Securing Dynamic Websites using LAPP and ModSecurity

Brad Baker
CS526
May 7th, 2008

Overview

Project Goals

- Research potential security configurations for LAPP or LAMP web servers including ModSecurity.
- Implement a basic LAPP system and test security configuration

Test Environment

- Web servers
  - Ubuntu 7.10
  - Apache 2.2.4
    - Mod_security
    - Mod_unique_id
    - Mod_php
  - Php 5.2.3
  - Postgresql 8.2.3
  - Curl, lua, libxml2
- Web application
  - Created a custom PHP application with Postgresql
  - Built a custom login method
    - Maximum login attempts
    - Auto session timeout
- Client machine
  - Windows Vista
  - Initiated basic malicious requests
  - Acted as log console server

The Problem

- Dynamic web applications are subject to a wide variety of threats, including:
  - Poorly implemented custom applications
  - Use of popular software packages that may contain vulnerabilities and be exploit targets
  - Unpatched or slowly patched server software
  - Unknown exploits to server software
  - SQL injection, cross-site scripting, application and software specific vulnerabilities.

Basic Solutions

- Quality application development
- Prompt patching and updating for server software
- Layers of access control including firewalls and server hardening
- These solutions are not always ideal:
  - Secure development practices not always used. Software packages could be delivered with vulnerabilities.
  - Patching takes time and risks server stability. Unknown exploits cannot be patched against.
  - Machine hardening may not protect the application.

Additional Solutions

- Additional methods to protect systems include:
  - Intrusion detection systems (IDS) on the network
    - Proactive, not focused on web requests, bad with SSL
  - Chroot jail for Apache server
    - Reactive, protects system but not Apache process
  - Suhosin for PHP installation
    - Proactive, protects PHP from malicious requests and unknown flaws
  - ModSecurity
    - Proactive, focused on web protocols, can analyze SSL traffic

ModSecurity

- Current Version: 2.5.3 (April 24, 2008)
  - Copyright © Breach Security, Inc. (http://www.breach.com)
- ModSecurity is a Web Application Firewall
- Module works between the Apache server process and the client
- Operation is controlled by robust rule processing including regular expression pattern matching
- Analyzes request and response data, blocks transmission, logs transactions for analysis

Strengths

- Module provides:
  - HTTP protection, Common Web Attacks Protection, Automation detection, Trojan Protection, Error Hiding
- Protects from unknown vulnerabilities, allows time for patching application code and server software.
- Standard core rules provide defense against potential attacks. Rules are optimized and cover a variety of attacks.
- Negligible performance decrease.

Example Rules

1. Example rule for PHP information leakage (response analysis)

```
SecRule RESPONSE_BODY
  "<b>Warning<\/b>.{0,100}?:.{0,1000}?\bon line\b"
  "phase:4,t:none,ctl:auditLogParts=+E, deny,
  log,auditlog,status:500,msg:'PHP Information Leakage',
  id:'970009',tag:'LEAKAGE/ERRORS',severity:'4'"
```

2. Example rule for invalid ascii values

```
SecRule REQUEST_FILENAME|REQUEST_HEADERS_NAMES|
  REQUEST_HEADERS| !REQUEST_HEADERS:Referer
  "@validateByteRange 32-126" \
  "phase:2,deny,log,auditlog,status:400,msg:'Invalid
  character in request',
  id:'960018',tag:'PROTOCOL_VIOLATION/EVASION',
  severity:'4',t:none,t:urlDecodeUni"
```

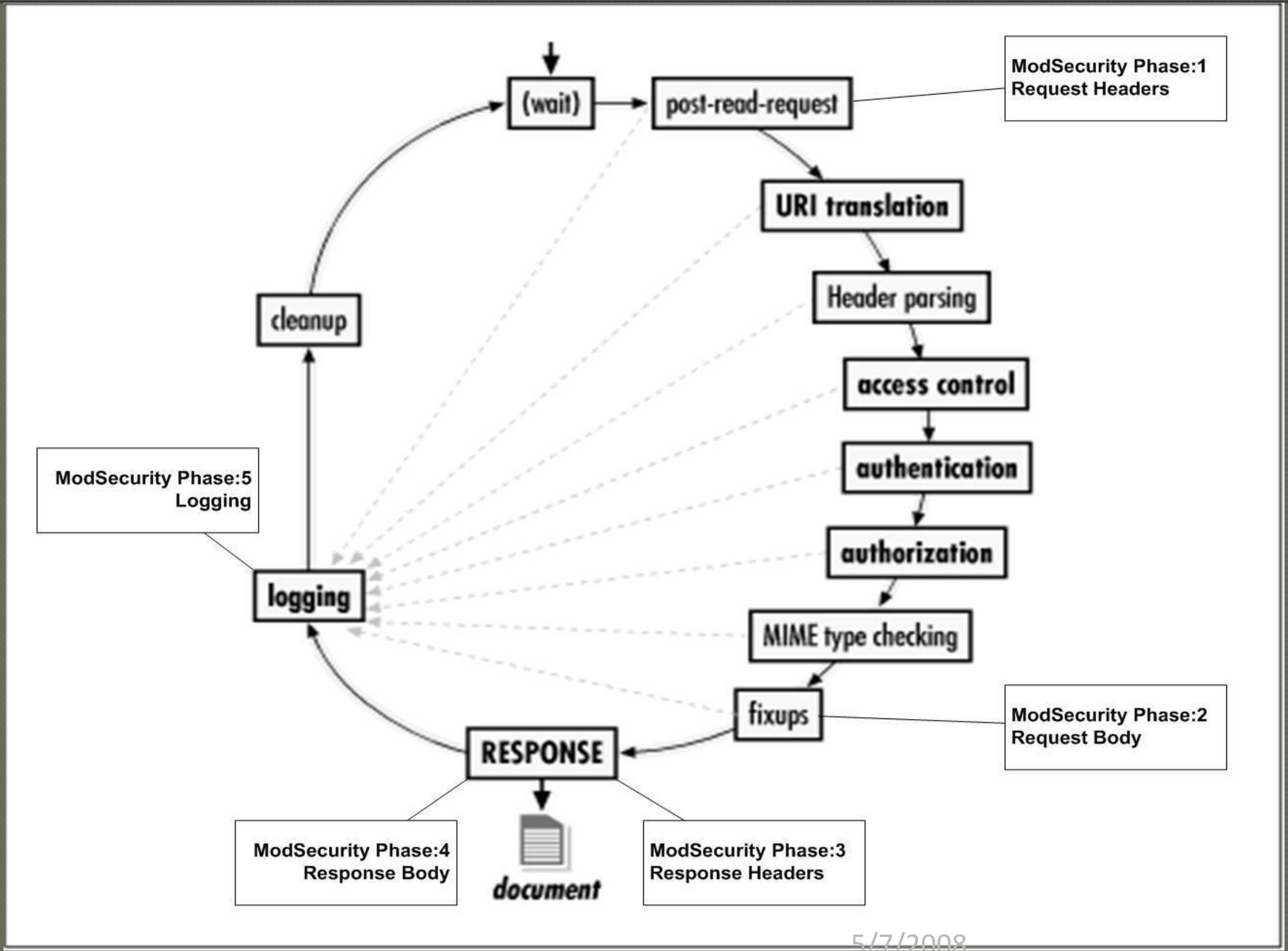
3. Example rule to block requests with numeric host in header:

```
SecRule REQUEST_HEADERS:Host "^[\d\.]+$"
  "phase:2,t:none,deny,log,auditlog,status:400,msg:'Host
  header is a numeric IP address', severity:'2',
  id:'960017',tag:'PROTOCOL_VIOLATION/IP_HOST'"
```

Processing Phases

- Rules can process against one of the following processing phases:
  1. Request headers
  2. Request body
  3. Response headers
  4. Response body
  5. Logging
- This approach allows protection against malicious requests and information leakage in response data

Processing Phases



(wait)
post-read-request
ModSecurity Phase:1 Request Headers
URI translation
Header parsing
access control
authentication
authorization
MIME type checking
fixups
ModSecurity Phase:2 Request Body
RESPONSE
ModSecurity Phase:3 Response Headers
document
ModSecurity Phase:4 Response Body
logging
ModSecurity Phase:5 Logging
cleanup

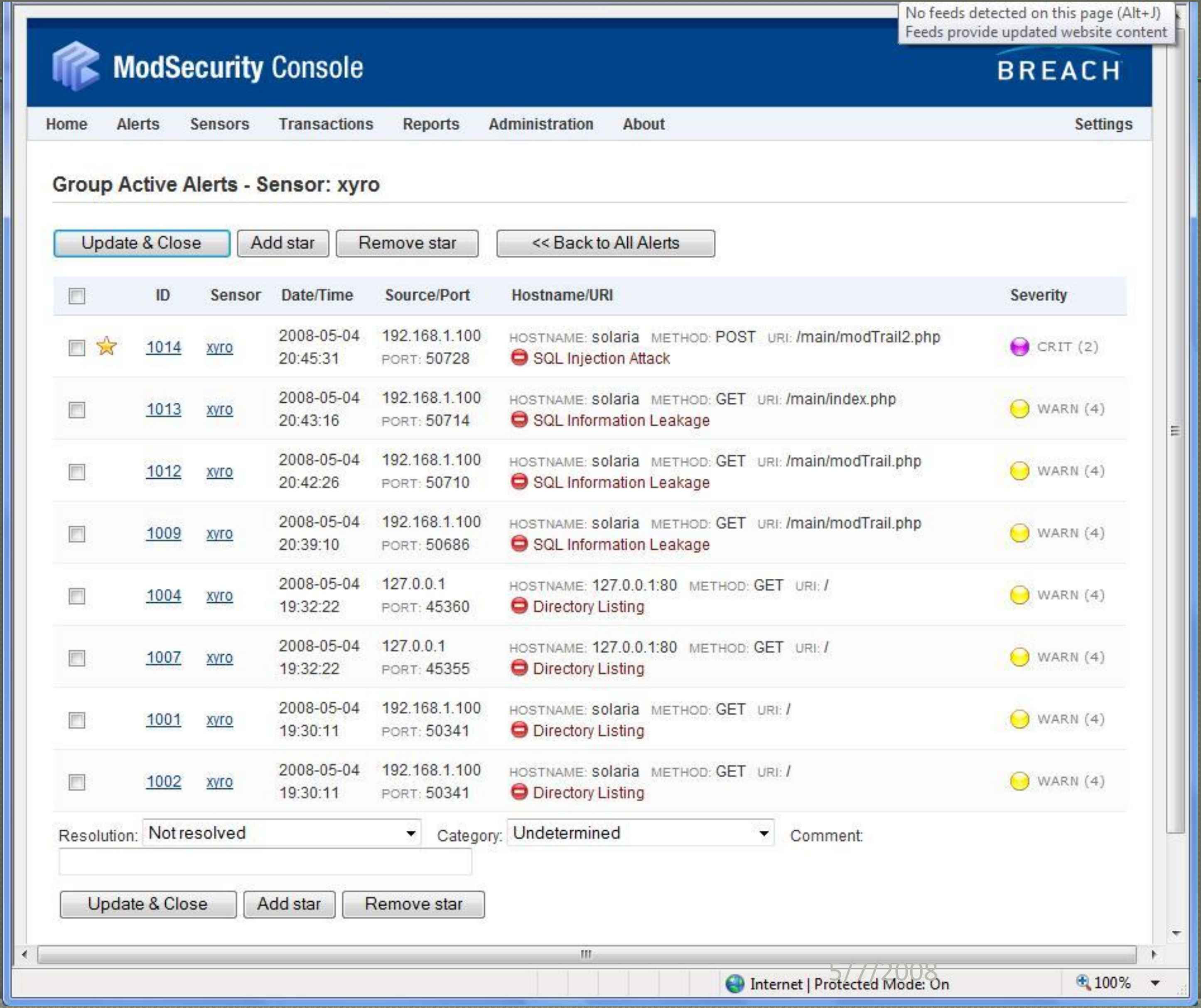
Log File Data

```
--a0c36e2a-A--[03/May/2008:09:13:03 --0600]
71TDcMCoAWQAABuUA9gAAAAD 192.168.1.101 49828 192.168.1.100 80--
a0c36e2a-B--POST /main/modTrail2.php?trailid=7 HTTP/1.1

--a0c36e2a-C--
tname=1&tlocate=1+%27%3Binsert+into%0D%0A%0D%0A&tdesc=&trailid=7&a
dduser=1&addtime=2008-04-30+22%3A30%3A11.423323

--a0c36e2a-H--Message: Access denied with code 501 (phase 2).
Pattern match
"(?:\b(?:(?:s(?:elect\b(?:.{1,100}?\b(?:(?:length|count|top)\b.{1,
100}?\bfrom|from\b.{1,100}?\bwhere)|.*?\b(?:d(?:ump\b.*\bfrom|ata_
type)|(?:to_(?:numbe|cha)|inst)r))|p_(?:(?:addextendedpro|sqlexe)c
|(?:oacreat|prepar)e|execute(?:sql)?|makewebtask)|ql_(? ..." at
ARGS:tlocate. [file
"/etc/apache2/conf/modsecurity/rulesAll/modsecurity_crs_40_generic
_attacks.conf"] [line "66"] [id "950001"] [msg "SQL Injection
Attack"] [data "insert into"] [severity "CRITICAL"] [tag
"WEB_ATTACK/SQL_INJECTION"]Action: Intercepted (phase 2)Stopwatch:
1209827583116144 3646 (490* 2404 -)Producer: ModSecurity for
Apache/2.5.3 (http://www.modsecurity.org/); core
ruleset/1.6.1.Server: Apache/2.2.4 (Ubuntu) PHP/5.2.3-1ubuntu6.3
```

Modsecurity Console

- Current Version: 1.0.4 (April 25, 2008)
  - Copyright © Breach Security, Inc. (http://www.breach.com)
- Uses mlogc log collector
  - Separately installed and configured in ModSecurity
- Apache with ModSecurity enabled publishes output files to console service
- Console provides framework for log analysis, attack detection and email alerts
- Console can operate on external server

Console Log Entries



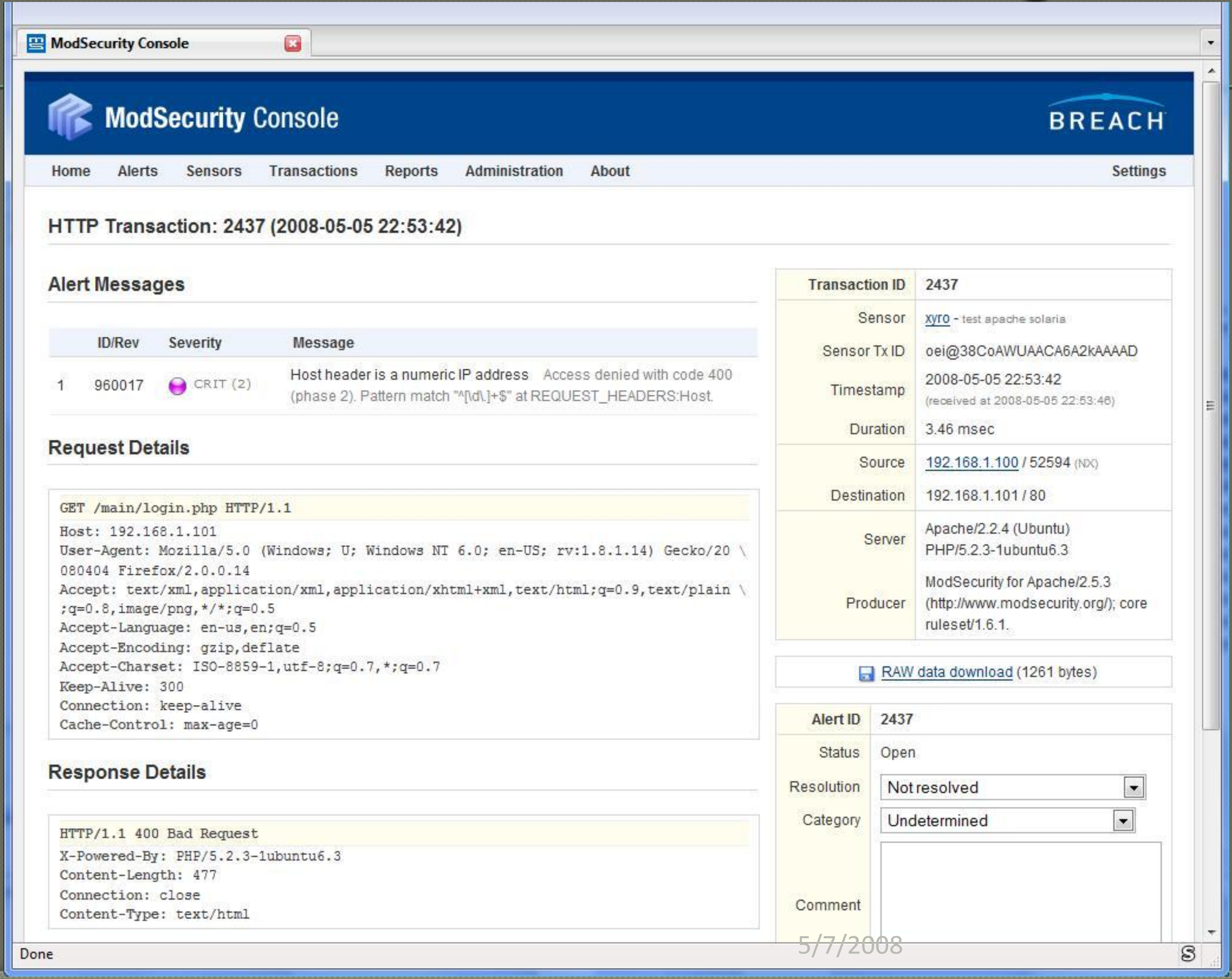
No feeds detected on this page (Alt+J)
Feeds provide updated website content
ModSecurity Console
BREACH
Home Alerts Sensors Transactions Reports Administration About Settings
Update & Close
Add star
Remove star
<< Back to All Alerts
Internet | Protected Mode: On
100%

Group Active Alerts - Sensor: xyro

	ID	Sensor	Date/Time	Source/Port	Hostname/URI	Severity
☐ ☆	1014	xyro	2008-05-04 20:45:31	192.168.1.100 PORT: 50728	HOSTNAME: solaria METHOD: POST URI: /main/modTrail2.php SQL Injection Attack	CRIT (2)
☐	1013	xyro	2008-05-04 20:43:16	192.168.1.100 PORT: 50714	HOSTNAME: solaria METHOD: GET URI: /main/index.php SQL Information Leakage	WARN (4)
☐	1012	xyro	2008-05-04 20:42:26	192.168.1.100 PORT: 50710	HOSTNAME: solaria METHOD: GET URI: /main/modTrail.php SQL Information Leakage	WARN (4)
☐	1009	xyro	2008-05-04 20:39:10	192.168.1.100 PORT: 50686	HOSTNAME: solaria METHOD: GET URI: /main/modTrail.php SQL Information Leakage	WARN (4)
☐	1004	xyro	2008-05-04 19:32:22	127.0.0.1 PORT: 45360	HOSTNAME: 127.0.0.1:80 METHOD: GET URI: / Directory Listing	WARN (4)
☐	1007	xyro	2008-05-04 19:32:22	127.0.0.1 PORT: 45355	HOSTNAME: 127.0.0.1:80 METHOD: GET URI: / Directory Listing	WARN (4)
☐	1001	xyro	2008-05-04 19:30:11	192.168.1.100 PORT: 50341	HOSTNAME: solaria METHOD: GET URI: / Directory Listing	WARN (4)
☐	1002	xyro	2008-05-04 19:30:11	192.168.1.100 PORT: 50341	HOSTNAME: solaria METHOD: GET URI: / Directory Listing	WARN (4)

Resolution: Not resolved Category: Undetermined Comment:

Console Log Detail



HTTP Transaction: 2437 (2008-05-05 22:53:42)

Alert Messages

	ID/Rev	Severity	Message
1	960017	● CRIT (2)	Host header is a numeric IP address Access denied with code 400 (phase 2). Pattern match "^[\d\.]+$" at REQUEST_HEADERS:Host.

Request Details

```
GET /main/login.php HTTP/1.1
Host: 192.168.1.101
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.8.1.14) Gecko/20 \
080404 Firefox/2.0.0.14
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain \
;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cache-Control: max-age=0
```

Response Details

```
HTTP/1.1 400 Bad Request
X-Powered-By: PHP/5.2.3-1ubuntu6.3
Content-Length: 477
Connection: close
Content-Type: text/html
```

Transaction ID	2437
Sensor	xyro - test apache solaria
Sensor Tx ID	oei@38CoAWUAACA6A2kAAAAD
Timestamp	2008-05-05 22:53:42 (received at 2008-05-05 22:53:46)
Duration	3.46 msec
Source	192.168.1.100 / 52594 (NX)
Destination	192.168.1.101 / 80
Server	Apache/2.2.4 (Ubuntu) PHP/5.2.3-1ubuntu6.3
Producer	ModSecurity for Apache/2.5.3 (http://www.modsecurity.org/); core ruleset/1.6.1.

RAW data download (1261 bytes)

Alert ID	2437
Status	Open
Resolution	Not resolved
Category	Undetermined
Comment	

Done

Conclusions / Future Research

- Modsecurity is an effective tool for securing web applications on apache.
- Complicated regular expressions makes new rule development a challenge.
- Log collection console appears to have DoS issue with large volume of rejected requests.
- Ideal solution is software patching, application hardening and application specific rules in addition to core rule set.

References

- ModSecurity:
  1. http://www.modsecurity.org/index.php
  2. http://www.onlamp.com/pub/a/apache/2003/11/26/mod_security.html
  3. http://www.securityfocus.com/infocus/1739
  4. http://www.linuxjournal.com/article/8708
  5. http://www.debian-administration.org/articles/65
- Chroot
  1. http://howtoforge.com/chrooted_debian_sarge_lamp_on_ubuntu_desktop
- Suhosin
  1. http://www.hardened-php.net/suhosin/
  2. http://isc.sans.org/diary.html?storyid=2163
- Misc
  1. http://www.ibm.com/developerworks/web/library/wa-lampsec/?ca=dgr-lnxw07LampSecurity
  2. http://www.askapache.com/htaccess/mod_security-htaccess-tricks.html
  3. http://www.postgresql.org/