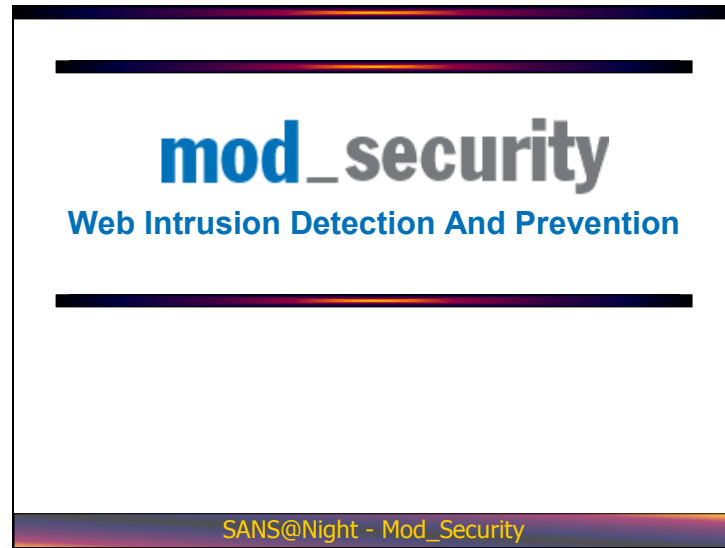


Slide 1



Author: Ryan C. Barnett

Presentation: Mod\_Security – An Intrusion Prevention module for Apache

Email: [RCBarnett@hushmail.com](mailto:RCBarnett@hushmail.com)

Date: Dec. 4th, 2003

Copyright © 2003 Ryan C. Barnett  
All Rights Reserved

## Who Am I?

- Center for Internet Security's Apache Benchmark Project Team Leader
- Web Application Security Consortium (WASC) Member
- Member of SANS Top 20 Vulnerabilities Team
- SANS Instructor – Securing Apache
  - Intrusion Analyst (GCIA)
  - Forensic Analyst (GCFA)
  - Incident Handler (GCIH)
  - Unix Security (GCUX)
  - Security Essentials (GSEC)
- Incident Response Team Member

SANS@Night - Mod\_Security

This page intentionally left blank.

## What Will This Presentation Cover?


- Why current network security strategies fail to protect the web tier
- Why Firewalls, NIDS and HIDS fails
- Introducing Mod\_Security
- Whisker vs. Mod\_Security
  - Common web attacks with Mod\_Security countermeasures
- Real Examples

SANS@Night - Mod\_Security

This page intentionally left blank.

### Updated Class Slides Available

- SANS has quarterly updates for course content
- Unfortunately, Whitehat/Blackhat tools and tactics are NOT on this schedule!
- I am constantly updating the live presentation to provide current info
- Class participants can download updated PDF slides

[http://apachebenchmark.sourceforge.net/Mod\\_Security.zip](http://apachebenchmark.sourceforge.net/Mod_Security.zip)

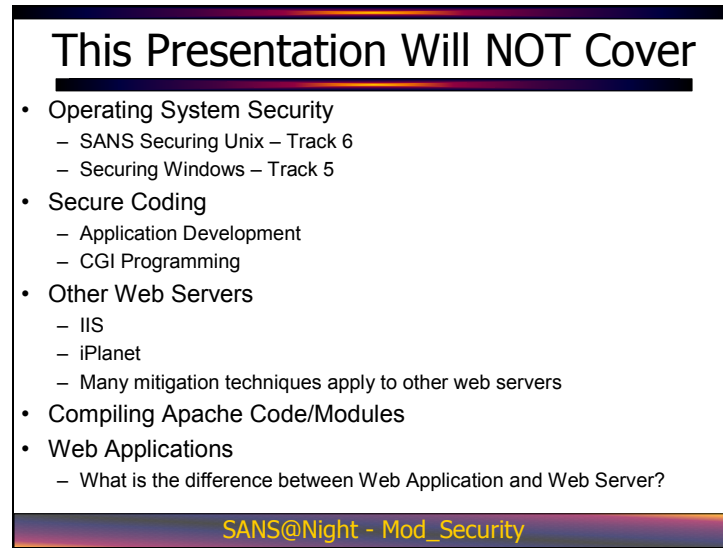
This page intentionally left blank.

## What Will This Presentation Cover?

- Mixed Audience
  - Technical – Web Admins/Security Admins
  - Management – Information Security Officers
- Basic Knowledge of Unix and Web Administration
  - HTTP – Web Servers
- Focus on Apache/Unix Servers (RedHat for Examples)
- Discuss many web security strategies
- Dragnet Approach
  - Examples ARE real – Names/IPs have been changed or removed
- Ask Questions
  - If you don't understand an issue ask – This is YOUR class
  - Q&A sessions before/after breaks

SANS@Night - Mod\_Security

This page intentionally left blank.

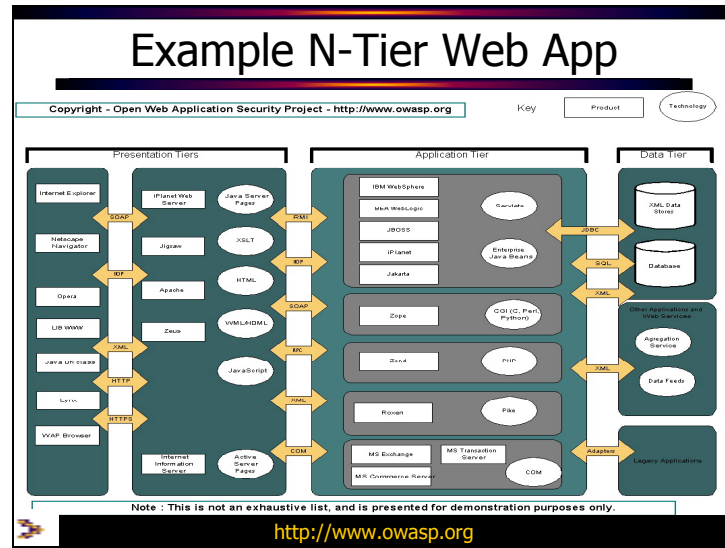
A presentation slide with a black border and a white background. The title "This Presentation Will NOT Cover" is at the top in a large, bold, black font, underlined. Below the title is a bulleted list of topics that will not be covered. The list includes "Operating System Security" (with sub-points "SANS Securing Unix – Track 6" and "Securing Windows – Track 5"), "Secure Coding" (with sub-points "Application Development" and "CGI Programming"), "Other Web Servers" (with sub-points "IIS", "iPlanet", and "Many mitigation techniques apply to other web servers"), "Compiling Apache Code/Modules", and "Web Applications" (with sub-point "What is the difference between Web Application and Web Server?"). At the bottom of the slide, there is a footer in a yellow font that reads "SANS@Night - Mod\_Security".

## This Presentation Will NOT Cover

- Operating System Security
  - SANS Securing Unix – Track 6
  - Securing Windows – Track 5
- Secure Coding
  - Application Development
  - CGI Programming
- Other Web Servers
  - IIS
  - iPlanet
  - Many mitigation techniques apply to other web servers
- Compiling Apache Code/Modules
- Web Applications
  - What is the difference between Web Application and Web Server?

SANS@Night - Mod\_Security

This page intentionally left blank.




In essence a Web Application is a client/server software application that interacts with users or other systems using HTTP. For a user the client would most likely be a web browser like Internet Explorer or Netscape Navigator; for another software application this would be an HTTP user agent that acts as an automated browser. The end user views web pages and is able to interact by sending choices to and from the system. The functions performed can range from relatively simple tasks like searching a local directory for a file or reference, to highly sophisticated applications that perform real-time sales and inventory management across multiple vendors, including both Business to Business and Business to Consumer e-commerce, workflow and supply chain management, and legacy applications. The technology behind web applications has developed at the speed of light. Traditionally simple applications were built with a common gateway interface application (CGI) typically running on the web server itself and often connecting to a simple database (again often on the same host). Modern applications typically are written in Java (or similar languages) and run on distributed application servers, connecting to multiple data sources through complex business logic tiers.

There is a lot of confusion about what a web application actually consists of. While it is true that the problems so often discovered and reported are product specific, they are really logic and design flaws in the application logic, and not necessarily flaws in the underlying web products themselves.

For this class, we will be focusing on the Presentation layer. Presentation Tiers are responsible for presenting the data to the end user or system. The web server serves up data and the web browser renders it into a readable form, which the user can then interpret. It also allows the user to interact by sending back parameters, which the web server can pass along to the application. This "Presentation Tier" includes web servers like Apache and Internet Information Server and web browsers like Internet Explorer and Netscape Navigator. It may also include application components that create the page layout.

## Discussing Security Issues

- Most of the security issues will be discussed from both the Attacker's and Defender's perspectives
- Sections from my SANS@Night "Preventing Web Site Defacements" Presentation



SANS@Night - Mod\_Security

**Discussion:**

We will be taking two different approaches to discussing Web Server security issues. We will address each concern by looking from both the Attacker's and the Defender's Perspectives. Many of the demonstrations are taken directly from my "Preventing Web Site Defacements" presentation. I am currently scheduled to give this presentation during SANS@Night at many of the upcoming SANS Security Conferences. If you are interested in learning more about Web Defacements, please attend this presentation.





## Security Issue - Attacker

BLACK HAT

- Attacker's Perspective In Green
- Web Site/Server Vulnerability
- What Can Be Exploited
- Scanning
- Attack Methods
- Exploit Examples

SANS@Night - Mod\_Security

**Discussion:**

This slide represents the Attacker's perspective about a particular security issue. When you see this slide, we will be discussing reconnaissance techniques, how the vulnerability can be exploited, as well as, showing exploit examples.



**Security Issue - Defender**

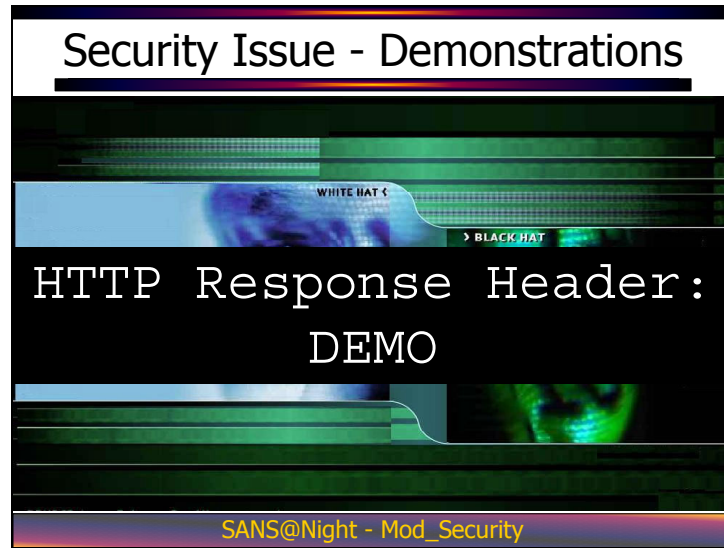
WHITE HAT <

- Defender's Perspective In Blue
- Mod\_Security Countermeasures
- Minimize Vulnerability
- Monitoring
- Identifying
- Alerting

SANS@Night - Mod\_Security

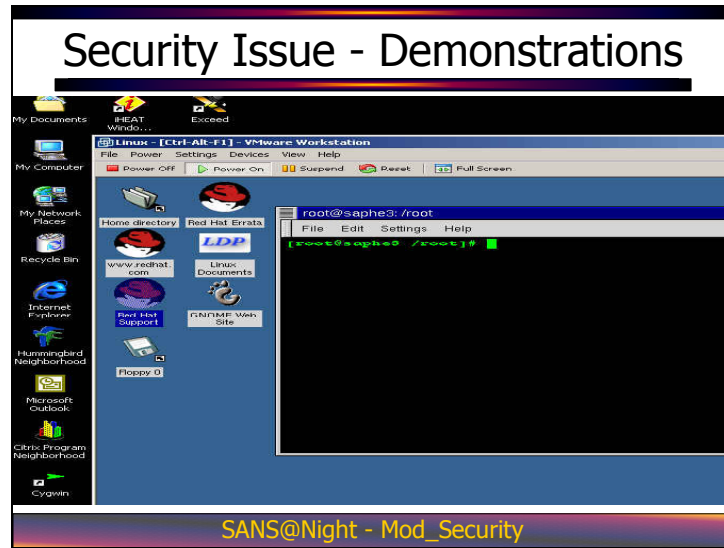
**Discussion:**

This slide will discuss the same security issue, however, this time it will be from the Defender's perspective. We will show how to effectively mitigate any exposures to this attack. Our main goal is to prevent the exploit from being successful, with supplemental goals of at least Identifying and Alerting security personnel of the attack.



**Discussion:**

When we see slides such as this, we will be stepping outside of the PowerPoint presentation to actually show a live demonstration of the particular security issue. I have two different VMWare – RedHat Linux virtual hosts running on my laptop and will use these hosts to demonstrate the different attacks.




**Discussion:**

This slide shows a screen shot of my VMware virtual machine setup. My host OS is Windows 2000 and my Guest OS is RedHat Linux 7.2. I also have another VMware guest OS server running RedHat Linux 6.2. This server will be used to demonstrate various security issues which effect older/mis-configured versions of software.

## Conventions Used

- Blinking Arrow Above Footer
- Web Links to Related Information

<http://www.sans.org/cdieast03/night.php>

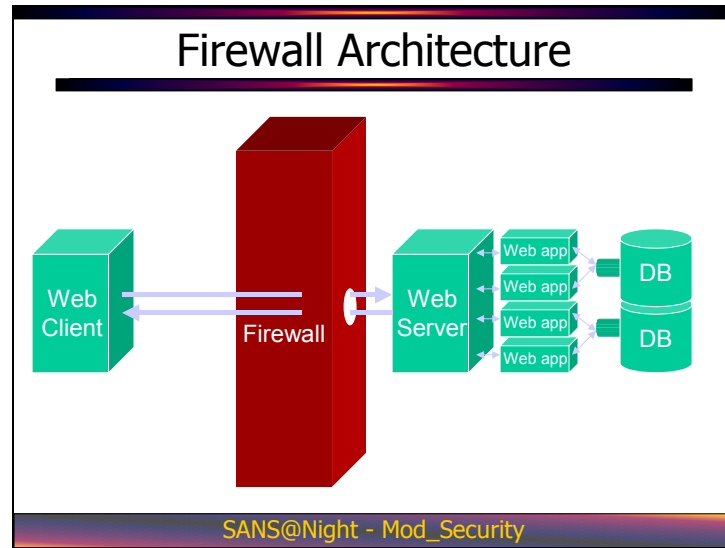
This page intentionally left blank.

## Current Network Security Strategies

- Firewalls
  - Create ACLs based on
    - Originating IP address
    - Destination IP address
    - Destination Port/Service
  - Either allow/deny
- Do not usually inspect (or understand) application level communication
- Port 80 is usually left open by default...

SANS@Night - Mod\_Security

Most firewalls do not, for various reasons, inspect packets at the application layer. They usually rely on packet header inspection and compare these parameters with rules bases. This lack of application layer inspection means that firewalls cannot provide adequate protection for Web Servers. Firewalls are utilized as the main perimeter protection tool, meaning they effectively determine which ports into the corporate network are open or closed. The firewall captures HTTP traffic and, typically, concentrates on analyzing the communication parameters of the traffic. It checks the destination port, the source and destination IP addresses, and similar other attributes. However, a firewall's weakness lies in its inability to verify the data portion (e.g., requests) of the communication consistently. This allows the request to appear legitimate to the firewall. When it arrives at the Web server, it is serviced normally. However, the request may be malicious and exploit a server vulnerability, producing undesired results.



This page intentionally left blank.

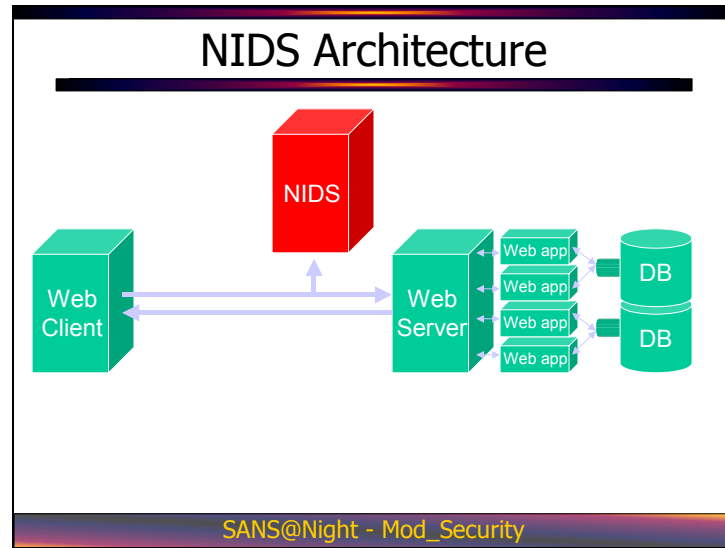
## Current Network Security Strategies

- Network IDS
  - Usually a 3<sup>rd</sup> party sniffing host
  - Not implemented as an application gateway (Snort-Inline, etc...)
  - Identifies and alerts, but does not prevent attacks
  - Since it sniffs packets off the network, it can not inspect HTTP over Secure Socket Layer (SSL)

SANS@Night - Mod\_Security

Intrusion Detection Systems are the next layer of defense in addition to the firewall. They usually only detect network attacks and do not provide real time prevention. Increasing evidence shows that Network IDS (NIDS) products have limited detection capabilities and inherent difficulties properly identifying attack attempts. As a result, many attacks are left undetected, and false positives are generated.





This page intentionally left blank.

## Current Network Security Strategies

- Host IDS
  - Monitors local activity on a system
  - Usually one of the following:
    - Checks local files for changes by creating a hash database (Tripwire, Aide, etc...)
    - An agent which monitors processes and/or log files for signs of illegal user/daemon activity (Brute Forcing logins, Buffer Overflow Attacks)
  - Technically, a program such as SWATCH can monitor the web server's log files, but it will not prevent these attacks

```

HIDS Strategies

# ./tripwire
Tripwire(tm) ASR (Academic Source Release) 1.3.1
File Integrity Assessment Software
(c) 1992, Purdue Research Foundation, (c) 1997, 1999 Tripwire
Security Systems, Inc. All Rights Reserved. Use Restricted to
Authorized Licensees.
### Phase 1: Reading configuration file
### Phase 2: Generating file list
### Phase 3: Creating file information database
### Phase 4: Searching for inconsistencies
###
###          Total files scanned:      10746
###          Files added:              1
###          Files deleted:            0
###          Files changed:            0
###
###          Total file violations:     1
###
added: -rwxr--r-- root    2198 Sep 24 04:13:05 2001 /etc/init.d/sendmail.old
SANS@Night - Mod_Security
```

**Discussion:**

We will be discussing many of these common web vulnerabilities during the security configuration steps of this presentation.

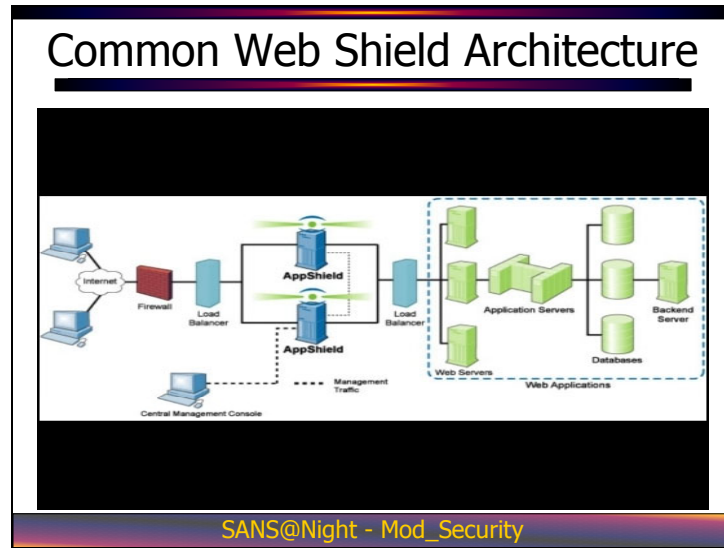
## New Breed of Web Tools

- Application Level Firewalls
  - Raptor
- Web Shields
  - AppShield (Sanctum)
  - InterDo (KaVaDo)
  - Ubizen DMZ/Shield (Ubizen)
  - McAfee Intercept
  - PitBull

SANS@Night - Mod\_Security

**Discussion:**

We will be discussing many of these common web vulnerabilities during the security configuration steps of this presentation.



**Discussion:**

We will be discussing many of these common web vulnerabilities during the security configuration steps of this presentation.

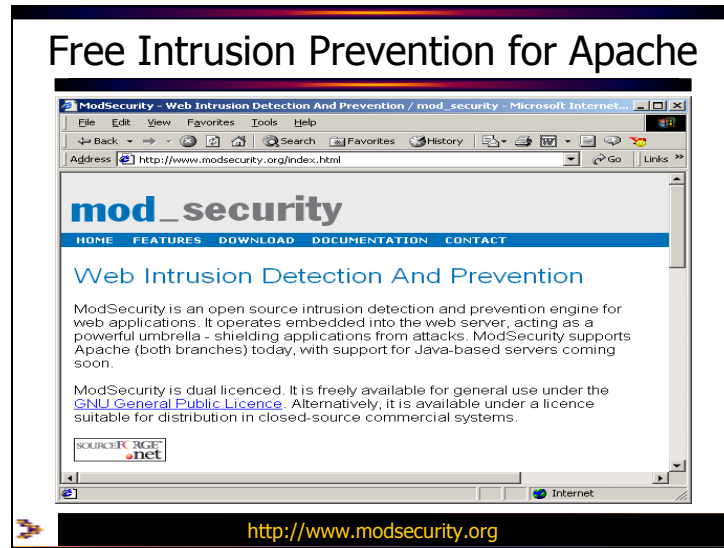
## Pros and Cons

- Pros
  - Commercial Apps have easy installation
  - Correlating data and alerts
  - Able to prevent attacks since it proxies the http requests
  - Able to inspect SSL traffic since it will decrypt it first
- Cons
  - Can not detect absolutely everything (0-Day Exploits?)
  - Can not always update attack signatures quickly
  - \$\$\$

SANS@Night - Mod\_Security

**Discussion:**

We will be discussing many of these common web vulnerabilities during the security configuration steps of this presentation.



**Discussion:**

We will be discussing many of these common web vulnerabilities during the security configuration steps of this presentation.

## Mod\_Security: Features

- **Request filtering**; incoming requests are analyzed as they come in, and before they get handled by the web server or other modules.
- **Anti-evasion techniques**; paths and parameters are normalized before analysis takes place in order to fight evasion techniques.
- **Understanding of the HTTP protocol**; since the engine understands HTTP, it performs very specific and fine granulated filtering.
- **POST payload analysis**; the engine will intercept the contents transmitted using the POST method, too.
- **Audit logging**; full details of every request (including POST) can be logged for later analysis.
- **HTTPS filtering**; since the engine is embedded in the web server, it gets access to request data after decryption takes place.

SANS@Night - Mod\_Security

**Discussion:**

We will be discussing many of these common web vulnerabilities during the security configuration steps of this presentation.



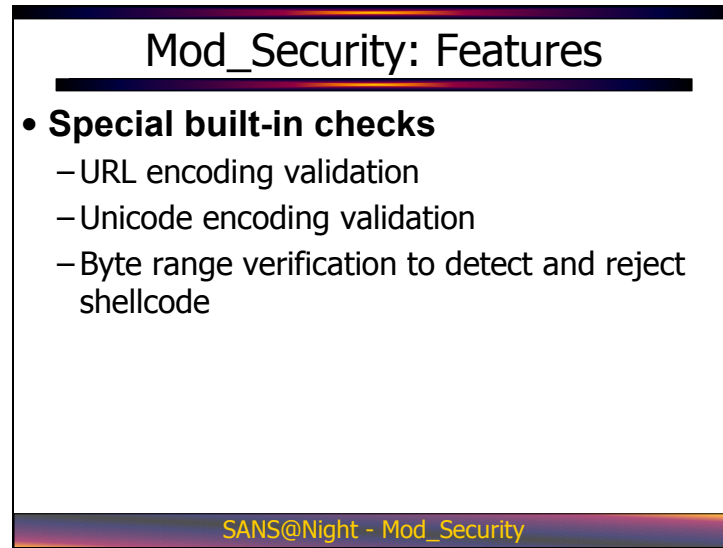
## Mod\_Security: Features

- **Anti-evasion techniques**
  - Remove multiple forward slash characters
  - Treat backslash and forward slash characters equally (Windows only)
  - Remove directory self-references
  - Detect and remove null-bytes (%00)
  - Decode URL encoded characters

SANS@Night - Mod\_Security

**Discussion:**

We will be discussing many of these common web vulnerabilities during the security configuration steps of this presentation.

A presentation slide titled "Mod\_Security: Features" with a decorative header and footer. The header has a black background with a rainbow gradient. The footer has a white background with a rainbow gradient. The main content area is white with a black border.

## Mod\_Security: Features

- **Special built-in checks**
  - URL encoding validation
  - Unicode encoding validation
  - Byte range verification to detect and reject shellcode

SANS@Night - Mod\_Security

**Discussion:**

We will be discussing many of these common web vulnerabilities during the security configuration steps of this presentation.

## Slide 27

Byte Range Chart: 32-125											
<div></div>											
00	0	48	a	96	.	144	A	192	a	240	
01	1	49	b	97	,	145	A	193	A	241	
02	2	50	b	98		146	A	194	B	242	
03	3	51	c	99	'	147	A	195	C	243	
04	4	52	d	100	"	148	A	196	D	244	
05	5	53	e	101	~	149	A	197	E	245	
06	6	54	f	102	*	150	A	198	F	246	
07	7	55	g	103	-	151	C	199	G	247	
08	8	56	h	104	_	152	E	200	H	248	
09	9	57	i	105	~	153	E	201	I	249	
10	:	58	j	106	~	154	E	202	J	250	
11	;	59	k	107	~	155	E	203	K	251	
12	<	60	l	108	~	156	I	204	L	252	
13	=	61	m	109	~	157	I	205	M	253	
14	>	62	n	110	~	158	I	206	N	254	
15	?	63	o	111	Y	159	I	207	O	255	
16	@	64	p	112	Y	160	D	208			
17	A	65	q	113	Y	161	O	209			
18	B	66	r	114	Y	162	O	210			
19	C	67	s	115	E	163	O	211			
20	D	68	t	116	E	164	O	212			
21	E	69	u	117	Y	165	O	213			
22	F	70	v	118	Y	166	O	214			
23	G	71	w	119	Y	167		215			
24	H	72	x	120	Y	168	O	216			
25	I	73	y	121	@	169	O	217			
26	J	74	z	122	@	170	U	218			
27	K	75	[	123	@	171	U	219			
28	L	76	]	124	~	172	U	220			
29	M	77	^	125	~	173	Y	221			
30	N	78	~	126	~	174	Y	222			
31	O	79		127	~	175	B	223			
32	P	80	€	128	~	176	A	224			
33	Q	81	€	129	~	177	A	225			
34	R	82	€	130	~	178	A	226			
35	S	83	€	131	~	179	A	227			
36	T	84	€	132	~	180	A	228			
37	U	85	€	133	~	181	A	229			
38	V	86	€	134	~	182	U	230			
39	W	87	€	135	~	183	C	231			
40	X	88	€	136	~	184	C	232			
41	Y	89	€	137	~	185	C	233			
42	Z	90	€	138	~	186	A	234			
43	[	91	€	139	~	187	E	235			
44	\	92	€	140	~	188	E	236			
45	]	93	€	141	~	189	I	237			
46	^	94	€	142	~	190	I	238			
47	_	95	€	143	~	191	I	239			

SANS@Night - Mod\_Security

### Discussion:

This slide shows an example ASCII code chart. In order to read the chart, you can start in the second column and go down to the number “32” and then moved directly to the left to the corresponding entry in the first column – “space”. The number 33 = !, 65 = A, etc...

## Mod\_Security: Features

- **Rules**
  - Any number of custom rules supported
  - Rules are formed using regular expressions
  - Negated rules supported
  - Each container (VirtualHost, Location, ...) can have different configuration
  - Analyzes headers
  - Analyzes individual cookies
  - Analyzes environment variables
  - Analyzes server variables
  - Analyzes individual page variables
  - Analyzes POST payload
  - Analyzes script output

SANS@Night - Mod\_Security

**Discussion:**

We will be discussing many of these common web vulnerabilities during the security configuration steps of this presentation.

## Mod\_Security: Features

- **Actions**
  - Reject request with status code
  - Reject request with redirection
  - Execute external binary on rule match
  - Log request
  - Stop rule processing and let the request through
  - Rule chaining
  - Skip next x number of rules on match
  - Pauses for a number of milliseconds

SANS@Night - Mod\_Security

**Discussion:**

We will be discussing many of these common web vulnerabilities during the security configuration steps of this presentation.

## Mod\_Security: Features

- **Change the identity of the web server**
- **Easy to use internal chroot functionality**
- **Audit log to log complete requests**
- **Debug log**
- **Smart enough to apply rules only to dynamic resources**

SANS@Night - Mod\_Security

**Discussion:**

We will be discussing many of these common web vulnerabilities during the security configuration steps of this presentation.

### Mod\_Security Benefits

- Provides Intrusion Prevention Gateway for HTTP if used as a Reverse Proxy Server
  - Malicious requests are not passed to production hosts
- Can inspect ANY HTTP Client Request Header
- Uses Regular Expressions for rules
- Can quickly create new rules/filters to protect Apache until patches are released
  - Chunking Exploit

```
SecFilterSelective HTTP_TRANSFER_ENCODING "chunked"
```

SANS@Night - Mod\_Security

#### Discussion:

Mod\_Security gives us many important features. One of the most important features is that it gives us the ability to quickly implement new Filters to protect our Apache web server (or any internal web server if we are using Apache as a Proxy Server!) when new exploits are released. For example, when the Apache Chunking exploit was released, it included information about what parameters the client could send to exploit this vulnerability. Armed with this information, we can now edit the httpd.conf file and add in new filters to protect our Apache servers until a new patch is available.

If we add in the following Mod\_Security directive:

**SecFilterSelective HTTP\_TRANSFER\_ENCODING "chunked"**

We can deny any client requests that try to exploit this vulnerability. After implementing this directive and trying to exploit this vulnerability, Mod\_Security logs the attack in the error\_log with the following entry:

**[Sun Mar 30 18:27:29 2003] [error] [client 127.0.0.1] mod\_security: Access denied with code 403. Pattern match "chunked" at HEADER.**

## Installing Mod\_Security

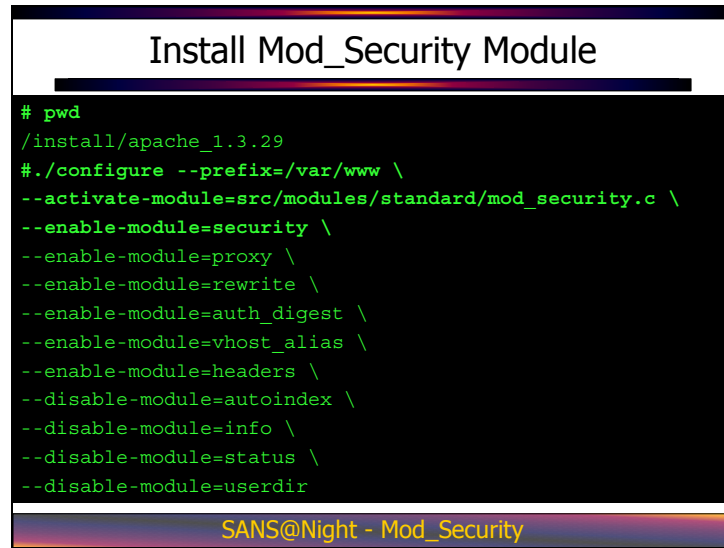
- **Download archive**
  - [http://www.modsecurity.org/download/mod\\_security-1.7.2.tar.gz](http://www.modsecurity.org/download/mod_security-1.7.2.tar.gz)
- **Place the mod\_security.c file into your Apache modules directory**
  - /path/to/apache\_1.3.29/src/modules/standard/
- **Compile Apache with mod\_security**
- **Update the httpd.conf file**

SANS@Night - Mod\_Security

**Discussion:**

We will be discussing many of these common web vulnerabilities during the security configuration steps of this presentation.



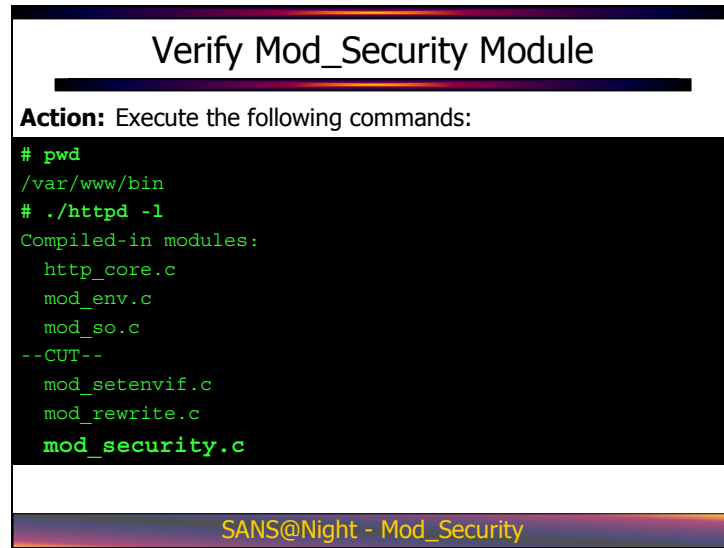


```
# pwd
/install/apache_1.3.29
#./configure --prefix=/var/www \
--activate-module=src/modules/standard/mod_security.c \
--enable-module=security \
--enable-module=proxy \
--enable-module=rewrite \
--enable-module=auth_digest \
--enable-module=vhost_alias \
--enable-module=headers \
--disable-module=autoindex \
--disable-module=info \
--disable-module=status \
--disable-module=userdir
```

SANS@Night - Mod\_Security

**Discussion:**

After compiling the Apache source to include the mod\_security module, we execute the httpd binary with the “-l” flag to display which modules are compiled into the binary.



The image is a terminal window with a black background and green text. At the top, the title 'Verify Mod\_Security Module' is centered in a white box. Below the title, the text 'Action: Execute the following commands:' is displayed. The terminal shows the execution of two commands: '# pwd' which returns '/var/www/bin', and '# ./httpd -l' which lists compiled-in modules. The modules listed are http\_core.c, mod\_env.c, mod\_so.c, followed by a separator '--CUT--', and then mod\_setenvif.c, mod\_rewrite.c, and mod\_security.c. At the bottom of the terminal window, the text 'SANS@Night - Mod\_Security' is displayed in a yellow font.

```
Verify Mod_Security Module

Action: Execute the following commands:

# pwd
/var/www/bin
# ./httpd -l
Compiled-in modules:
  http_core.c
  mod_env.c
  mod_so.c
--CUT--
  mod_setenvif.c
  mod_rewrite.c
  mod_security.c

SANS@Night - Mod_Security
```

**Discussion:**  
After compiling the Apache source to include the mod\_security module, we execute the httpd binary with the “-l” flag to display which modules are compiled into the binary.

## Hands On: IfModule

**Action:** Edit/Verify the httpd.conf file and check for the following lines:

```
<IfModule mod_security.c>
SecFilterEngine On
SecAuditEngine On
SecAuditLog logs/audit_log
SecFilterDebugLog logs/modsec_debug_log
SecFilterDebugLevel 3
SecFilterDefaultAction deny,log,status:403
SecFilter "\.\/"
SecFilter "<(.\|n)+>"
SecFilterSelective "HTTP_USER_AGENT|HTTP_HOST" "^$"
SecFilterSelective POST_PAYLOAD "!image/(jpeg|bmp|gif)"
-- CUT --
</IfModule>
```

SANS@Night - Mod\_Security

### Discussion:

We will not give a brief explanation of these Mod\_Security directives:

**SecFilterEngine On** - Turn the Mod\_Security filtering engine On or Off

**SecAuditEngine On** - The audit engine works independently and can be turned On or Off on the per-server or on the per-directory basis

**SecAuditLog logs/audit\_log** - The name and location of the audit log file

**SecFilterDebugLog logs/modsec\_debug\_log** - This is the new debugging log file. This file is useful when testing new Mod\_Security rulesets.

**SecFilterDebugLevel 4** - This sets the debug level of Mod\_Security. Levels are 1 –9.

**SecFilterDefaultAction deny,log,status:403** - This line specifies the default action to take when Mod\_Security finds a match. This line translates to: Deny the connection attempt, log the connection information and return a 403 – Forbidden status code to the client. This feature is interesting in that you can actually specify any HTTP status code you wish. This behavior can wreak havoc on Vulnerability Scanners!

**SecFilter "\.\/"** - Prevents directory traversal attacks.

**SecFilter "<(.\|n)+>"** - Prevent XSS attacks (HTML/Javascript injection)

**SecFilterSelective "HTTP\_USER\_AGENT|HTTP\_HOST" "^\$"** - Require HTTP\_USER\_AGENT and HTTP\_HOST headers

**SecFilterSelective POST\_PAYLOAD "!image/(jpeg|bmp|gif)"** - When allowing file uploads, only allow images. Note that this is not foolproof, a determined attacker could get around this by inserting hidden html tags.

Mod\_Security + Snort Signatures = Mod\_Snort?

- **Snort has several Web Attack Signature files**
  - web-attack-responses.rules
  - web-client.rules
  - web-iis.rules
  - web-attacks.rules
  - web-coldfusion.rules
  - web-misc.rules
  - web-cgi.rules
  - web-frontpage.rules
  - web-php.rules
- **Can't we use these signatures somehow???**


SANS@Night - Mod\_Security

**Discussion:**

We will be discussing many of these common web vulnerabilities during the security configuration steps of this presentation.

### Snort2modsec.pl

- I wanted a script that would translate snort's web attack files into mod\_security's syntax
- This would help to automate the updating of IDS rules into mod\_security
- I initially found a PERL script for the Zeus web server that basically did what I needed: sniff-snort.pl
- Mod\_Security creator then updated it for our use
- ```
./snort2modsec.pl web-attacks.rules  
> mod_security.rules
```

 <http://www.modsecurity.org/documentation/snort2modsec.pl>

**Discussion:**

We will be discussing many of these common web vulnerabilities during the security configuration steps of this presentation.



#### Discussion:

We want to leverage the outstanding Snort signature files, which are continually updated by the dedicated Snort user community, by implementing these into our Mod\_Security module. We can rely on new Snort signature files to supply us with updated Web Attack signatures. This screen shows the normal Snort attack signature file entry format. We are interested in the “uricontent” portion of the signatures. We will need to extract these content sections from the Snort signatures and translate them into our Mod\_Security format. After extracting the appropriate information from the various “WEB” attack Snort signature files, we have a Mod\_Security signature file with **373** signatures!

**Discussion:**

These entries use Regular Expression matching. **CAUTION** – You might have to escape meta-characters or else Apache will complain and might SegFault. For instance, the final entry on the slide shows how I had to escape the “+” signs with back-slashes. The URL Requests will be inspected, and if the alert signature between the “” is found anywhere in the request it is acted upon. The SecFilter directive tells Mod\_Security that you want to look for the text strings within the the URL Request line only. If you want to create attack signatures for other portions of the client request headers, you will need to use the SecFilterSelective Variable names are the same as in mod\_rewrite. Some variables have special prefixes that tell the module where to look for values but prefixes are stripped before looking variables up:

HTTP\_header for headers

ENV\_variable for environment

ARG\_variable for URL argument

Variables special to mod\_security are:

# ARGS - filter arguments, either QUERY\_STRING or POST\_PAYLOAD

# ARGS\_NAMES - variable names only

# ARGS\_VALUES - variable values only

# POST\_PAYLOAD - as the name says

# Other supported variables are:

REMOTE\_ADDR

REMOTE\_HOST

REMOTE\_USER

REMOTE\_IDENT

REQUEST\_METHOD

SCRIPT\_FILENAME

```
Example Mod_Security AuditLog Entry

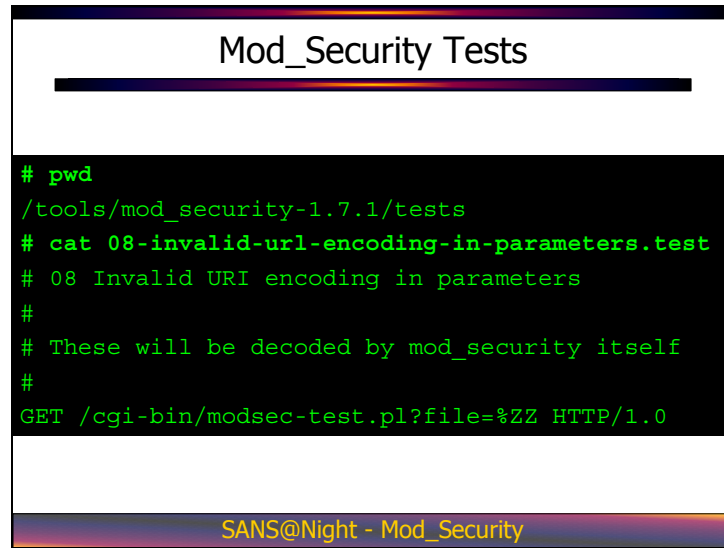
# tail /var/www/logs/error_log
[Sun Mar 23 18:35:16 2003] [error] [client 192.168.1.100] mod_security:
Access denied with code 403. Pattern match "/store.cgi" at THE_REQUEST.
# tail -17 /var/www/logs/audit_log
=====
Request: 192.168.1.100 - - [Sun Mar 23 17:59:26 2003] "GET /cgi-bin/
printenv HTTP/1.1" 200 1043
Handler: cgi-script
-----
GET /cgi-bin/printenv HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application
/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword, */*
Accept-Encoding: gzip, deflate
Accept-Language: en-us
Connection: Keep-Alive
Host: 192.168.1.101
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)

SANS@Night - Mod_Security
```

**Discussion:**

This slide shows an excerpt from the Mod\_Security audit\_log file. If configured, Mod\_Security will capture and store both the client and server headers and the data payload of dynamic content requests – I.E.- CGI scripts and POST requests. This data can prove to be tremendously useful when investigating Web Attacks.





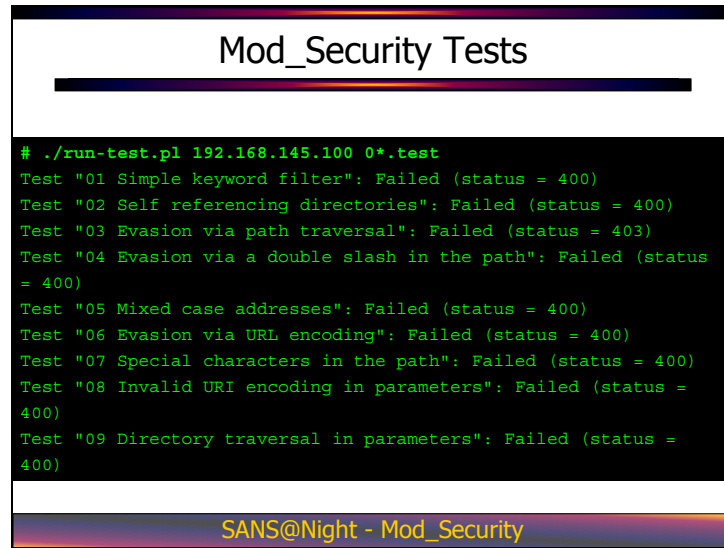
```
Mod_Security Tests

# pwd
/tools/mod_security-1.7.1/tests
# cat 08-invalid-url-encoding-in-parameters.test
# 08 Invalid URI encoding in parameters
#
# These will be decoded by mod_security itself
#
GET /cgi-bin/modsec-test.pl?file=%ZZ HTTP/1.0
```

SANS@Night - Mod\_Security

**Discussion:**

This slide shows an excerpt from the Mod\_Security audit\_log file. If configured, Mod\_Security will capture and store both the client and server headers and the data payload of dynamic content requests – I.E.- CGI scripts and POST requests. This data can prove to be tremendously useful when investigating Web Attacks.



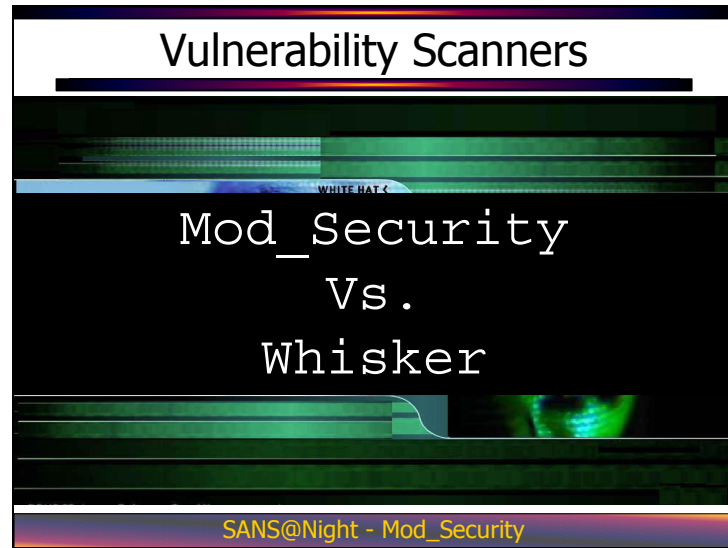
```
# ./run-test.pl 192.168.145.100 0*.test
Test "01 Simple keyword filter": Failed (status = 400)
Test "02 Self referencing directories": Failed (status = 400)
Test "03 Evasion via path traversal": Failed (status = 403)
Test "04 Evasion via a double slash in the path": Failed (status = 400)
Test "05 Mixed case addresses": Failed (status = 400)
Test "06 Evasion via URL encoding": Failed (status = 400)
Test "07 Special characters in the path": Failed (status = 400)
Test "08 Invalid URI encoding in parameters": Failed (status = 400)
Test "09 Directory traversal in parameters": Failed (status = 400)
```

SANS@Night - Mod\_Security

**Discussion:**


This slide shows an excerpt from the Mod\_Security audit\_log file. If configured, Mod\_Security will capture and store both the client and server headers and the data payload of dynamic content requests – I.E.- CGI scripts and POST requests. This data can prove to be tremendously useful when investigating Web Attacks.

Slide 43




This space intentionally left blank.

## Vulnerability Scanners



BLACK HAT

- Applications created to automate the process of scanning hosts for known vulnerabilities
  - Whisker
  - Nessus
  - ISS Security Scanner
  - SATAN
- Originally created by Security Admins to assist with securing their own networks and hosts
- Hacker community uses these same tools to conduct their own "Security Audits"



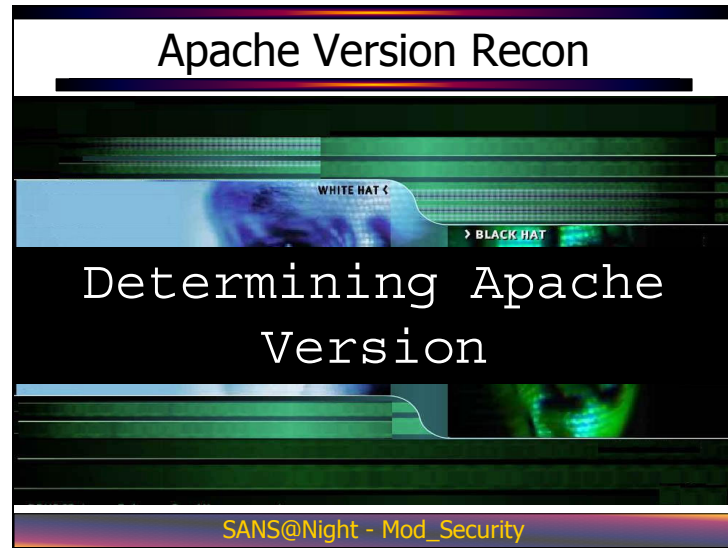
WHISKER

<http://packetstormsecurity.org.pk/papers/IDS/whiskerids.html>

**Discussion:**

Vulnerability Scanners are applications that were originally created to assist security personnel with auditing their own hosts and networks. It didn't take long, however, for the hacker community to get their hands on these same tools and conduct "Security Audits" of their own. These tools are designed to access a selected host(s) and look for a set list of vulnerabilities. The best of these scanners are: NESSUS, ISS Security Scanner and Whisker.

Slide 45



This space intentionally left blank.



**Discussion:**

By connecting to port 80 on the target host using either telnet or netcat, an attacker can issue an HTTP "HEAD" request to identify the web server software from the response header. In the demo connection above, the "Server:" line shows that this host is using Apache/1.3.12. With this information, an attacker can search various hacker sites for any known vulnerabilities or exploits for this version of web server software.

### Example – Whisker's Light Fingerprinting

```
# now do some light fingerprinting...
-- CUT --
my $Aflag=0;
$req{whisker}->{uri}='//';
if(!_do_request(\%req,%G_RESP)){
    _d_response(\%G_RESP);
    if($G_RESP{whisker}->{code}==200){
        $req{whisker}->{uri}='%2f';
        if(!_do_request(\%req,%G_RESP)){
            _d_response(\%G_RESP);
            $Aflag++ if($G_RESP{whisker}-
>{code}==404);
        }
    }
}

m_re_banner('Apache',$Aflag);
```

SANS@Night - Mod\_Security

**Discussion:**

This slide shows some sample code taken from Whisker 2.1's main.test file. This file runs specific “pre-test” to help set up the remainder of the scans. Here is some comments from within the main.test file:

```
#
# The preserver test is very special and important--first, it handles
# the checking for custom 404 pages, and configures whisker to
# respond appropriately (the values it sets are used by the internal
# d_response in order to manipulate the result). Secondly, it
# fingerprints the server and adjusts the server banner accordingly,
# which could change how all future tests are ran: hence, it needs
# to be the first thing ran.
#
```

The section of code in the slide runs two tests to determine if the target web server is in fact an Apache server, regardless of what the Banner might report. The first request is a “GET //” and if the HTTP Status Code is a 200, then the next request is sent. The second request is “GET/%2f”, which is URI Encoded – and translates to “GET //”. This time Apache returns a 404 – Not Found error code. Other web servers – IIS – do not return the same status codes for these requests.

### Example – Whisker's Light Fingerprinting

```
-----
Title: Server banner
Id: 100
Severity: Informational

The server returned the following banner:
Microsoft-IIS/4.0
-----

Title: Alternate server type
Id: 103
Severity: Informational

Testing has identified the server might be a 'Apache' server. This
Change could be due to the server not correctly identifying itself (the
Admins changed the banner). Tests will now check for this server type
as well as the previously identified server types.
-----
```

SANS@Night - Mod\_Security

**Discussion:**

We have just ran a whisker scan and it is telling us, based on the pre-tests that this web server may in fact be an Apache server. Not only does this alert the attacker, but Whisker will also add in all of the Apache tests during its scan.



## Alter the Apache Banner

- Traditionally, in order to modify the HTTP Response "Server" token you either had to:
  - Edit the uncompiled source code, or
  - Edit the compiled binary with an editor
- Mod\_Security has a directive so you can set this in the httpd.conf file
- Changing the Banner does **NOT** make your system more secure!
  - If your Apache version is vulnerable to an exploit, changing the banner will not fix this
  - It will, however, confuse Script Kiddies and automated worms that trigger on the banner

WHITE HAT

SANS@Night - Mod\_Security

**Discussion:**

It is possible to edit out and/or alter (for deception purposes) the "Server" field information displayed by a web server's response headers. In order to accomplish this task, the web server configuration file that contains the server version information must be edited.

**IMPORTANT:**


There has been much debate in Apache circles as to the amount of protection that can be gained by changing the http Server: token information. While altering the banner info alone, and not taking any other steps to hide the software version, probably doesn't provide much protection from REAL people who are actively conducting reconnaissance, it does help with regards to blocking automated WORM programs. Due to the increase in popularity of using worms to mass infect systems, this method of protecting your web servers becomes vital. This step could certainly buy organizations some time during the patching phase when new worms are released into the wild and they are configured to attack systems based on the server token response.

### Apache-ssl-bug.c Code

```
if ( (arch == -1) || (arch >= MAX_ARCH) ){
    DEBUG("Checking version");
    if (strncmp(a,"Apache",6)){
        printf("The web server is not Apache\n\n");
        return 1;
    }
}
*****
# ./apache-ssl-bug -t 0 192.168.145.100

Apache & OpenSSL 0.9.6 Exploit
Made by andy^ after the bugtraq.c worm

Trying to exploit 192.168.145.100
The web server is not Apache
FAILED
```

 <http://packetstormsecurity.nl/0209-exploits/apache-ssl-bug.c>

**Discussion:**

This slide shows some excerpts from the apache-ssl-bug source code, which was found on the Packetstorm Web Site. This section of code shows where the Worm actually inspects the response headers from the target web server and if it does not contain the string “Apache” then it exits. I then show an example of running the tool.

## Edit the Server Banner Code

- Change the banner information provided in response to a HTTP requests
- The server banner can help an attacker to determine what exploits will work against your server version
- What should we change it to?
  - IIS
  - iPlanet
- Why not just remove the Server: header?
  - No server banner = Apache version or ServerMask
- Set value in httpd.conf  
`SecServerSignature "My-Server/1.0"`

SANS@Night - Mod\_Security

### Discussion:

We want to remove/modify the default HTTP Response Header parameter for the “Server:” token to hide the identity of our web server software. In order to exploit any potential weaknesses of a web server, an attacker must first identify the target’s web server software. By connecting to port 80 on the target host using either telnet or netcat, an attacker can issue an HTTP “HEAD” request to identify the web server software from the response header.

**IMPORTANT** – Consult your legal department for verification of legal issues involved with changing your web server HTTP Response Header info. Private Sector and Government Agencies have different standards concerning public release of information. It has been debated in legal circles if altering this parameter constitutes “Lying” to the public about which web server software you are using. You should refer to the Request for Comment (RFC #2616) for HTTP 1.1 protocol and read the following section concerning the SERVER header info:

### 14.38 Server

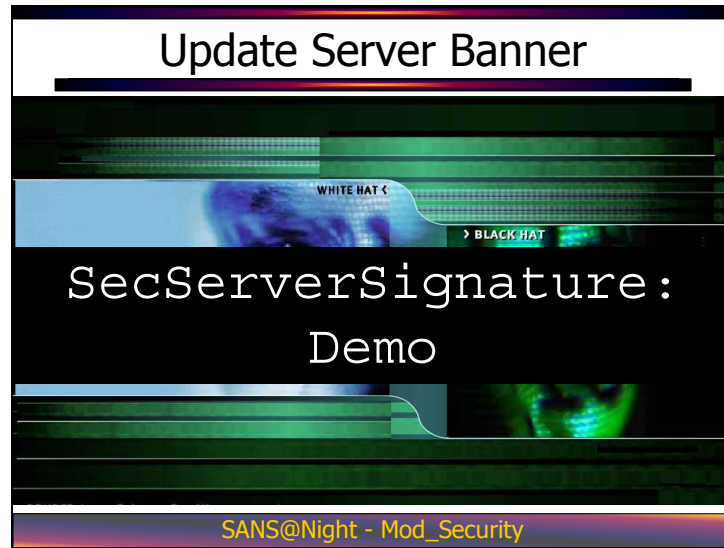
The Server response-header field contains information about the software used by the origin server to handle the request. The field can contain multiple product tokens (section 3.8) and comments identifying the server and any significant subproducts. The product tokens are listed in order of their significance for identifying the application. Server = "Server" ":" 1\*( product | comment )

Example: Server: CERN/3.0 libwww/2.17

If the response is being forwarded through a proxy, the proxy application **MUST NOT** modify the Server response-header. Instead, it **SHOULD** include a Via field (as described in section 14.45).

**Note: Revealing the specific software version of the server might allow the server machine to become more vulnerable to attacks against software that is known to contain security holes. Server implementers are encouraged to make this field a configurable option.**

The full RFC 2616 document is located at the World Wide Web Consortium web site. Legal Counsel will want to justify the security vulnerability associated with not altering this information. Most Legal departments should approve this security measure as “Mis-Information” to protect web assets. You can pick the name of any web server software.



**Discussion:**

By connecting to port 80 on the target host using either telnet or netcat, an attacker can issue an HTTP "HEAD" request to identify the web server software from the response header. In the demo connection above, the "Server:" line shows that this host is using Apache/1.3.12. With this information, an attacker can search various hacker sites for any known vulnerabilities or exploits for this version of web server software.

### Hands On: Edit the Server Banner Code

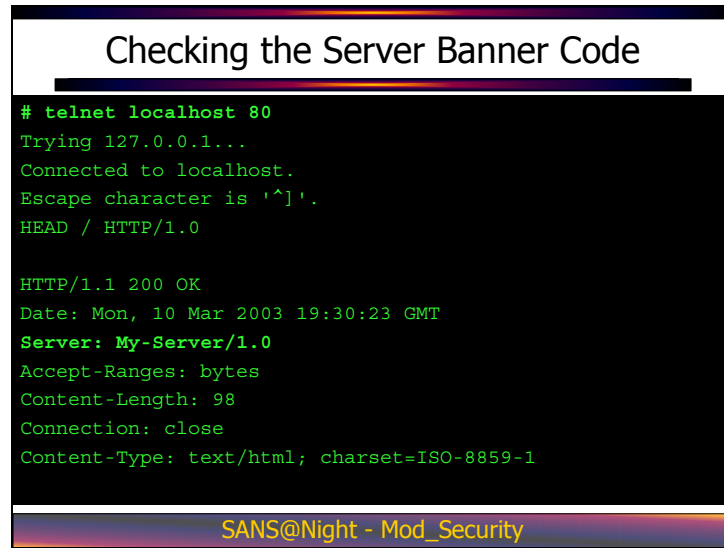
**Action:** Execute the following commands to edit the Server Banner Token Information

```
# pwd
/var/www/conf
# cp httpd.conf httpd.conf.orig
# vi httpd.conf
# diff httpd.conf httpd.conf.orig
229c229
<     SecServerSignature "My-Server/1.0"
---
>
```

SANS@Night - Mod\_Security

**Discussion:**

Before we edit the contents of the httpd.conf file, we make a backup copy in case anything goes wrong with our editing. We then use vi to edit the file and change the SecServerSignature setting to “My-Server/1.0”. We then use diff to show the updated section of code.



```
# telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 10 Mar 2003 19:30:23 GMT
Server: My-Server/1.0
Accept-Ranges: bytes
Content-Length: 98
Connection: close
Content-Type: text/html; charset=ISO-8859-1
```

SANS@Night - Mod\_Security

**Discussion:**

In this slide, we connect to the localhost web server and inspect the new “Server” banner displayed by Mod\_Security.

## Whisker Scanning

- Whisker can do a "smart" scan to reduce the network/log noise associated with kitchen-sink scanning
- It will scan parent directories prior to searching for files
- You can update the main.test file to scan multiple directories
  - /cgi-local, /cgi, /cgibin, etc...

SANS@Night - Mod\_Security

### Discussion:

In order to catch a wider array of vulnerability scans run against our web server, we can use SecFilter directives to list common files and directories which are targeted by tools such as Whisker. This has the benefit of being able to catch vulnerability scanners without filling up your cgi-bin with the fake CGIs. This directive entry can act as another crude IDS database of attack signatures. Remember that the SecFilter directive uses Regular Expressions to match the listings. This provides added flexibility, however, care should be taken when listing files to have the desired effect. If this directive is working properly, Apache will redirect all requests for the files listed here to our 400 CGI script. We will then be notified of this connection attempt and can quickly correlate data to identify what type of vulnerability scan is being run. This technique is extremely effective at identifying attacks from tools such as Whisker, which will check for the existence of a directory prior to requesting files within that directory. Here is an excerpt from the Whisker README file:

1. /cgi-bin is pretty damn common, I'll give you that. But I've also been on many a hosting provider that used /cgi-local. And I've seen people use /cgi, /cgibin, etc. Fact of the matter is that it could also be /~user/cgi-bin, or /~user/cgis, etc. Then there's some scripts that are all over the place, like wwwboard, which may or may not have it's own directory.

**Point of the point: wouldn't it be nice to define multiple directories?**

2. You know what really irks me? Seeing a CGI scanner thrash around through /cgi-bin or whatnot, when /cgi-bin doesn't even exist. Talk about noisy in the logs. Now, if we waste a brain cell, we can see that if we query the /cgi-bin directory (by itself), we'll get a 200 (ok), 403 (forbidden), or 302 (for custom error pages) if it exists, or a 404 if it doesn't. Wow. So if we just do a quick check on /cgi-bin, and get a 404, we can save our however many /cgi-bin CGI checks we were going to make. That could save you 65 entries in the httpd logs.

**Point of the point: save noise/time by querying parent dirs**

By denying access to these directories, we are able to catch stealth scans such as Whisker.

```
Whisker main.test File

sub m_dir_driller{
  my @temp=warray('spider');
  my @general=qw(
    temp    prv    source  backup
    bak     old    db      lib
    inc     include dat     data
    test    save   tmp     archive
  );
  --CUT--
  Whisker scans for common subdirectories in already-existing
  Directories found while spidering the website. The finding of a
  directory does not immediately signal a problem; rather, you
  should go back and review the contents of each found URL to
  determine if there is any sensitive material in those
  directories.
}
```

SANS@Night - Mod\_Security

**Discussion:**

This slide shows a small section of Whisker's main.test file. If you look at the entries listed, they show which files will be searched for on a target host. There are literally hundreds of known vulnerable scripts and files that are available on Web Servers, not just Apache. It is important to note, most Web Vulnerability Scanners function in the same basic way. They all search a web server for a list of files and report back if the files exist. If the web server sends an HTTP status code of 200, then this file does exist. The attacker will then review the results from the vulnerability scan looking for exploitable files/scripts/applications. Once these files are identified, then the attacker will actually conduct an exploit attack.

**Due to the fact that these Vulnerability Scans normally happen during the "Reconnaissance" Phase of a Web Attack, which precedes the "Exploit" Phase, it is crucial that we have some sort of alerting mechanisms in place to identify these scans.**



### Fake Directories - Honeypot Mentality

- Cuts down on data overload normally associated with reviewing Web Server log files looking for malicious attempts
- Virtually no "False Positives"
- No one should ever be accessing these directories or scripts
- Access attempts for any of our fake directories are suspect by nature

<http://www.tracking-hackers.com/misc/faq.html>



**Discussion:**

Our use of Mod\_Security adheres to the Honeypot security methodology. Here are some excerpts from Lance Spitzner's honeypot web site:

**Detection**

While honeypots add little value to prevention, I feel they add extensive value to detection. For many organizations, it is extremely difficult to detect attacks. Often organizations are so overwhelmed with production activity, such as gigabytes of system logging, that it can be extremely difficult to detect when a system is attacked, or even when successfully compromised. Intrusion Detection Systems (IDS) are one solution designed for detecting attacks. However, IDS administrators can be overwhelmed with false positives. False positives are alerts that were generated when the sensor recognized the configured signature of an "attack", but in reality was just valid traffic. The problem here is that system administrators may receive so many alerts on a daily basis that they cannot respond to all of them. Also, they often become conditioned to ignore these false positive alerts as they come in day after day, similar to the story of "the boy who cried wolf". The very IDS sensors that they were depending on to alert them to attacks can become ineffective unless these false positives are reduced. This does not mean that honeypots will never have false positives, only that they will be dramatically fewer than with most IDS implementations.

Honeypots can simplify the detection process. Since honeypots have no production activity, all connections to and from the honeypot are suspect by nature. By definition, anytime a connection is made to your honeypot, this is most likely an unauthorized probe, scan, or attack. Anytime the honeypot initiates a connection, this most likely means the system was successfully compromised. This helps reduce both false positives and false negatives greatly simplifying the detection process. By no means should honeypots replace your IDS systems or be your sole method of detection. However, they can be a powerful tool to complement your detection capabilities.

## Foiling Vulnerability Scanners

- Create Mod\_Security Directives to catch requests to these directories
- You must make sure that there are no False Positives! You do not want to deny accesses to legitimate directories

```
SecFilter "/(bak|backup|archive|  
scripts|cgi-ocal|htbin|cgibin|cgis|cgi  
|win-cgi |cgi-win)/"
```

WHITE HAT <

SANS@Night - Mod\_Security

**Discussion:**


To effectively identify vulnerability scanner attacks, we can implement Mod\_Security directives to catch attempts to access common directories listed from Whisker.

Slide 59



This page intentionally left blank.

## Accessing OS Commands



- If I can find a CGI program that is vulnerable, I might be able to trick it into accessing some system utilities such as "ls" and "cat"
- By accessing these system tools, I am able to gain further information about the system such as
  - Contents of files
  - Directory Structures
  - Who is on the systems

SANS@Night - Mod\_Security

### Discussion:

Nearly every programming language allows the use of so called "system-commands", and many applications make use of this type of functionality. System-interfaces in programming and scripting languages pass input (commands) to the underlying operating system. The operating system executes the given input and returns its output to stdout along with various return-codes to the application such as successful, not successful etc. System commands can be a very convenient feature, which with little effort can be integrated into a web-application. Common usage for these commands in web applications are filehandling (remove,copy), sending emails and calling operating system tools to modify the applications input and output in various ways (filters).

Depending on the scripting or programming language and the operating-system it is possible to:

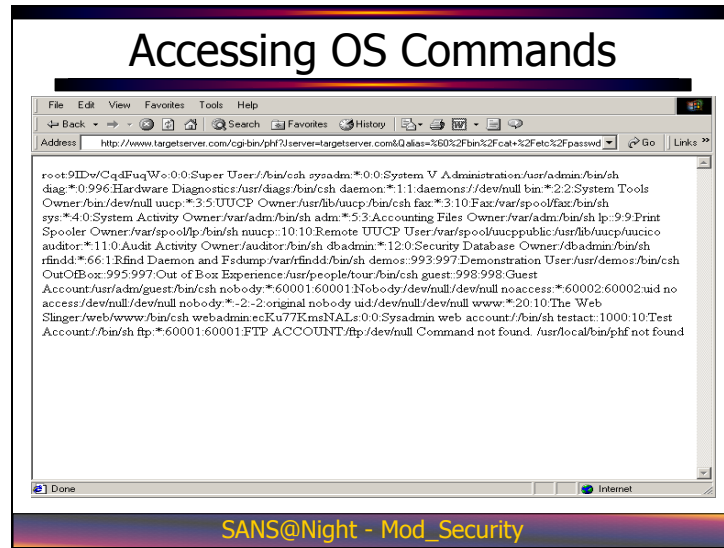
- Alter system commands

- Alter parameters passed to system commands

- Execute additional commands and OS command line tools.

- Execute additional commands within executed command

## Slide 61



This slide shows an example vulnerable CGI script (PHF), where the attacker was able to gain access to the `/etc/passwd` file. There are many overlapping layers of security which could have prevented this issue. The issue we are focusing on at this stage is the ability of the Apache web server to access and use certain OS level commands. In this case, if you look at the URL field, you can see that the PHF script accessed the `/bin/cat` command.

How can we prevent this from happening? We can restrict the ability of the Apache web server from accessing these commands.

## Disallow Directory Access

- Use Snort rules (earlier)
- Disallow access to OS level binaries/directories

```
SecFilter  
"/(etc|bin|sbin|tmp|var|opt|dev|kernel)/"
```

WHITE HAT <

SANS@Night - Mod\_Security

**Discussion:**

Here we create a Mod\_Security directive to identify access attempts to common OS level directories.

```
Directory Access AuditLog Entry


=====
Request: 192.168.145.1 - - [Wed Oct 29 11:29:18 2003] "GET /autohtml.php?op=modload&mainfile=x&name=/etc/passwd HTTP/1.1" 403 741
Handler: cgi-script
-----
GET /autohtml.php?op=modload&mainfile=x&name=/etc/passwd HTTP/1.1
Connection: Close
Content-Length: 0
Host: 192.168.145.100
User-Agent: Mozilla/4.75 (Nikto/1.30 )
mod_security-message: Access denied with code 403. Pattern match
"/(etc|bin|sbin|tmp|var|opt|dev|kernel)/" at THE_REQUEST.
mod_security-action: 403
--CUT--
=====

SANS@Night - Mod_Security
```

**Discussion:**

This slide shows an excerpt from the Mod\_Security audit\_log file. If configured correctly, Mod\_Security will capture and store both the client and server headers and the data payload of requests. This data can prove to be tremendously useful when investigating Web Attacks.

## URL Manipulation



- HTTP Request Manipulation
- Can send unacceptable Meta-Characters in the URL Request line and Client Headers
  - Directory Traversal
  - OS Command Execution
- The goal is to trick the Web Server into executing these un-intended commands

SANS@Night - Mod\_Security


**Discussion:**

An attacker will commonly try to issue HTTP requests that are formatted appropriately to execute system commands that were not intended for use by the Web Server.



## URL Attack Signatures

- **Directory Traversal** – “.”, “..”, “...”  
`http://host/cgi-bin/lame.cgi?file=../../../../etc/motd`
- **Hex Value** – “%20” Space, “%00” Null Requests  
`http://host/cgi-bin/lame.cgi?page%00=ls%20-a|`
- **Pipe Request** – “|”  
`http://host/cgi-bin/lame.cgi?page=ps%20-ef|grep%20root`
- **Semi-Colon Requests** – “;”  
`http://host/cgi-bin/lame.cgi?page=id;uname%20-a`
- **Redirect Requests** – “<”, “>”, “>>”  
`http://host/cgi-gi?page=echo%20“You’re%20wned”>index.htm`
- **System Commands** – “ls”, “echo”, “cat”, “tftp”, “ps”  
`http://host/cgi-bin/bad.cgi?doh=ps%20-aux`

 <http://www.cgisecurity.com/papers/fingerprint-port80.txt>

### Discussion:

This section has examples of common fingerprints used in exploitation of both web applications, and web servers. This section is not supposed to show you every possible fingerprint, but instead show you the majority of what exploits and attacks will look like. These signatures should pick up most of the known and unknown holes an attacker may use against you. This section also describes what each signature is used for, or how it may be used in an attack.

## Disallow Meta-Characters

- Define our Secfilter Rule to ONLY allow acceptable characters
  - Upper/Lowercase Letters
  - Numbers
  - Forward Slash
  - Period
  - Dash
  - UnderScore

SecFilter "[^a-zA-Z0-9|\.|\/|\\_|\-]"

WHITE HAT <

SANS@Night - Mod\_Security


### Discussion:

The concept is to implement a security layer that will validate ALL HTTP requests sent to a Web Server. This layer should be able to read packet payloads and effectively terminate all TCP connections that are contain unacceptable characters.

We want to use Mod\_Security to inspect all of the incoming client HTTP Requests. We will define a rule set which will only allow **ACCEPTABLE** characters. This takes a proactive approach, as opposed to normal Intrusion Detection where the IDS uses an attack signature database of **FORBIDDEN** actions. This concept is based on a similar mindset to the Access Control mechanisms of TCP-Wrappers ([ftp://ftp.porcupine.org/pub/security/tcp\\_wrapper.txt.Z](ftp://ftp.porcupine.org/pub/security/tcp_wrapper.txt.Z)). TCP-Wrappers controls access to services by utilizing two files - hosts.allow and hosts.deny. These files restrict access based on "**Where** you are coming from - IP Address or Hostname" and "**What** you want - Specified service I.E.- FTP." Since there is a limited number of valid hosts who should be using FTP to access your server, it is easy enough to list these IP's within the hosts.allow file and then simply add a line in the hosts.deny file that will deny everyone else. This is a simple, yet effective, method of access control.

Now, imagine that we apply the current mentality of IDS technology to TCP-Wrappers. This would be equivalent to having an **empty** hosts.allow file and then listing every single IP address that you **don't** want to FTP to your server in the hosts.deny file! That is sheer madness, however, that is how most IDS technologies work. Instead of listing what is acceptable (Valid HTTP Requests) on the network, the IDS' are specifying what is not allowed (i.e. - Attack Signatures). With Mod\_Security we can specify valid HTTP characters and only allow these requests to be served by the Apache child process.

## Searching for Vulnerable CGIs



- I can use a web scanner such as Whisker to search a target web server for vulnerable CGI scripts
- Once the scanner has identified a script, I can then try to exploit it 😊

SANS@Night - Mod\_Security

**Discussion:**

An attacker will commonly use a vulnerability scanner to automate the process of locating vulnerable programs.



**Discussion:**

This shows an excerpt from the Whisker main.test file and lists some of the CGI scripts it will check.

## Disallow Access to Non-Valid CGIs

- Define an inverted Secfilter Rule to ONLY allow access to legitimate CGI scripts
- For example, say you only have 3 legitimate CGI scripts
  - Script1.cgi
  - Script2.cgi
  - Script3.cgi

```
<Directory "/var/www/cgi-bin">  
SecFilter "!(Script[1-3]\.cgi)"  
Deny from all  
</Directory>
```

WHITE HAT <

01010101

SANS@Night - Mod\_Security

**Discussion:**

Here we are using a similar approach to allowing only certain characters to be sent to our web server. Instead of specifying individual characters, we specify individual files. This is feasible since you will most likely only have a small number of valid CGI scripts.

## Tracking Security Events

- Combining Mod\_Security with CGI scripts for error pages provides detailed email alerts for attacks
- Use custom CGI error pages for – 401/403 Status Codes
- The CGI scripts automate many important tasks
  - Uses Environmental Variables from the printenv script – sends this info to WebAdmin instead of to the client
  - Issues HTML page to attacker with Warning Banner
  - Notifies WebAdmin via Email
- Emails contain the following info:
  - The CGI Environment Variables (Full Client HTTP Header Info)
  - A URL hyperlink to immediately run a Traceroute and WHOIS on the attacker's IP address.

WHITE HAT

SANS@Night - Mod\_Security

**Discussion:**

SysAdmins need to keep tabs on all of these security related issues with their web servers. To assist with this monitoring, the web server should be configured to use custom CGI error response pages for 403 server response codes generated by Mod\_Security. The error pages are PERL CGI scripts that are initiated every time the server issues either of these response codes. These scripts accomplish many important tasks including issuing an html warning banner to the client and immediately sending an e-mail notification to the SysAdmin. The e-mail message automates the process of manually collecting security related session information from the web server access and error logs for the request.

The hyperlink feature, within the e-mail message, is useful for tracking down the appropriate "network abuse" contact personnel responsible for the attacker's IP segment. While not every 403 message warrants these investigative actions, repeated errors identified from a certain IP address should be handled appropriately. This CGI alert e-mail system facilitates the prompt notification of proper personnel.

### CGI Alerts Analysis

- Can determine if the alert was caused by a Vulnerability Scanner or a Browser Request
- Number of emails received
- Time interval
  - If requests/emails are rapid -> Scanner
  - If requests/emails are sporadic -> Browser
- User Agent Field
  - (Mozilla/4.7 [en] (Win95; U)) -> Netscape
  - (Mozilla/4.0 (compatible; MSIE 5.01; Windows 98) -> IE
  - Blank – “-” -> Scanner/Unknown Application

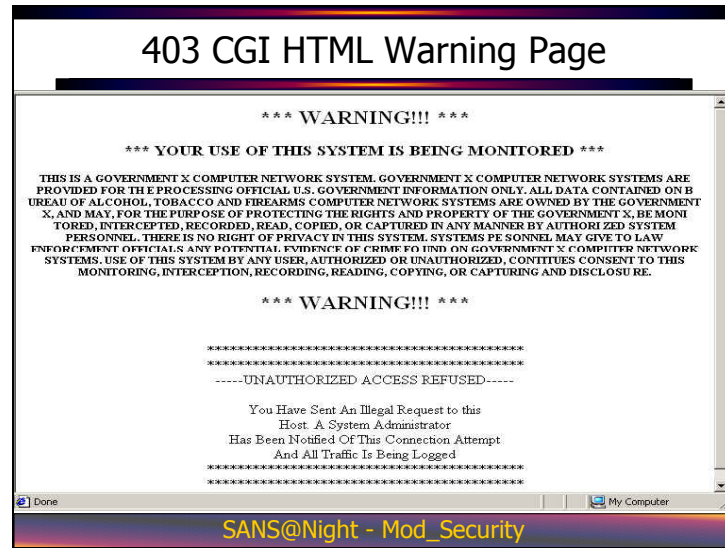
WHITE HAT <

SANS@Night - Mod\_Security

#### Discussion:

By examining these e-mail alerts, it is possible to determine if the attacker was either conducting a vulnerability scan or trying to exploit the CGI scripts directly. Notice the "User Agent" line from the e-mail message above? This information, taken from one of the PERL CGI script's environmental variables, can aid in determining what application triggered the script. Since this variable is blank, this attempt was most likely executed by an automated script or application such as Whisker or ISS. If the "User Agent" field had specified a browser such as, Netscape (Mozilla/4.7 [en] (Win95; U)) or Internet Explorer (Mozilla/4.0 (compatible; MSIE 5.01; Windows 98), this would indicate an attempt to exploit a vulnerable CGI script rather than conducting a vulnerability scan. The final e-mail parameter to consider is the "Date Stamp." If these five emails happen very rapidly, odds are an automated attack was executed. If inconsistent delays are present between the access attempts, odds are the attacker was using a browser. These delays are indicative of manually typing in the URL information into a browser. You will want to edit the information to be appropriate for your site.

## Slide 72

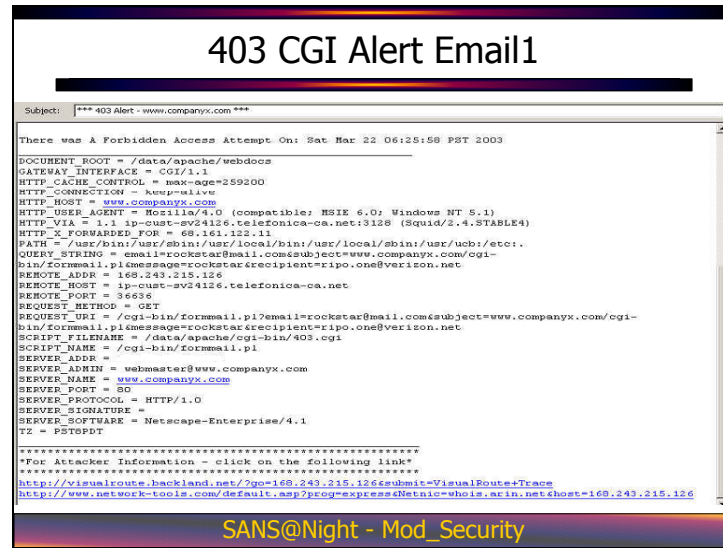


### Discussion:

This slide shows an example of the type of 403 error page which can be displayed by the CGI scripts. You will notice that both a WARNING Banner and notice of session logging are being displayed.



## Slide 73



### Discussion:

This email message shows an example of a 403 alert email message. Let's take a moment to analyze this HTTP session.

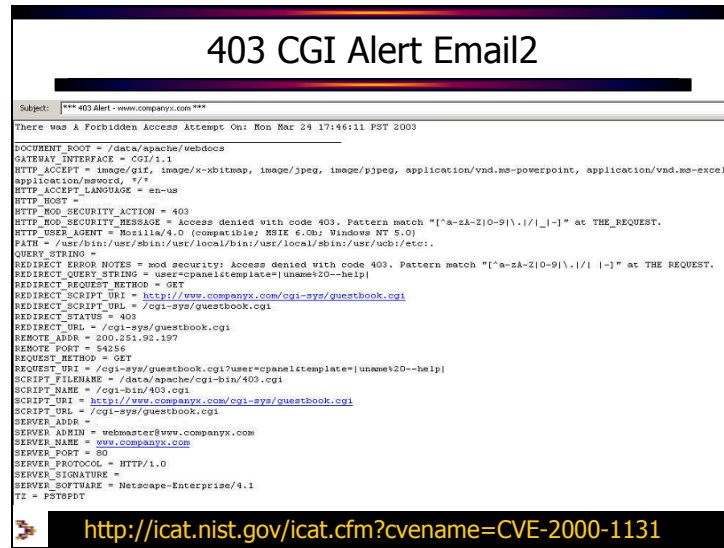
Since this email was generated by a 403 – Forbidden, the first thing to look at is “What was the URL Request?”. This information is located on the REQUEST\_URI line. In this email – the client requested the following URL: **/cgi-bin/formmail.pl?email=rockstar@mail.com&subject=www.companyx.com/cgi-bin/formmail.pl&message=rockstar&recipient=ripo.one@verizon.net**

This is an obvious SPAM Relay search. The client is checking to see if our web server has a Formmail.pl script which, if not configured correctly, could allow anyone to send email through our server. These types of Open Spam Relays make it vastly more difficult to track down spammers.

Another interesting entry is the HTTP\_USER\_AGENT info. Most Spam Relay searching is conducted by automated scripts – which do not provide any user agent info. This connection does contain user agent information. It appears that the client was using a Microsoft 6.0 version of Internet Explorer. Could this information be spoofed within a script/tool? Sure, however it is important to remember this concept – if the user agent field is empty you can be sure that it was NOT a legitimate web browser.

The last interesting piece of info is the HTTP\_VIA and HTTP\_X\_FORWARDED\_FOR entries. We will be discussing these HTTP headers further in the TRACE section, however it is sufficient to note that these client headers indicate that this person is using a Squid Proxy server to connect to our web server. Does the use of Proxy Servers always indicate malicious intent? No. In this case, however, it did. If we wanted to try and track this person down, the HTTP\_X\_FORWARDED\_FOR information comes into play. Lucky for us, the Squid Proxy Server was configured to append this HTTP header, which gives us the REAL IP address of the sender!

## Slide 74



### Discussion:

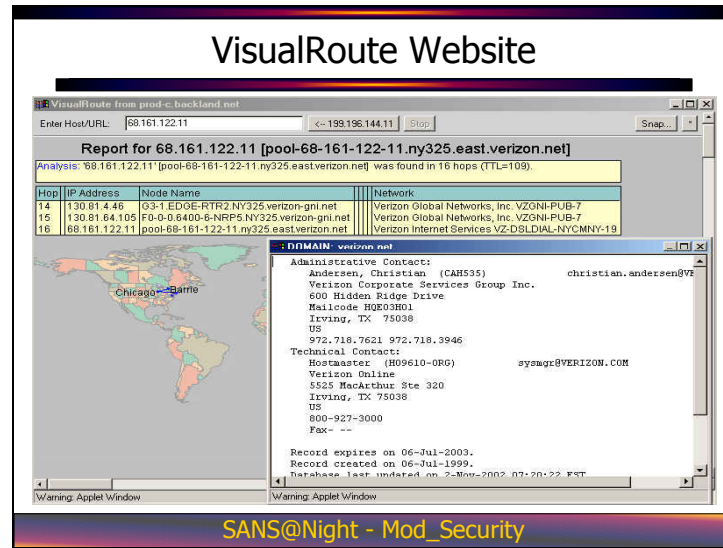
This email message shows an example of a 403 alert email message. Let's take a moment to analyze this HTTP session.

This email was sent because Mod\_Security identified illegal characters in the Request URI.

We specified our Mod\_Rewrite Regular Expression to only allow acceptable characters: SecFilter "[^a-zA-Z[0-9]\\.|/|\_|-]"

The client was trying to access a vulnerable CGI script named - /cgi-sys/guestbook.cgi

The client tried to trick the guestbook.cgi script into executing the "uname" OS command by using the "|" pipe meta-character.



### Discussion:

This slide shows how the URL link within an email can be used to immediately run a trace on the VisualRoute Web Site to locate the network block owners of an offending IP address. In this case, we are using the IP address – 68.161.122.11 – which was identified by the HTTP\_X\_FORWARDED\_FOR client header from the previous slide. This IP address appears to be a Verizon DSL home internet user based in New York. The network block owners are Verizon and when we can click on the network owner's name on the screen, the VisualRoute application will run a WHOIS Domain Query for this name and report back to contact information. Armed with this data, appropriate Security Personnel could contact the owners.

What is interesting about running this trace is that if we did not have the HTTP\_X\_FORWARDED\_FOR information, we would have run a trace on the Proxy IP address – 168.243.215.126 – instead. This IP address is located in Spain. That is quite a big difference in location if we were starting an investigation! Always consider the use of Proxy Servers when conducting a web investigation.

## Slide 76

**DShield.Org Website**

The screenshot shows a web browser window titled "DShield - IP Info - Microsoft Internet Explorer". The address bar shows "http://www.dshield.org/ipinfo.php?ip=129.95.74.205". The page content includes a search bar with the IP address "129.95.74.205" and a "Submit" button. Below this, the "IP Info" section displays the IP address and hostname. The "DShield Profile" section shows a table with attack statistics. The "Top 10 Ports hit by this source:" section shows a table of ports and attacks. The "Last Fightback Sent:" section shows the status of a fightback. The "Whois:" section shows the IP's origin and location. The footer of the browser window shows "SANS@Night - Mod\_Security".

**IP Info**

Check another IP Address:

**IP Address:** 129.95.74.205  
**HostName:** pgt5.ese.ogi.edu

**DShield Profile:**

|                           |                          |
|---------------------------|--------------------------|
| Country:                  | US                       |
| Contact E-mail:           | don@ADMIN.OGI.EDU        |
| Total Records against IP: | 125339                   |
| Number of targets:        | 64766                    |
| Date Range:               | 2003-07-17 to 2003-07-17 |

[Update Summary](#)

**Top 10 Ports hit by this source:**

| Port  | Attacks | Start      | End        |
|-------|---------|------------|------------|
| 80    | 125334  | 2003-07-17 | 2003-07-17 |
| 22788 | 4       | 2003-07-17 | 2003-07-17 |

**Last Fightback Sent:** sent to don@ADMIN.OGI.EDU on 2003-07-17 02:30:45

**Whois:**

|             |                             |
|-------------|-----------------------------|
| OrgName:    | Oregon Graduate Institute   |
| OrgIP:      | 004                         |
| Address:    | 19600 NW Von Neumann        |
| City:       | Beaverton                   |
| StateProv:  | OR                          |
| PostalCode: | 97006                       |
| Country:    | US                          |
| NetRange:   | 129.95.0.0 - 129.95.255.255 |
| CIDR:       | 129.95.0.0/16               |
| NetName:    | OREGRADNET                  |

**SANS@Night - Mod\_Security**


### Discussion:

This slide shows an example of using the Dshield.org hyperlink feature of the CGI alert emails. The main advantage to using Dshield is that they are able to correlate data from a vast amount of hosts. This provides a wider view of attacks and allows you to better gauge the threat level presented to you.

In the example above, you can see that this client IP address has been up to no good – due to the “Total Records against IP” number.

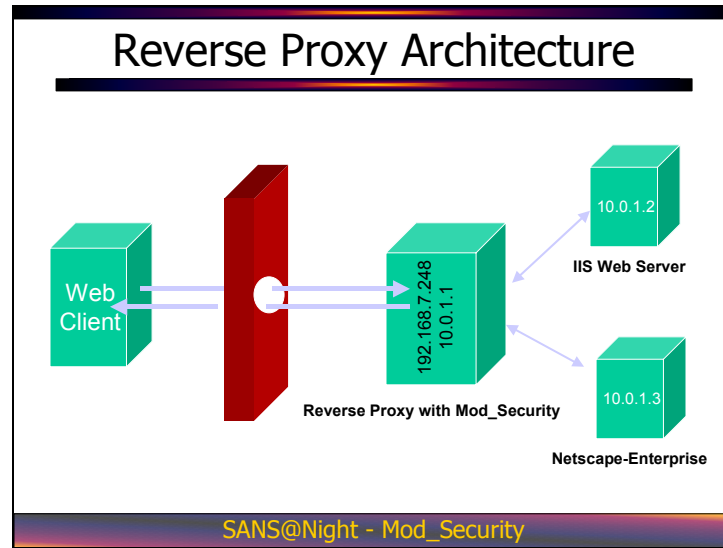
### Mod\_Security as a Reverse Proxy

- **Single point of access**
- **Increased performance**
- **Network isolation**
- **Network topology hidden from the outside world**
- **You can implement filters to protect “vulnerable” web servers**
  - Until patches are available

<http://www.securityfocus.com/infocus/1739>

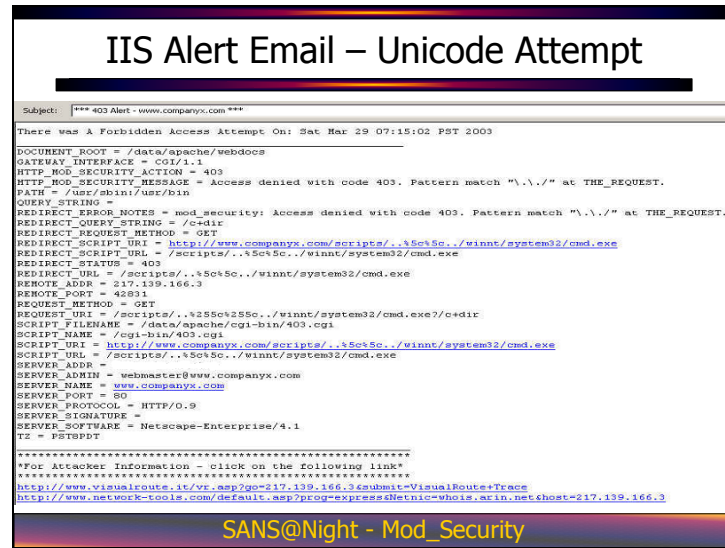
**Discussion:**

We will be discussing many of these common web vulnerabilities during the security configuration steps of this presentation.



**Discussion:**

We will be discussing many of these common web vulnerabilities during the security configuration steps of this presentation.



### Discussion:

This email message shows an example of a 403 alert email message. Let's take a moment to analyze this HTTP session.

This email was generated by a NIMDA Worm Scan.

In this case, Mod\_Security spotted this scan because we had specified a generic Directory Traversal directive – SecFilter “\.\.”. This caught the NIMDA Scan since it sends numerous requests trying to access cmd.exe by encoding Directory Traversal attempts.

Examples:

GET /scripts/..%5c../winnt/system32/cmd.exe?/c+dir

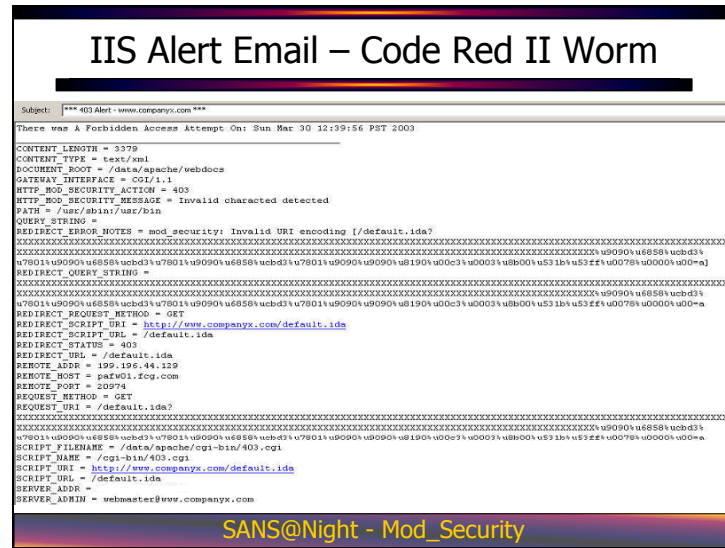
GET /\_vti\_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe?/c+dir

GET /\_mem\_bin/..%5c../..%5c../..%5c../winnt/system32/cmd.exe?/c+dir

GET /scripts/..xc1\x1c../winnt/system32/cmd.exe?/c+dir

Instead of specifying every individual signature for NIMDA, we can catch most of these attacks by using our directory traversal signature.

## Slide 80



### Discussion:

This email message shows an example of a 403 alert email message generated by Mod\_Security. By inspecting this email alert, we can see that it was generated by a Code Red II Worm Scan.

The HTTP\_MOD\_SECURITY\_MESSAGE states that there are “Invalid characters detected”. These characters are detected since Code Red uses them during infection of IIS.

Mod\_Security further states that there is “Invalid URI encoding”.



## Final Thoughts

- No system is 100% secure
- Majority of attacks can be effectively deterred by minimal security measures
- Consider the issues discussed and determine their relevance to your environment
- Mod\_Security is highlighted in the upcoming CIS Apache Benchmark
- Thank you for your time
- Questions?
- Please fill out the SANS Class Evaluation!
- **Send feedback to:**  
**RCBarnett@Hushmail.com**

SANS@Night - Mod\_Security

### Discussion:

Unfortunately, no matter how many security measure are implemented, no system will ever become 100% secure. There is an old security adage that addresses this fact: "The only 100% Secure System, is the one that is not plugged into the network and is still in it's cardboard box." A non-networked web server is counter productive since its sole purpose is to allow clients access to information. The goal of all WebAdmins should be to mitigate the associated risk involved with running a public web server. Since it is not possible to completely secure an Internet system, WebAdmins need to formulate a plan to both prevent and reduce the impact of a successful web site attacks. By taking appropriate security measures, tremendous progress towards protecting web servers can be made. Hopefully, the techniques outlined in this presentation will assist WebAdmins to this end.

Please send all questions and feedback to the following email address:

RCBarnett@hushmail.com

Thank you for your time and Good Luck with securing your Apache Web Servers!

Slide 82



**Discussion:**

If time permits, we will have an Open Audit Workshop where you can run some of the tools mentioned during the presentation.