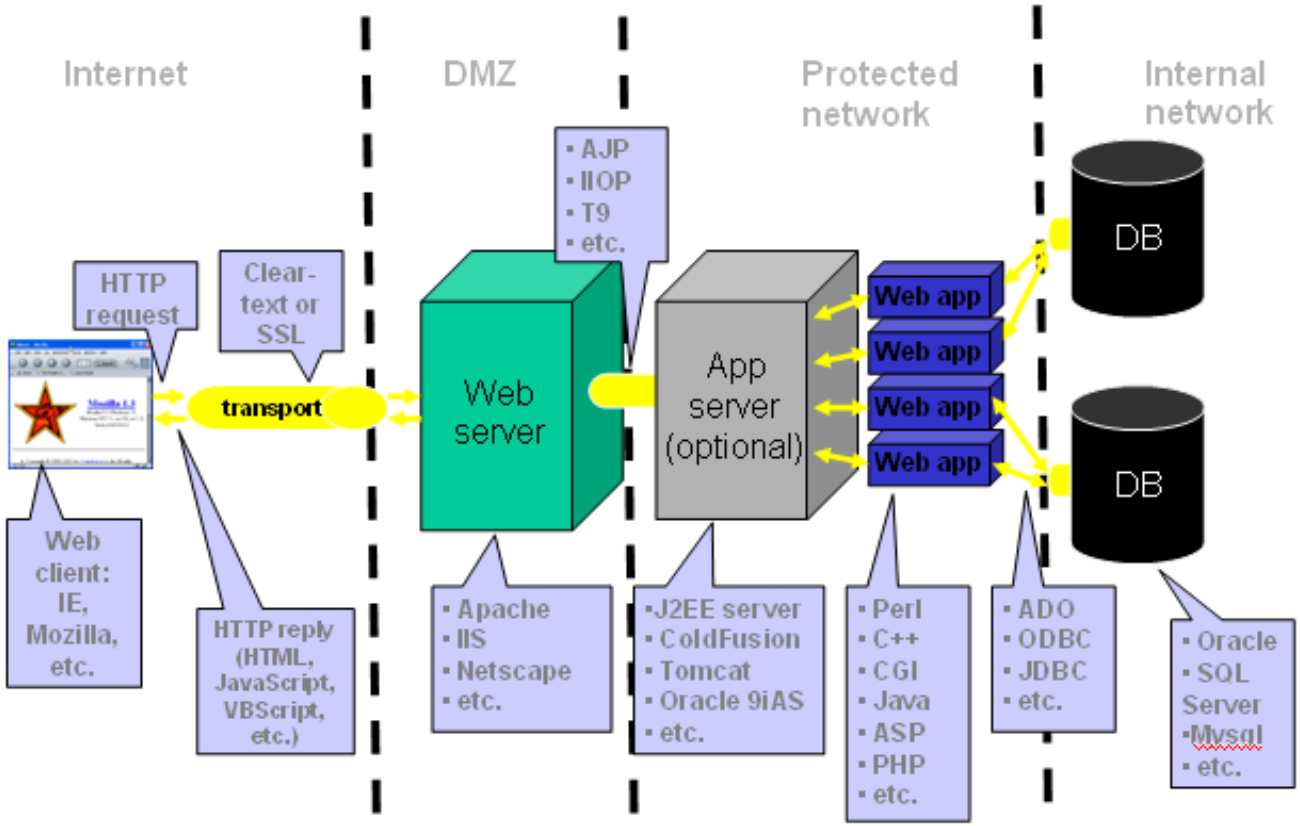
Method 3: Use a Password that is at least 15 Characters Long

- Windows store an LM hash value that cannot be used to authenticate the user

Web Application Security

ACIS Professional Center

The Evolution of Web Applications

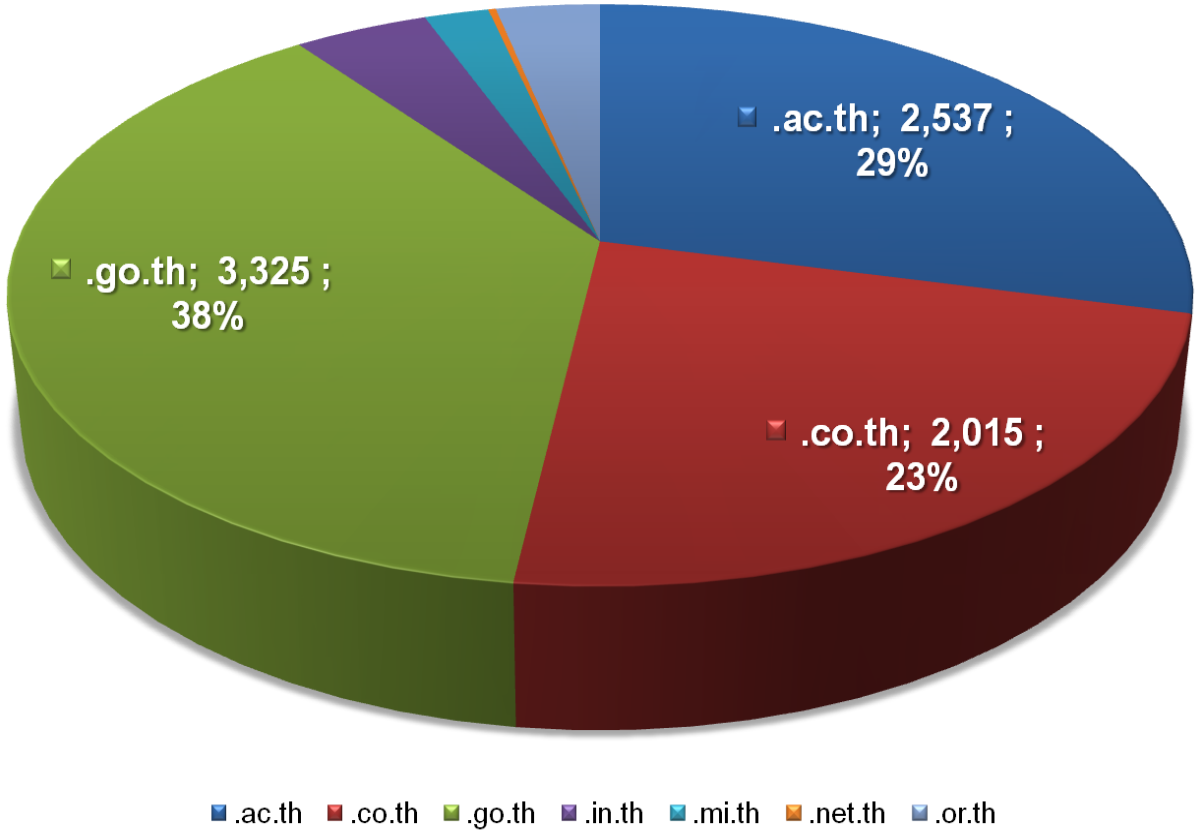


The Evolution of Web Applications

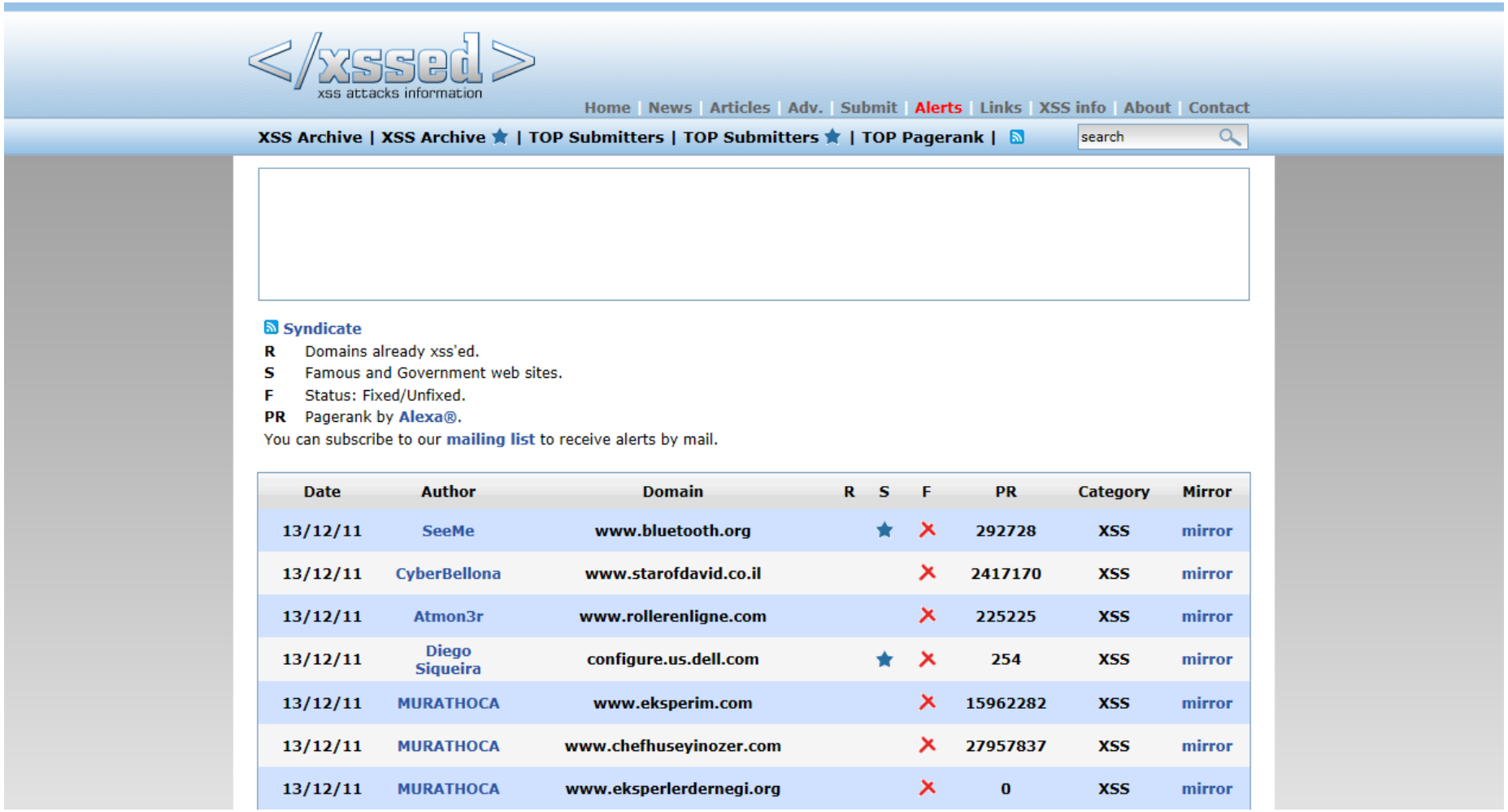
- You can be Vulnerable…
  - 7 out of 10 sites are vulnerable
  - 70% of cyber attacks are on web ports
  - 95% of companies are hacked through web ports
  - Most popular attacks are towards Web Servers
  - Web ports are popular targets – Securing 80 & 443 is a concern
  - Web 2.0 – More on web ..

The Evolution of Web Applications

- Zone-H Digital Attacks in Thailand (Web Defacement)



www.xssed.com



www.xssed.com

Web Application Technologies

- HTTP Request and Response
  - Methods – GET, POST, HEAD
  - New Methods in HTTP 1.1
  - Header fields: Server, Host, Length, etc.
  - Response codes – 200, 403, 404, 500, etc.

What's Changed?

It's About Risks, Not Just Vulnerabilities

- New title is: "The Top 10 Most Critical Web Application Security Risks"

OWASP Top 10 Risk Rating Methodology

- Based on the OWASP Risk Rating Methodology, used to prioritize Top 10

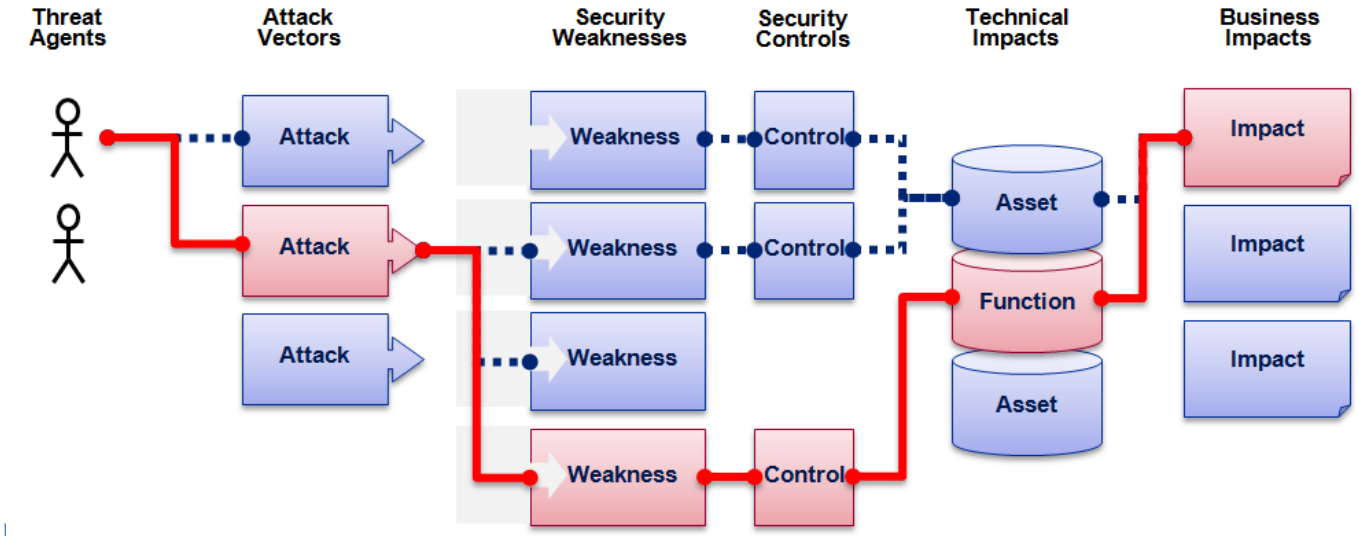
2 Risks Added, 2 Dropped

- **Added: A6 – Security Misconfiguration**
  - Was A10 in 2004 Top 10: Insecure Configuration Management
- **Added: A8 – Unvalidated Redirects and Forwards**
  - Relatively common and VERY dangerous flaw that is not well known
- **Removed: A3 – Malicious File Execution**
  - Primarily a PHP flaw that is dropping in prevalence
- **Removed: A6 – Information Leakage and Improper Error Handling**
  - A very prevalent flaw, that does not introduce much risk (normally)

Mapping from 2007 to 2010 Top 10

OWASP Top 10 – 2007 (Previous)		OWASP Top 10 – 2010 (New)
A2 – Injection Flaws	↑	A1 – Injection
A1 – Cross Site Scripting (XSS)	=	A2 – Cross Site Scripting (XSS)
A7 – Broken Authentication and Session Management	=	A3 – Broken Authentication and Session Management
A4 – Insecure Direct Object Reference	=	A4 – Insecure Direct Object References
A5 – Cross Site Request Forgery (CSRF)	+	A5 – Cross Site Request Forgery (CSRF)
<was T10 2004 A10 – Insecure Configuration Management>	↑	A6 – Security Misconfiguration (NEW)
A10 – Failure to Restrict URL Access	+	A7 – Failure to Restrict URL Access
<not in T10 2007>	↓	A8 – Unvalidated Redirects and Forwards (NEW)
A8 – Insecure Cryptographic Storage	↓	A9 – Insecure Cryptographic Storage
A9 – Insecure Communications	-	A10 – Insufficient Transport Layer Protection
A3 – Malicious File Execution		<dropped from T10 2010>
A6 – Information Leakage and Improper Error Handling	-	<dropped from T10 2010>

OWASP Top 10 Risk Rating Methodology

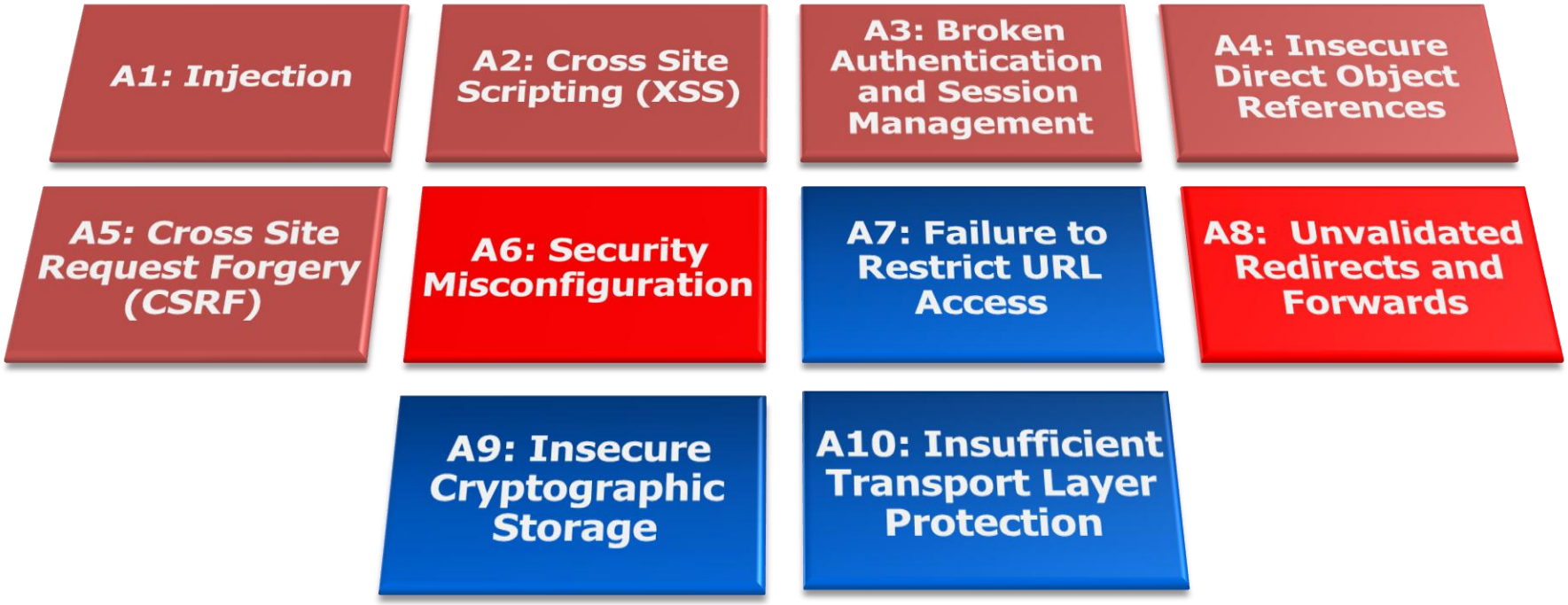


Threat Agent	Attack Vector	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impact
	1 Easy	Widespread	Easy	Severe	
?	Average	Common	Average	Moderate	?
	2 Difficult	Uncommon	Difficult	Minor	
	3	2	1	1	2
			1.3	*	2

XSS Example

2.6 weighted risk rating

The 'new' OWASP Top Ten (2010 rc1)



OWASP
The Open Web Application Security Project
http://www.owasp.org

http://www.owasp.org/index.php/Top_10

A1 – Injection

Injection means…

- Tricking an application into including unintended commands in the data sent to an interpreter

Interpreters…

- Take strings and interpret them as commands
- SQL, OS Shell, LDAP, XPath, Hibernate, etc…

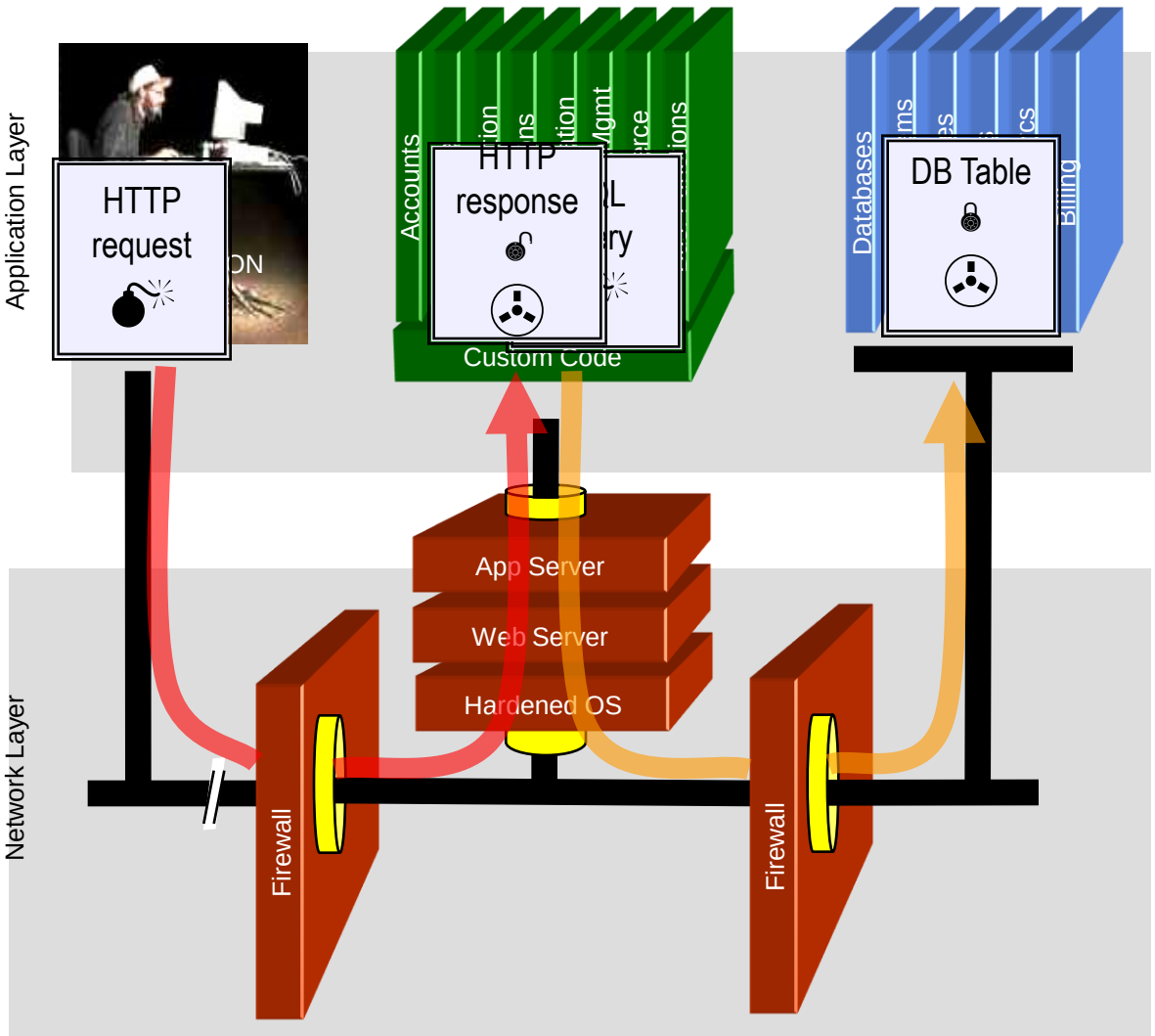
SQL injection is still quite common

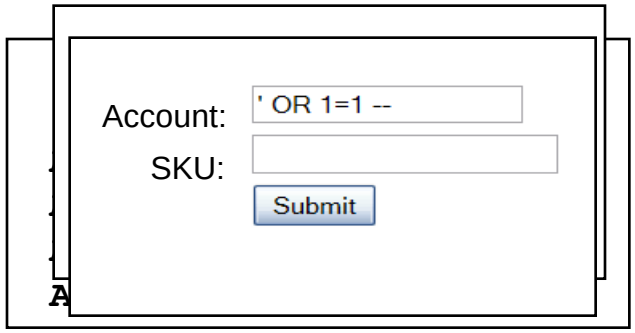
- Many applications still susceptible (really don't know why)
- Even though it's usually very simple to avoid

Typical Impact

- Usually severe. Entire database can usually be read or modified
- May also allow full database schema, or account access, or even OS level access

SQL Injection – Illustrated





1. Application presents a form to the attacker
2. Attacker sends an attack in the form data
3. Application forwards attack to the database in a SQL query
4. Database runs query containing attack and sends encrypted results back to application
5. Application decrypts data as normal and sends results to the user

A1 – Avoid Injection Flaws

- Recommendations
  1. Avoid the interpreter entirely, or
  2. Use an interface that supports bind variables (e.g., prepared statements, or stored procedures),
     - Bind variables allow the interpreter to distinguish between code and data
  3. Encode all user input before passing it to the interpreter
  - Always perform 'white list' input validation on all user supplied input
  - Always minimize database privileges to reduce the impact of a flaw

- References
  - For more details, read the new http://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

A2 – Cross-Site Scripting (XSS)

Occurs any time…

- Raw data from attacker is sent to an innocent user's browser

Raw data…

- Stored in database
- Reflected from web input (form field, hidden field, URL, etc…)
- Sent directly into rich JavaScript client

Virtually every web application has this problem

- Try this in your browser – javascript:alert(document.cookie)

Typical Impact

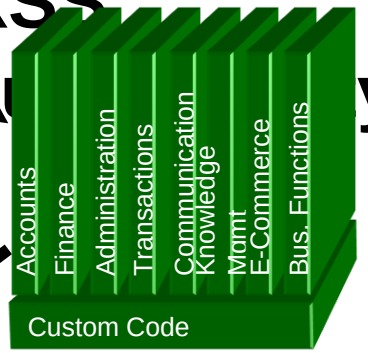
- Steal user's session, steal sensitive data, rewrite web page, redirect user to phishing or malware site
- Most Severe: Install XSS proxy which allows attacker to observe and direct all user's behavior on vulnerable site and force user to other sites

Cross-Site Scripting Illustrated

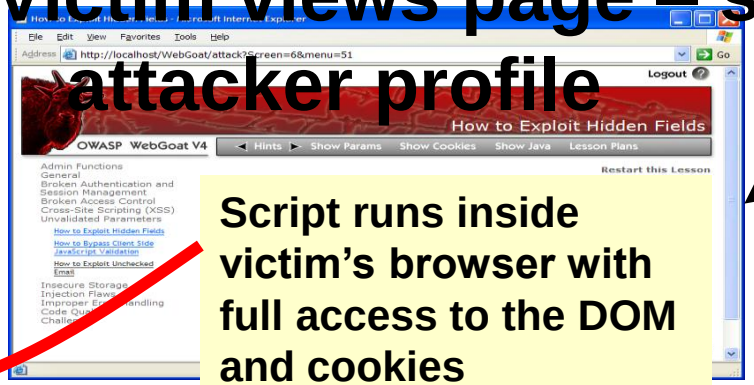
1 Attacker sets the trap –

Attacker enters a malicious script into a web page that stores the data on the server

Application with stored XSS v y



2 Victim views page – sees attacker profile



Script runs inside victim's browser with full access to the DOM and cookies

3 Script silently sends attacker Victim's session cookie

A2 – Avoiding XSS Flaws

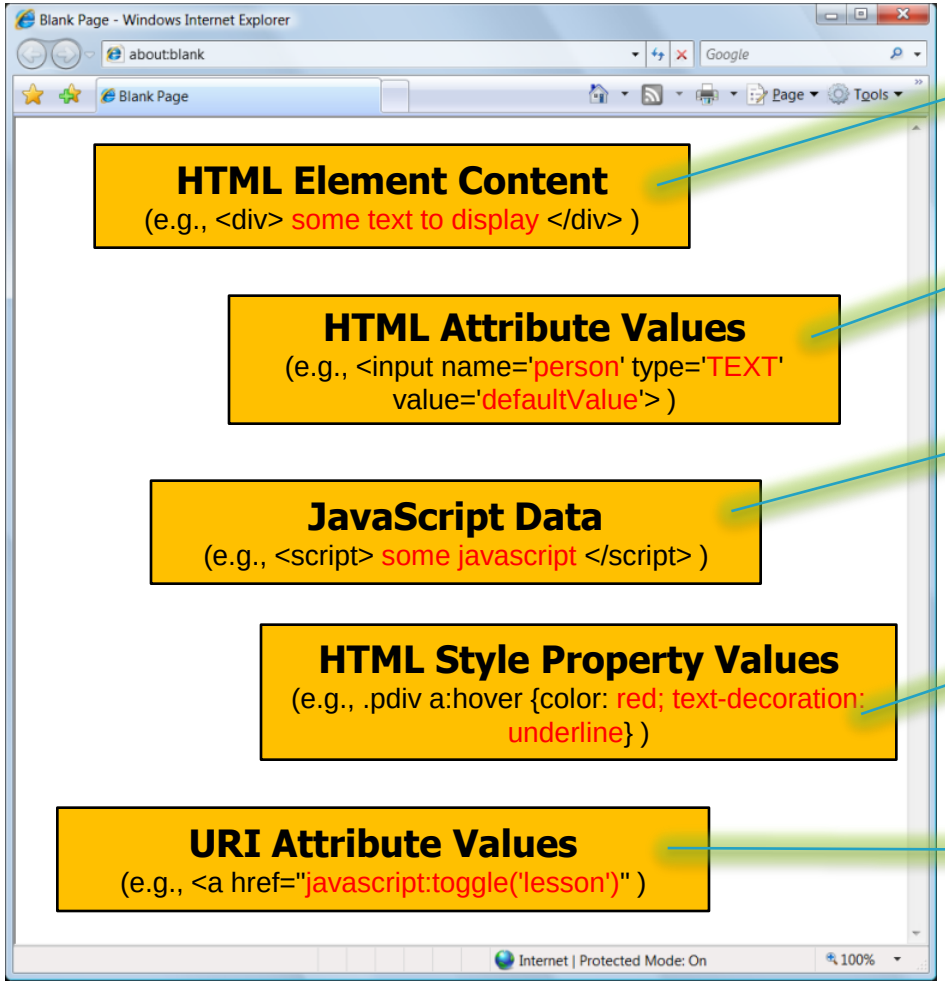
- Recommendations
  - Eliminate Flaw
    - Don't include user supplied input in the output page
  - Defend Against the Flaw
    - Primary Recommendation: Output encode all user supplied input

      (Use OWASP's ESAPI to output encode:

      http://www.owasp.org/index.php/ESAPI
    - Perform 'white list' input validation on all user input to be included in page
    - For large chunks of user supplied HTML, use OWASP's AntiSamy to sanitize this HTML to make it safe

      See: http://www.owasp.org/index.php/AntiSamy

Safe Escaping Schemes in Various HTML Execution Contexts



#1: (&, <, >, ") → &entity; (', /) → &#xHH;
ESAPI: encodeForHTML()

#2: All non-alphanumeric < 256 → &#xHH
ESAPI: encodeForHTMLAttribute()

#3: All non-alphanumeric < 256 → \xHH
ESAPI: encodeForJavaScript()

#4: All non-alphanumeric < 256 → \HH
ESAPI: encodeForCSS()

#5: All non-alphanumeric < 256 → %HH
ESAPI: encodeForURL()

ALL other contexts CANNOT include Untrusted Data

Recommendation: Only allow #1 and #2 and disallow all others

See: www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet for more details

A3 – Broken Authentication and Session Management

HTTP is a "stateless" protocol

- Means credentials have to go with every request
- Should use SSL for everything requiring authentication

Session management flaws

- SESSION ID used to track state since HTTP doesn't
  - and it is just as good as credentials to an attacker
- SESSION ID is typically exposed on the network, in browser, in logs, …

Beware the side-doors

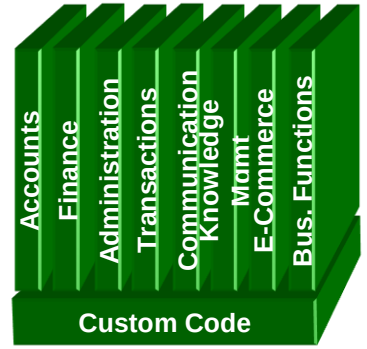
- Change my password, remember my password, forgot my password, secret question, logout, email address, etc…

Typical Impact

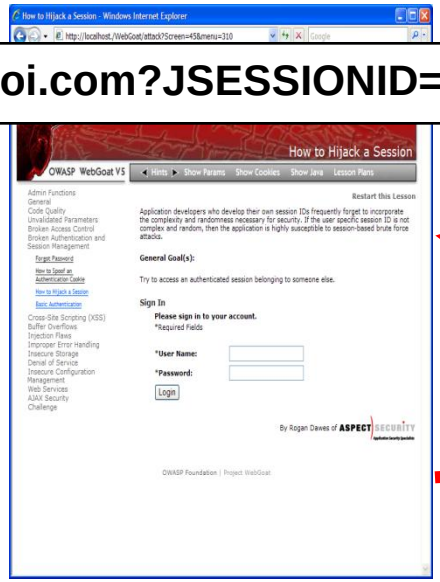
- User accounts compromised or user sessions hijacked

Broken Authentication Illustrated

1 User sends credentials



www.boi.com?JSESSIONID=9FA1DB9EA...



Site uses URL rewriting
(i.e., put session in URL)

2

3 User clicks on a link to http://www.hacker.com in a forum

Hacker checks referer logs on www.hacker.com and finds user's JSESSIONID

4

5 Hacker uses JSESSIONID and takes over victim's account

A3 – Avoiding Broken Authentication and Session Management

- Verify your architecture
  - Authentication should be simple, centralized, and standardized
  - Use the standard session id provided by your container
  - Be sure SSL protects both credentials and session id at all times

- Verify the implementation
  - Forget automated analysis approaches
  - Check your SSL certificate
  - Examine all the authentication-related functions
  - Verify that logoff actually destroys the session
  - Use OWASP's WebScarab to test the implementation

A4 – Insecure Direct Object References

How do you protect access to your data?

- This is part of enforcing proper "Authorization", along with A7 – Failure to Restrict URL Access

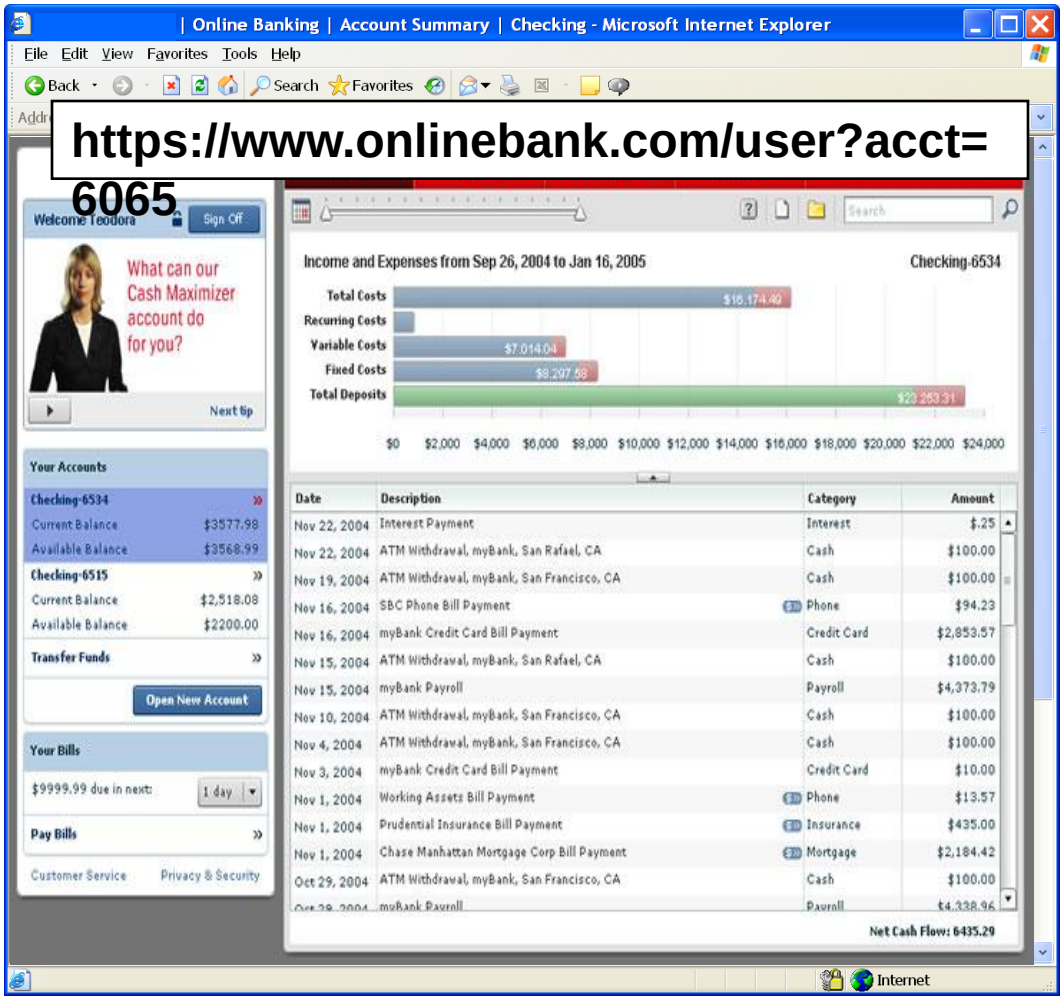
A common mistake …

- Only listing the 'authorized' objects for the current user, or
- Hiding the object references in hidden fields
- … and then not enforcing these restrictions on the server side
- This is called presentation layer access control, and doesn't work
- Attacker simply tampers with parameter value

Typical Impact

- Users are able to access unauthorized files or data

Insecure Direct Object References Illustrated



- Attacker notices his acct parameter is 6065

  ?acct=6065

- He modifies it to a nearby number

  ?acct=6066

- Attacker views the victim's account information

A4 – Avoiding Insecure Direct Object References

- Eliminate the direct object reference
  - Replace them with a temporary mapping value (e.g. 1, 2, 3)
  - ESAPI provides support for numeric & random mappings
    - IntegerAccessReferenceMap & RandomAccessReferenceMap

- Validate the direct object reference
  - Verify the parameter value is properly formatted
  - Verify the user is allowed to access the target object
    - Query constraints work great!
  - Verify the requested mode of access is allowed to the target object (e.g., read, write, delete)

A5 – Cross Site Request Forgery (CSRF)

Cross Site Request Forgery

- An attack where the victim's browser is tricked into issuing a command to a vulnerable web application
- Vulnerability is caused by browsers automatically including user authentication data (session ID, IP address, Windows domain credentials, …) with each request

Imagine…

- What if a hacker could steer your mouse and get you to click on links in your online banking application?
- What could they make you do?

Typical Impact

- Initiate transactions (transfer funds, logout user, close account)
- Access sensitive data
- Change account details

CSRF Vulnerability Pattern

- The Problem
  - Web browsers automatically include most credentials with each request
  - Even for requests caused by a form, script, or image on another site
- All sites relying solely on automatic credentials are vulnerable!
  - (almost all sites are this way)
- Automatically Provided Credentials
  - Session cookie
  - Basic authentication header
  - IP address
  - Client side SSL certificates
  - Windows domain authentication

CSRF Illustrated

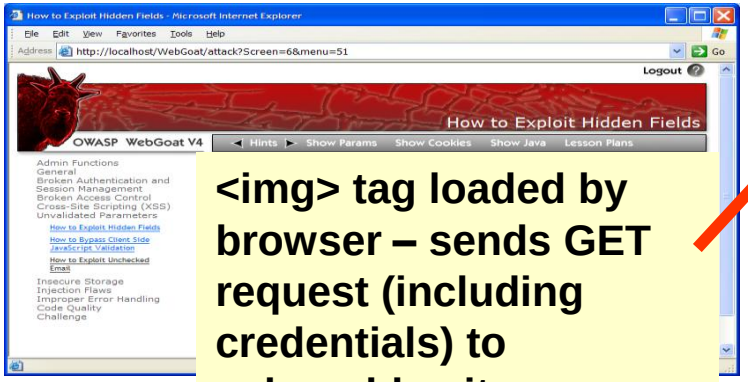
1

Attacker sets the trap on some website on the internet (or simply via an e-mail)

Hidden <img> tag contains attack against vulnerable site

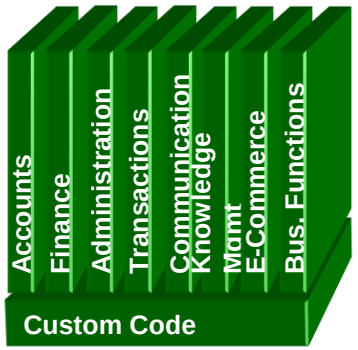
2

While logged into vulnerable site, victim views attacker site



<img> tag loaded by browser – sends GET request (including credentials) to vulnerable site

Application with CSRF vulnerability



3

Vulnerable site sees legitimate request from victim and performs the action requested

A5 – Avoiding CSRF Flaws

- Add a secret, not automatically submitted, token to ALL sensitive requests
  - This makes it impossible for the attacker to spoof the request
    - (unless there's an XSS hole in your application)
  - Tokens should be cryptographically strong or random

- Options
  - Store a single token in the session and add it to all forms and links
    - **Hidden Field:** `<input name="token" value="687965fdfaew87agrde" type="hidden"/>`
    - **Single use URL:** `/accounts/687965fdfaew87agrde`
    - **Form Token:** `/accounts?auth=687965fdfaew87agrde` **...**
  - Beware exposing the token in a referer header
    - Hidden fields are recommended
  - Can have a unique token for each function
    - Use a hash of function name, session id, and a secret
  - Can require secondary authentication for sensitive functions (e.g., eTrade)

- Don't allow attackers to store attacks on your site
  - Properly encode all input on the way out
  - This renders all links/requests inert in most interpreters

See the new: www.owasp.org/index.php/CSRF_Prevention_Cheat_Sheet for more details

A6 – Security Misconfiguration

Web applications rely on a secure foundation

- All through the network and platform
- Don't forget the development environment

Is your source code a secret?

- Think of all the places your source code goes
- Security should not require secret source code

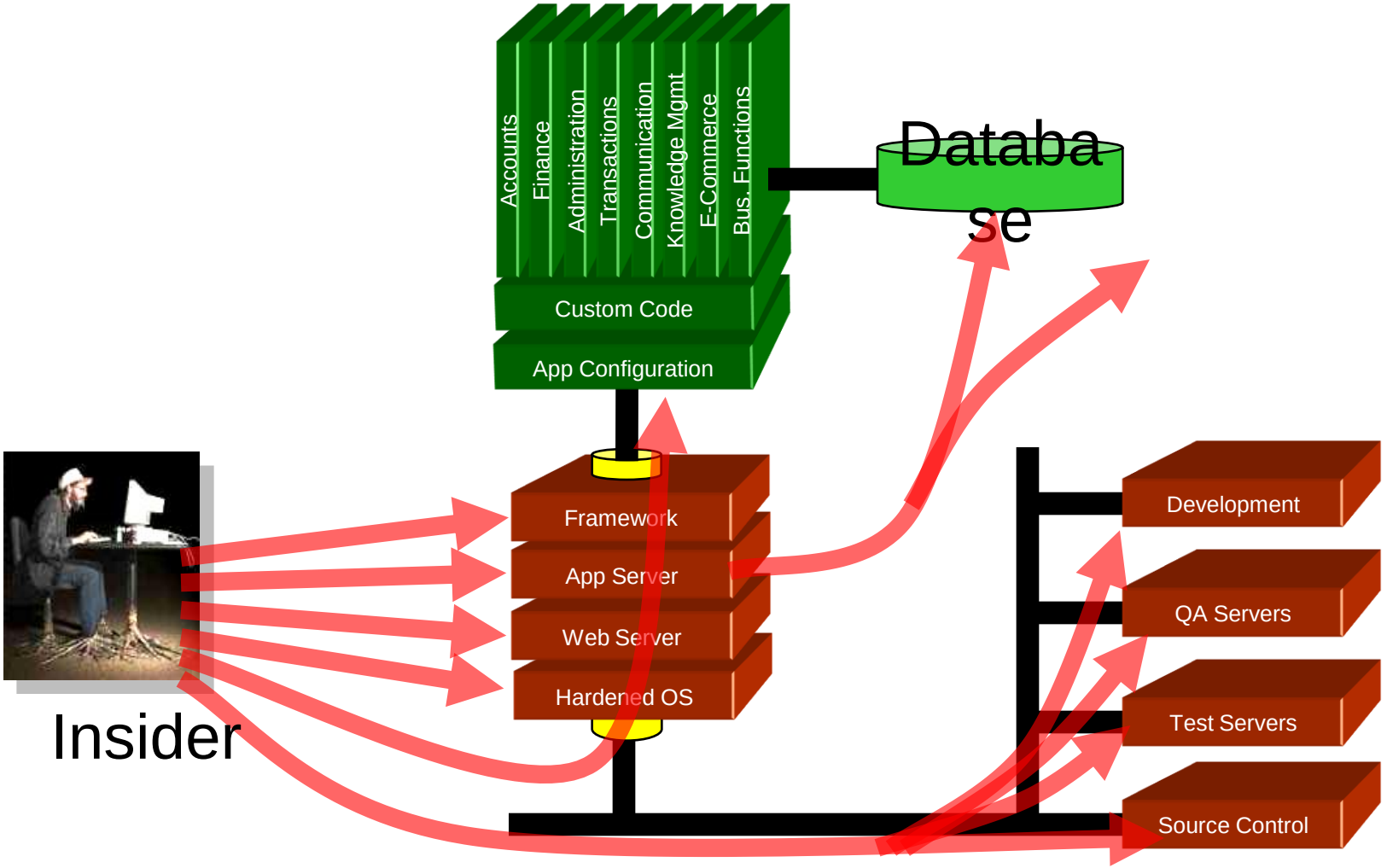
CM must extend to all parts of the application

- All credentials should change in production

Typical Impact

- Install backdoor through missing network or server patch
- XSS flaw exploits due to missing application framework patches
- Unauthorized access to default accounts, application functionality or data, or unused but accessible functionality due to poor server configuration

Security Misconfiguration Illustrated



A6 – Avoiding Security Misconfiguration

- Verify your system's configuration management
  - Secure configuration "hardening" guideline
    - Automation is REALLY USEFUL here
  - Must cover entire platform and application
  - Keep up with patches for ALL components
    - This includes software libraries, not just OS and Server applications
  - Analyze security effects of changes

- Can you "dump" the application configuration
  - Build reporting into your process
  - If you can't verify it, it isn't secure

- Verify the implementation
  - Scanning finds generic configuration and missing patch problems

A7 – Failure to Restrict URL Access

How do you protect access to URLs (pages)?

- This is part of enforcing proper "authorization", along with A4 – Insecure Direct Object References

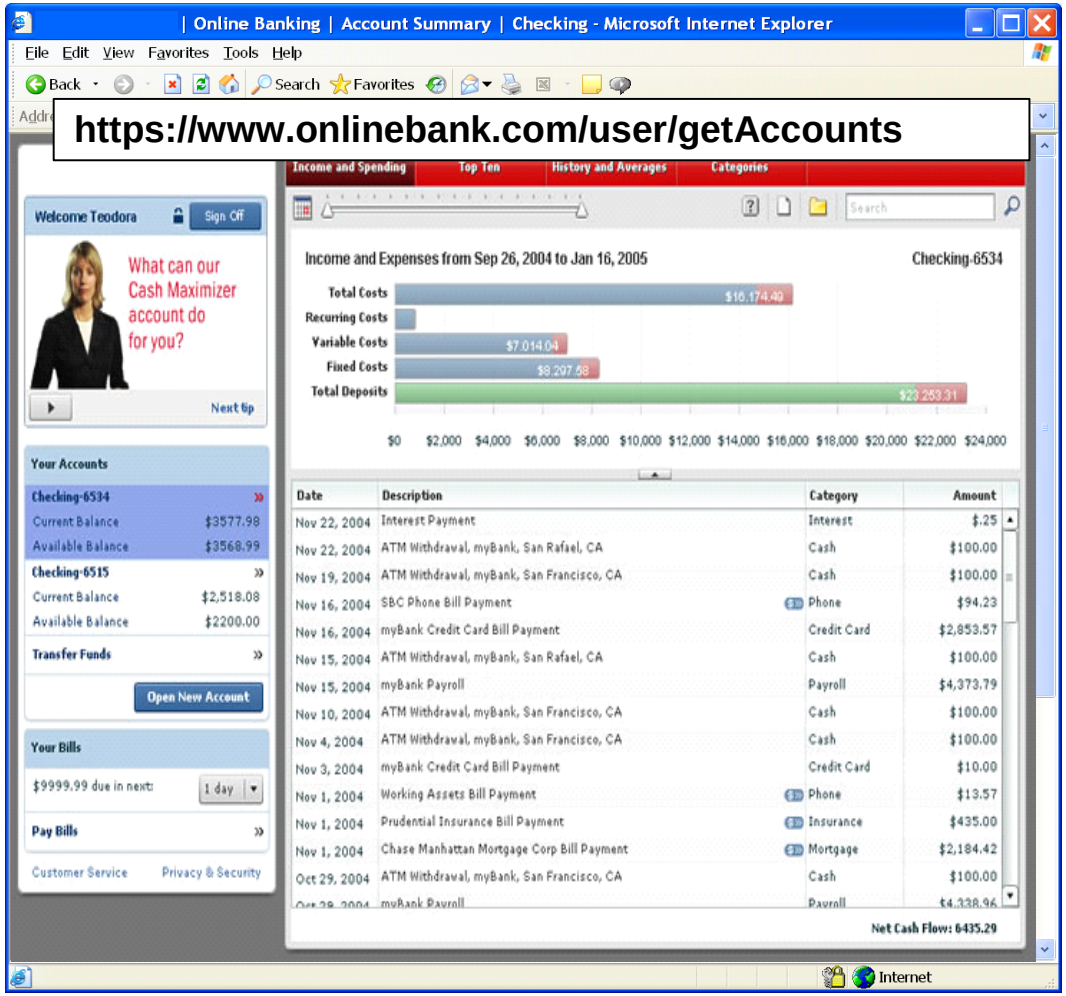
A common mistake …

- Displaying only authorized links and menu choices
- This is called presentation layer access control, and doesn't work
- Attacker simply forges direct access to 'unauthorized' pages

Typical Impact

- Attackers invoke functions and services they're not authorized for
- Access other user's accounts and data
- Perform privileged actions

Failure to Restrict URL Access Illustrated



- Attacker notices the URL indicates his role

  /user/getAccounts

- He modifies it to another directory (role)

  /admin/getAccounts, or

  /manager/getAccounts

- Attacker views more accounts than just their own

A7 – Avoiding URL Access Control Flaws

- For each URL, a site needs to do 3 things
  - Restrict access to authenticated users (if not public)
  - Enforce any user or role based permissions (if private)
  - Completely disallow requests to unauthorized page types (e.g., config files, log files, source files, etc.)
- Verify your architecture
  - Use a simple, positive model at every layer
  - Be sure you actually have a mechanism at every layer
- Verify the implementation
  - Forget automated analysis approaches
  - Verify that each URL in your application is protected by either
    - An external filter, like Java EE web.xml or a commercial product
    - Or internal checks in YOUR code – Use ESAPI's isAuthorizedForURL() method
  - Verify the server configuration disallows requests to unauthorized file types
  - Use WebScarab or your browser to forge unauthorized requests

A8 – Unvalidated Redirects and Forwards

Web application redirects are very common

- And frequently include user supplied parameters in the destination URL
- If they aren't validated, attacker can send victim to a site of their choice

Forwards (aka Transfer in .NET) are common too

- They internally send the request to a new page in the same application
- Sometimes parameters define the target page
- If not validated, attacker may be able to use unvalidated forward to bypass authentication or authorization checks

Typical Impact

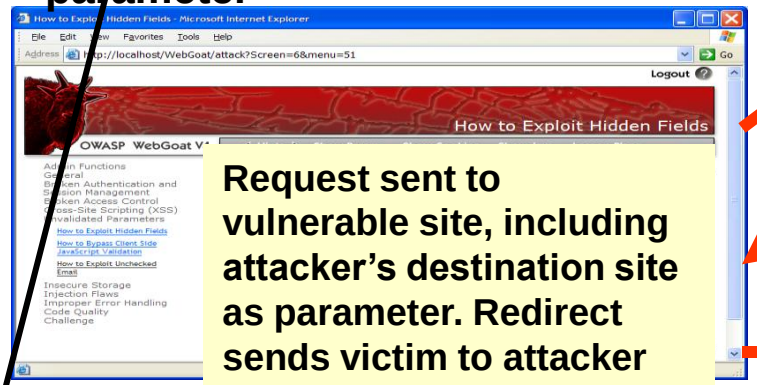
- Redirect victim to phishing or malware site
- Attacker's request is forwarded past security checks, allowing unauthorized function or data access

Unvalidated Redirect Illustrated

1 Attacker sends attack to victim via email or webpage

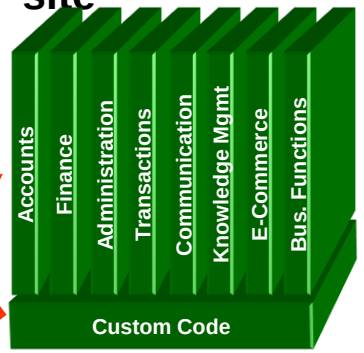
From: Internal Revenue Service
Subject: Your Unclaimed Tax Refund
Our records show you have an unclaimed federal tax refund.
Please click here to initiate your claim

2 Victim clicks link containing unvalidated parameter



Request sent to vulnerable site, including attacker's destination site as parameter. Redirect sends victim to attacker site

3 Application redirects victim to attacker's site

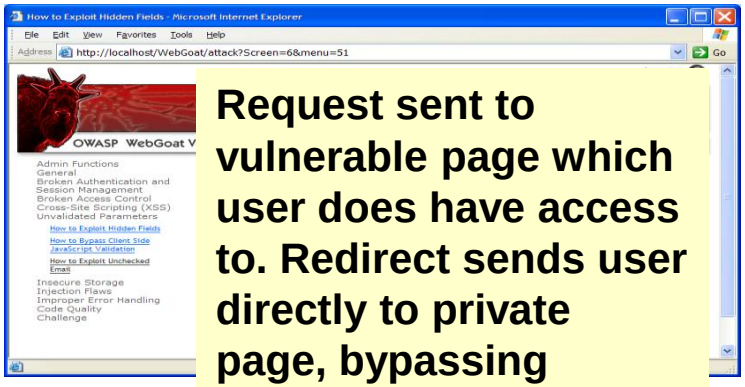


4 Evil site installs malware on victim, or phish's for private information

http://www.irs.gov/taxrefund/claim.jsp?year=2006& … &dest=www.evilsite.com

Unvalidated Forward Illustrated

1 **Attacker sends attack to vulnerable page they have access to**



Request sent to vulnerable page which user does have access to. Redirect sends user directly to private page, bypassing access control.

2 **Application authorizes request, which continues to vulnerable page**

Filter

```
public void doPost( HttpServletRequest request,
HttpServletResponse response) {
    try {
        String target = request.getParameter( "dest" ) );
        ...
        request.getRequestDispatcher( target
        ).forward(request, response);
    }
    catch ( ...
```

```
public void sensitiveMethod(
HttpServletRequest request,
HttpServletResponse response) {
    try {
          // Do sensitive stuff here.
          ...
    }
    catch ( ...
```

3 **Forwarding page fails to validate parameter, sending attacker to unauthorized page, bypassing access control**

A8 – Avoiding Unvalidated Redirects and Forwards

- There are a number of options
  1. Avoid using redirects and forwards as much as you can
  2. If used, don't involve user parameters in defining the target URL
  3. If you 'must' involve user parameters, then either
     a) Validate each parameter to ensure its valid and authorized for the current user, or
     b) (preferred) – Use server side mapping to translate choice provided to user with actual target page
  - Defense in depth: For redirects, validate the target URL after it is calculated to make sure it goes to an authorized external site
  - ESAPI can do this for you!!
    - See: SecurityWrapperResponse.sendRedirect(URL)
    - http://owasp-esapi-java.googlecode.com/svn/trunk_doc/org/owasp/esapi/filters/SecurityWrapperResponse.html#sendRedirect(java.lang.String)

- Some thoughts about protecting Forwards
  - Ideally, you'd call the access controller to make sure the user is authorized before you perform the forward (with ESAPI, this is easy)
  - With an external filter, like Siteminder, this is not very practical
  - Next best is to make sure that users who can access the original page are ALL authorized to access the target page.

A9 – Insecure Cryptographic Storage

Storing sensitive data insecurely

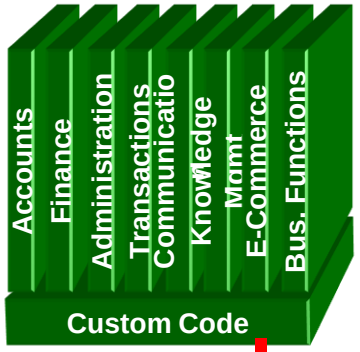
- Failure to identify all sensitive data
- Failure to identify all the places that this sensitive data gets stored
  - Databases, files, directories, log files, backups, etc.
- Failure to properly protect this data in every location

Typical Impact

- Attackers access or modify confidential or private information
  - e.g, credit cards, health care records, financial data (yours or your customers)
- Attackers extract secrets to use in additional attacks
- Company embarrassment, customer dissatisfaction, and loss of trust
- Expense of cleaning up the incident, such as forensics, sending apology letters, reissuing thousands of credit cards, providing identity theft insurance
- Business gets sued and/or fined

Insecure Cryptographic Storage Illustrated

1 Victim enters credit card number in form



Log files

2 Error handler logs CC details because merchant gateway is unavailable

3 Logs are accessible to all members of IT staff for debugging purposes

4 Malicious insider steals 4 million credit card numbers

A9 – Avoiding Insecure Cryptographic Storage

- Verify your architecture
  - Identify all sensitive data
  - Identify all the places that data is stored
  - Ensure threat model accounts for possible attacks
  - Use encryption to counter the threats, don't just 'encrypt' the data
- Protect with appropriate mechanisms
  - File encryption, database encryption, data element encryption
- Use the mechanisms correctly
  - Use standard strong algorithms
  - Generate, distribute, and protect keys properly
  - Be prepared for key change

A9 – Avoiding Insecure Cryptographic Storage

- Verify the implementation
  - A standard strong algorithm is used, and it's the proper algorithm for this situation
  - All keys, certificates, and passwords are properly stored and protected
  - Safe key distribution and an effective plan for key change are in place
  - Analyze encryption code for common flaws

A10 – Insufficient Transport Layer Protection

Transmitting sensitive data insecurely

- Failure to identify all sensitive data
- Failure to identify all the places that this sensitive data is sent
  - On the web, to backend databases, to business partners, internal communications
- Failure to properly protect this data in every location

Typical Impact

- Attackers access or modify confidential or private information
  - e.g, credit cards, health care records, financial data (yours or your customers)
- Attackers extract secrets to use in additional attacks
- Company embarrassment, customer dissatisfaction, and loss of trust
- Expense of cleaning up the incident
- Business gets sued and/or fined

Insufficient Transport Layer Protection Illustrated

External Victim

Custom Code

Backend Systems

Business Pa

Employees

External Attacker

1 External attacker steals credentials and

Internal Attacker

2 Internal attacker steals credentials

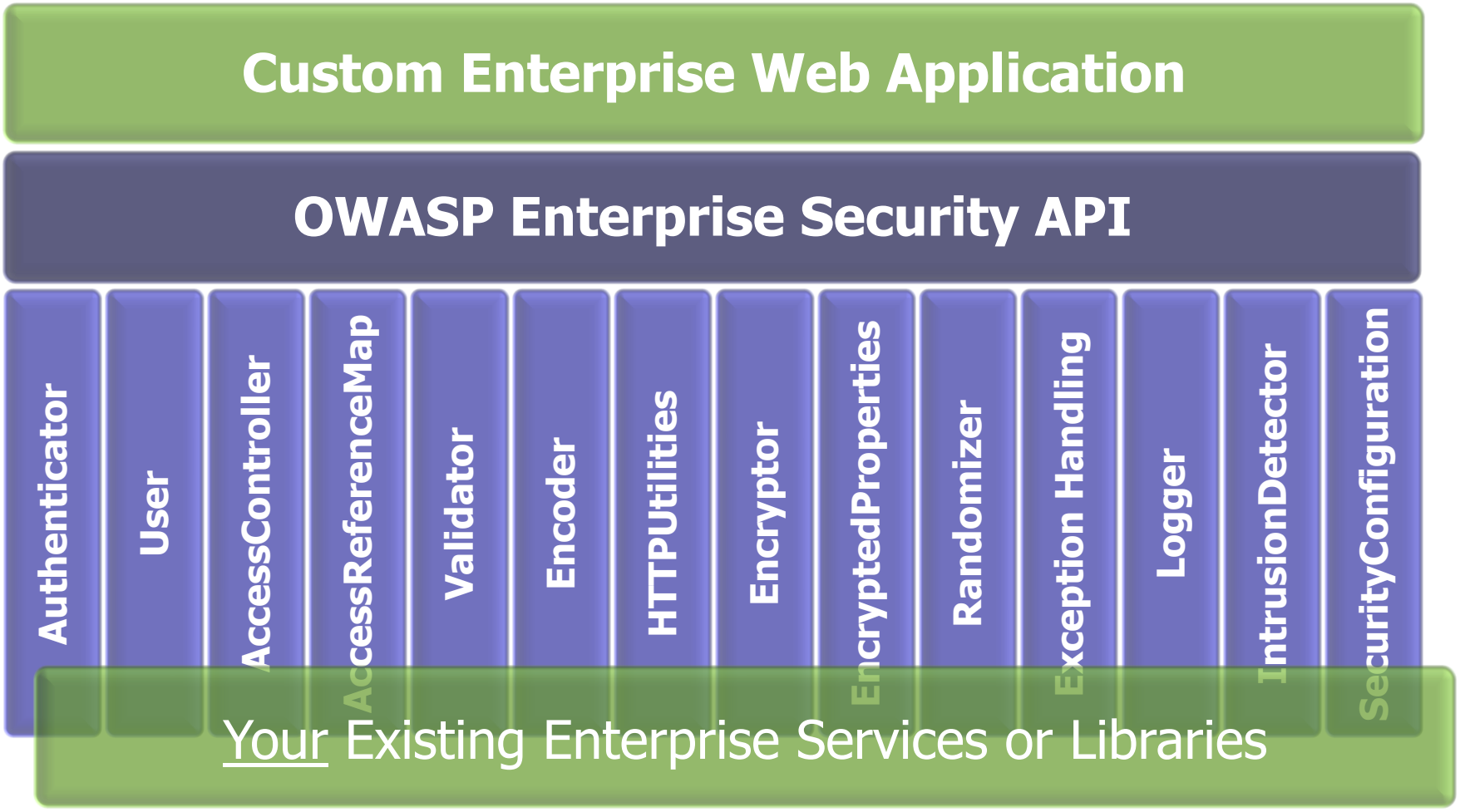
A10 – Avoiding Insufficient Transport Layer Protection

- Protect with appropriate mechanisms
  - Use TLS on all connections with sensitive data
  - Individually encrypt messages before transmission
    - E.g., XML-Encryption
  - Sign messages before transmission
    - E.g., XML-Signature
- Use the mechanisms correctly
  - Use standard strong algorithms (disable old SSL algorithms)
  - Manage keys/certificates properly
  - Verify SSL certificates before using them
  - Use proven mechanisms when sufficient
    - E.g., SSL vs. XML-Encryption
- See: http://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet for more details

Summary: How do you address these problems?

- Develop Secure Code
  - Follow the best practices in OWASP's Guide to Building Secure Web Applications
    - http://www.owasp.org/index.php/Guide
  - Use OWASP's Application Security Verification Standard as a guide to what an application needs to be secure
    - http://www.owasp.org/index.php/ASVS
  - Use standard security components that are a fit for your organization
    - Use OWASP's ESAPI as a basis for your standard components
    - http://www.owasp.org/index.php/ESAPI
- Review Your Applications
  - Have an expert team review your applications
  - Review your applications yourselves following OWASP Guidelines
    - OWASP Code Review Guide: http://www.owasp.org/index.php/Code_Review_Guide
    - OWASP Testing Guide: http://www.owasp.org/index.php/Testing_Guide

OWASP (ESAPI)



ESAPI Homepage: http://www.owasp.org/index.php/ESAPI