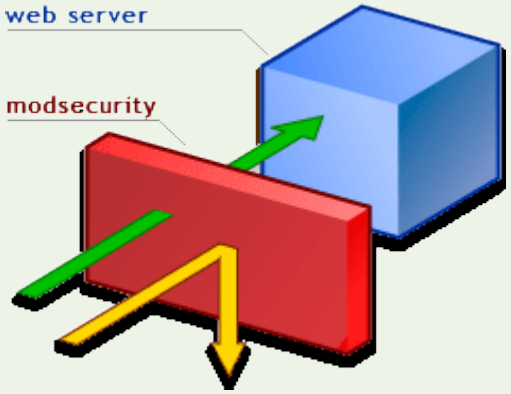
Advanced Web Application Defense with ModSecurity



Daniel Fernández Bleda
&
Christian Martorella

What The Hack!
Liempde (The Netherlands)
July 28-31, 2005

Who we are? (I)

- Christian Martorella:
  - +6 years experience on the security field, mostly doing audits and Pen-testing.
  - Open Information System Security Group, Barcelona chapter President.
  - First Improvised Security Testing Conference Organizer
  - Security Forest, ExploitTree maintainer.
  - CISSP, OPST

Who we are? (II)

- Daniel Fernández Bleda:
  - Security Engineer +5 years experience on the field.
  - Institute for Security and Open Methodologies (ISECOM), Member, and OSSTMM promoter.
  - Open Information System Security Group, Barcelona chapter Member.
  - CISSP, CISA, OPST/OPSA Trainer

Aim of the Presentation

- Security of web applications
- Introduce Modsecurity
- Show how can you protect your web applications
- Show our work in extending the features of this module.

Actual Scenario (I)

- Web Applications are insecure
- Today are the most vulnerable part of company infrastructure.
- Everyday the attacks at the application layer are growing:
  - SQL Injection
  - XSS
  - Command Injection
  - Buffer Overflows
  - Session manipulation
  - Etc..

Actual Scenario (II)

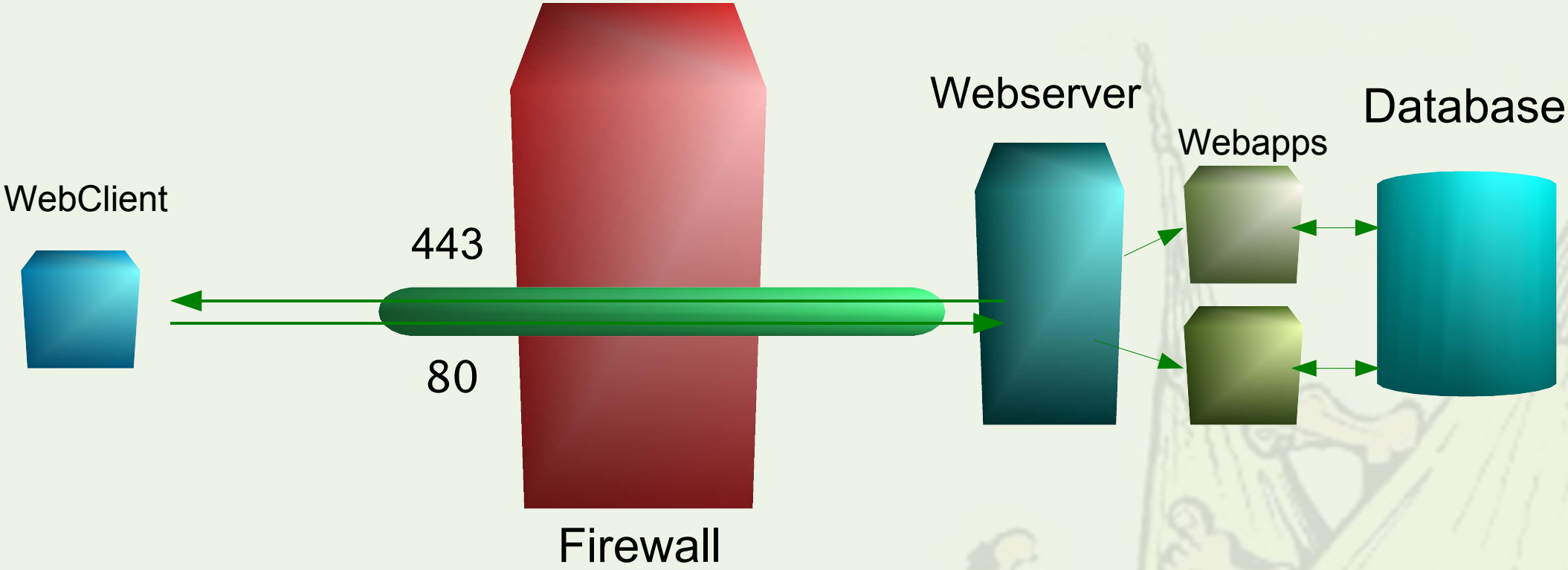
- The quantity of vulnerabilities is growing
- 80% of the last 30 exploits posted to Milw0rm target Web Applications: Wordpress, Phpbb, Xml-rpc, etc.
- We can find a lot of Firewalls to analyse and filter traffic at the network level, but at the Application level, what are our options? Open and Free, very little..
- There are secure networks with insecure applications that jeopardize all security of the company.

What are the Problems? (I)

- The web development is chaotic
- Lack of web security awareness.
- First users requirement, last security (if time allow)
- If it works don't touch it (sad but true)
- False sense of security:
  - We have a firewall, we are safe
  - We use SSL, we are safe

What are the Problems? (II)

Network firewalls doesn't do anything at this level:



What is ModSecurity?

- Intrusion Detection / Prevention for Web Applications
- Operate as an Apache Module.
- Open Source and GPL
- Development by Ivan Ristic

The purpose of ModSecurity is to increase web application security by protecting them from known and unknown attacks

When can we use it?

- Applications you can't modify: legacy applications, protected code like Zend encoder, Phpshield, etc.
- New vulnerability discovered, temporal protection until patch is released.
- Intrusion Detection .
- Extra layer of security.
- To protect Web services.
- Web applications operated by people other than original software developers.

Features (I)

- **Request filtering:** incoming requests are analysed as they come in, and before they get handled by the web server or other modules.
- **Output Filtering:** It could analyse the server response. (Error, Critical data, Ex. PHP Errors)
- **Understanding of the HTTP protocol**: since the engine understands HTTP, it performs very specific and fine granulated filtering.

Features (II)

- **Anti-evasion techniques**: paths and parameters are normalized before analysis takes place in order to fight evasion techniques.
  - Remove multiple forward slash characters
  - Treat backslash and forward slash characters equally
  - Remove directory self-references
  - Detect and remove null-bytes (%00)
  - Decode URL encoded characters

Features (III)

- **POST payload analysis**: the engine will intercept the contents transmitted using the POST method.

- **HTTPS and Compression**: since the engine is embedded in the web server, it gets access to request data after decryption and decompression takes place.

Features (IV)

- **Audit logging**: full details of every request (including POST) can be logged for later analysis.

For doing an effective forensic investigation in a web attack, we need at least:

- Source IP
- Time stamp
- HTTP method
- URI requested
- Full Http Data

Special built-in checks

- URL encoding validation
- Unicode encoding validation
- Byte range verification [0-255] detect and reject :

- Shellcode

- Cross Request Form Forgerie (CR,LF)

- If the language used in our application is English we can limit the byte range to [32-126].

Rules

- What is a rule?
- Rules are formed using regular expressions
- Any number of custom rules supported
- Also negated rules supported
- Analyses:
  - Headers
  - Environment variables
  - Server variables
  - Individual cookies
  - Individual page variables
  - POST payload
  - Script output

Actions (I)

- What are the actions?
- Reject request with status code [403,500, ..]
- Reject request with redirection
- Execute external binary on rule match
- Log request

Actions (II)

- Stop rule processing and let the request through
- Rule chaining
- Skip next n rules on match
- Pauses for a number of milliseconds

File Upload

- Intercept files being uploaded through the web server
- Store uploaded files on disk
- Execute an external script to approve or reject files (ClamAV anti-virus)

Other

- Change the identity of the web server: We can change the Server header to whatever we want, also will change the version in all server messages, like error, forbidden, etc.
- Easy to use internal chroot functionality.

How does it works

1. Parse the request.
2. Perform canonicalization and anti-evasion actions.
3. Perform special built-in checks.
4. Execute input rules:
   - If the request is allowed, then it will reach the handler where it executes.
   - After the request:
     1. Execute output rules.
     2. Log the request.

Security models

- A security is the posture we take at the time of setting rules for our protection systems.

- There are two security models:
  - Positive Model
  - Negative Model

Security models: Positive Model

- We allow what we know is right (safe).
- Like network firewall model “Deny All - Allow what we need”.
- Pros:
  - Better performance
  - Less false positives
- Cons:
  - More time to implement, we need to identify all the scripts in the application, and create rules for them.
- Example:
  - Page log.asp, the field Login could only accept characters [a-zA-Z0-9] and could be 12 char long.

Security Models: Negative Model (I)

- Deny what is dangerous
- Do we know everything that is bad for our applications?
- Pros:
  - Less time to implement, we create a general that affect the whole application.
- Cons:
  - More false positives
  - More processing time
- Example XSS:
  - There are a lot of tags and places to look for XSS, we can miss some of them leaving a hole in our application.

Security Models: Negative Model (II)

<object>...</object>

<embed>...</embed>

<applet>...</applet>

<script>...</script>

<script src="..."></script>

<iframe src="...">

<img src="javascript:...">

<b onMouseOver="...">

&{...};

Configuration

- Per Context configuration:

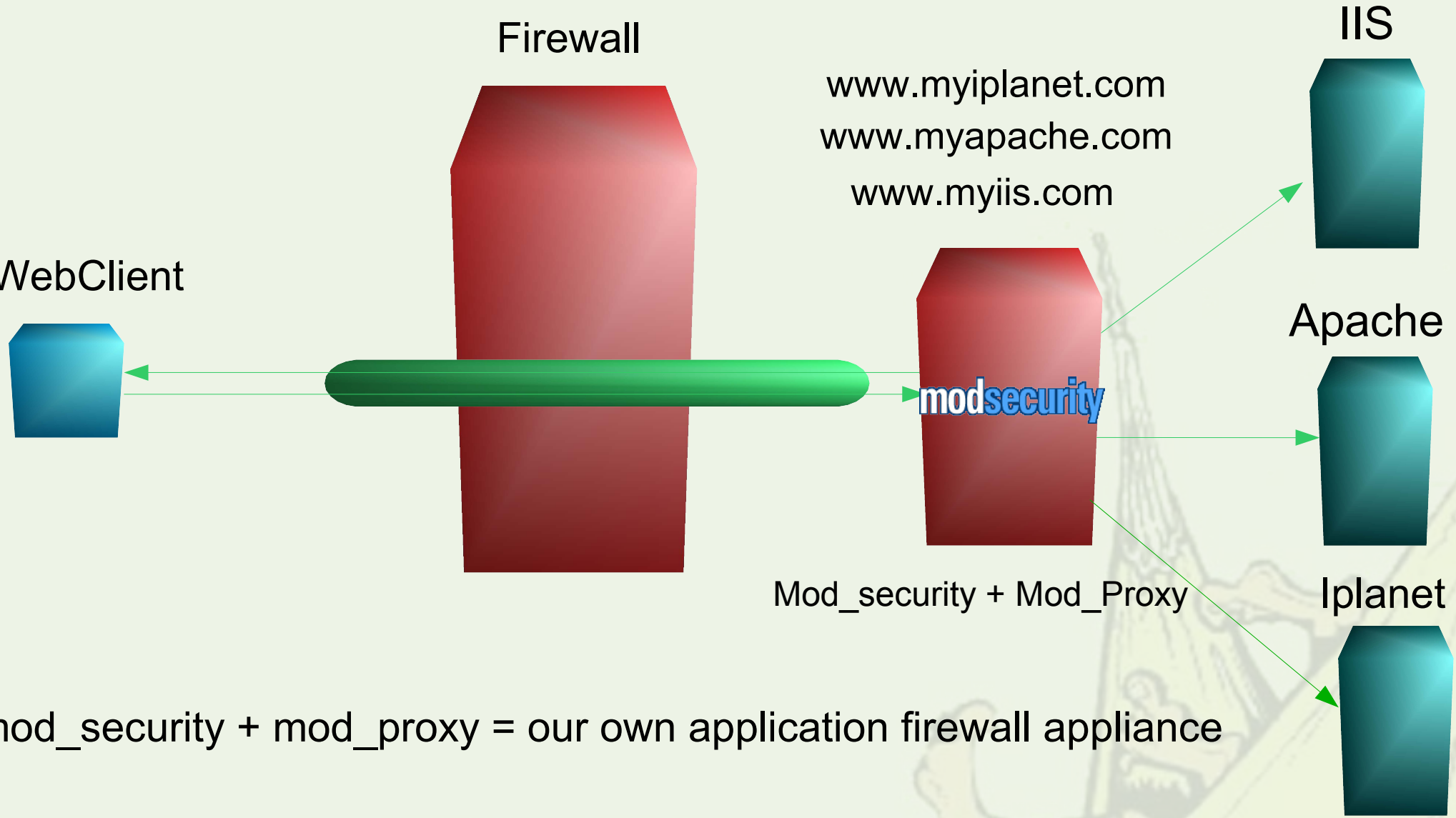
```
Main Server
   Virtual Host
Directory
```

- Inheritance: Virtual Host and Directory could Inherit the configuration of the Main Server. Same for Directory with Virtual Host.

Thoughts...

- What about my web server, i don't use Apache...

Reverse Proxy Model (I)



mod_security + mod_proxy = our own application firewall appliance

Reverse Proxy Model (II)

Advantages:

- Single point of access
- Increased Performance
- Network Isolation
- Network Topology Hidden from outside
- Protect any Web server, IIS, Iplanet, etc.

Disadvantages:

- Single point of Failure
- Increased Complexity

Extending Features

- Strip Comments
- Cookie cripto-signing
- Link cripto-signing
- Hidden field signing
- Web based logs console

- Some of these features are present in commercial firewalls, so we thought it will be great if an Open Source project like Modsecurity could do the same.

Strip Comments (I)

- Using the Libxml2 it allow us to build an HTML tree of the parsed code sended to the user.
- Having that, we could walk the tree looking for comments

<!-- xxxxxx ->

and cut them off.

- So just adding the directive:

  SecStripCommentCode On/Off

Strip Comments (II)

- Modsecurity will clean all commented code before sending the requested page to the user.
- We considered the common situation of commented script code to allow backward compatibility with old browsers. This comments are not stripped.
- We are working in the deletion of comments inside the code of different script languages (javascript, tcl, etc).

Cookie Signing

- Another feature is the cripto-signing of cookies, to prevent tampering (cookie poisoning, session fixation)
- With directives:

  SecSignCookies On/Off

  SecEncriptionPassword “password”
- We are using Cryptlib and do the cripto-sign with the Advanced Encryption Standard (AES) algorithm.
- Still working on it

Link Signing (I)

- As an extra layer of security we have added the option to sign all links browsables from "Entry points", so users are able only to follow the "intended flow" of the application.

- Again we use Cryptlib, to sign the links with the AES algorithm.

- The directives to control this:

  SecSignLinks On/Off
  SecEntryPoint "/index.asp" "/images" ... "entry page n"
  SecEncriptionPassword "password"

Link Signing (II)

- **Signing the links we can tackle this threats:**

  - Predictable Resource Location
  - Forceful Browsing: the attacker "forces" a URL by accessesing it directly instead of following links.

- Automated scanners like Nikto, dirb, and others, will be foiled.
- Example of a link cripto-signed:
- http://www.securesign.com/help.asp?pagina=ayuda-login.asp&**Secsign=MIHVBgkqhkiG9w0BBwOggccwgcQCAQAxcaNvAgEAoBsGCSqGSIb3DQEFDDAOBAhyxt2Mf3s4KQICAfQwIwYLKoZ...6DQDpC**

Form and Hidden Fields cripto-signing

Another security measure is the cripto-signing of form hidden fields and signing the forms itself, to prevent the values from being modified in the quantity, names, etc. of the inputs of the form.

Log Console (I)

- Web based Log Console
- Facilitate the log analysis task
- Snort Acid Style
- Some of the listing that it provides:
  - Top 10 attacks or attackers
  - Today attacks
  - Last 15 Attacks
- Other characteristics:
  - Attack details
  - Search
  - Delete records

Log Console (II)

Modsecurity Log - Mon Jul 18 16:50:19 CEST 2005 - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://

Modsecurity Logs for Laramies

Generated Mon Jul 18 16:50:19 CEST 2005.
1047 entries found.

- Home
- Today's Attacks
- Top 10 Attacks
- Last 15 Attacks
- Top 10 Source IPs

Main

Showing page **1** of **53** pages [Next] [Last Page]

20 Rows

#	Date	Source	Destination	Method	Url	Modsecurity Message	Action	Del
1055	2005-07-17 05:38:05	218.22.128.35	[redacted]	POST	/_vti_bin/_vti_aut/fp30reg.dll	Access denied with code 500. Pattern match "!^$" at HEADER	500	
1054	2005-07-13 13:58:50	82.57.91.193	[redacted].220.11	HEAD	/scripts/..%fc%80%80%80%80%af../winnt/system32 ...	Error normalizing REQUEST_URI: Invalid Unicode encoding: overlong ...	500	
1053	2005-07-13 13:58:44	82.57.91.193	[redacted].220.11	HEAD	/scripts/..%f8%80%80%80%af../winnt/system32/cm ...	Error normalizing REQUEST_URI: Invalid Unicode encoding: overlong ...	500	
1052	2005-07-13 13:58:37	82.57.91.193	[redacted].220.11	HEAD	/scripts/..%f0%80%80%af../winnt/system32/cmd.e ...	Error normalizing REQUEST_URI: Invalid Unicode encoding: overlong ...	500	
1051	2005-07-13 13:58:30	82.57.91.193	[redacted].220.11	HEAD	/scripts/..%e0%80%af../winnt/system32/cmd.exe? ...	Error normalizing REQUEST_URI: Invalid Unicode encoding: overlong ...	500	
1050	2005-07-13 13:58:17	82.57.91.193	[redacted].220.11	HEAD	/scripts/..%c1%af../winnt/system32/cmd.exe?/c+ ...	Error normalizing REQUEST_URI: Invalid Unicode encoding: overlong ...	500	
1049	2005-07-13 13:58:11	82.57.91.193	[redacted].220.11	HEAD	/scripts/..%c1%9f../winnt/system32/cmd.exe?/c+ ...	Error normalizing REQUEST_URI: Invalid Unicode encoding: overlong ...	500	
1048	2005-07-13 13:58:04	82.57.91.193	[redacted].220.11	HEAD	/scripts/..%c1%9f../..%c1%9f../..%c1%9f../winn ...	Error normalizing REQUEST_URI: Invalid Unicode encoding: overlong ...	500	
1047	2005-07-13 13:57:58	82.57.91.193	[redacted].220.11	HEAD	/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+ ...	Error normalizing REQUEST_URI: Invalid Unicode encoding: overlong ...	500	
1046	2005-07-13 13:57:52	82.57.91.193	[redacted].220.11	HEAD	/scripts/..%C1%9C..%C1%9C..%C1%9C..%C1%9Cwinnt ...	Error normalizing REQUEST_URI: Invalid Unicode encoding: overlong ...	500	
1045	2005-07-13 13:57:45	82.57.91.193	[redacted].220.11	HEAD	/scripts/..%c1%9c..%c1%9c..%c1%9c..%c1%9c../wi ...	Error normalizing REQUEST_URI: Invalid Unicode encoding: overlong ...	500	
1044	2005-07-13 13:57:31	82.57.91.193	[redacted].220.11	HEAD	/scripts/..%c1%1c../winnt/system32/cmd.exe?/c+ ...	Error normalizing REQUEST_URI: Invalid Unicode encoding: invalid b ...	500	
1043	2005-07-13 13:57:25	82.57.91.193	[redacted].220.11	HEAD	/scripts/..%c1%1c../..%c1%1c../..%c1%1c../winn ...	Error normalizing REQUEST_URI: Invalid Unicode encoding: invalid b ...	500	
1042	2005-07-13 13:57:18	82.57.91.193	[redacted].220.11	HEAD	/scripts/..%C1%1C..%C1%1C..%C1%1C..%C1%1Cwinnt ...	Error normalizing REQUEST_URI: Invalid Unicode encoding: invalid b ...	500	
1041	2005-07-13 13:57:11	82.57.91.193	[redacted].220.11	HEAD	/scripts/..%c1%1c..%c1%1c..%c1%1c..%c1%1c../wi ...	Error normalizing REQUEST_URI: Invalid Unicode encoding: invalid b ...	500	
1040	2005-07-13 13:56:59	82.57.91.193	[redacted].220.11	HEAD	/scripts/..%c0%af../winnt/system32/cmd.exe?/c+ ...	Error normalizing REQUEST_URI: Invalid Unicode encoding: overlong ...	500	
1039	2005-07-13 13:56:54	82.57.91.193	[redacted].220.11	HEAD	/scripts/..%c0%af../..%c0%af../..%c0%af../winn ...	Error normalizing REQUEST_URI: Invalid Unicode encoding: overlong ...	500	

Done Proxy: None

Log Console (III)

Modsecurity Log - Mon Jul 18 17:07:26 CEST 2005 - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://

Modsecurity Logs for Laramies

Generated Mon Jul 18 17:07:26 CEST 2005.

- Home
- Today's Attacks
- Top 10 Attacks
- Last 15 Attacks
- Top 10 Source IPs

Details

Attribute	Value
#	73
user_ip	192.168.1.8
request_method	GET
connection	close
destination	192.168.1.100
user_agent	Mozilla/4.75 (Nikto/1.35)
url	/a.jsp/<script>alert('Vulnerable')</script>
url(decoded)	/a.jsp/<script>alert('Vulnerable')</script>
server_protocol	HTTP/1.0
redirect_status	500
content_length	613
handler	(null)
content_type	text/html; charset=iso-8859-1
mod_security_message	Access denied with code 500. Pattern match "<[[:space:]]*script" at THE_REQUEST
mod_security_action	500
date	2005-06-28 00:00:03

Done Proxy: None

Open Proxy Honeypots (I)

- **Web App Security Consortium (WASC)**
- This project will use one of the web attacker's most trusted tools against him - the Open Proxy server. Instead of being the target of the attacks, we opt to be used as a conduit of the attack data in order to gather our intelligence. By deploying multiple, specially configured open proxy server (or proxypot), we aim to take a birds-eye look at the types of malicious traffic that traverse these systems.

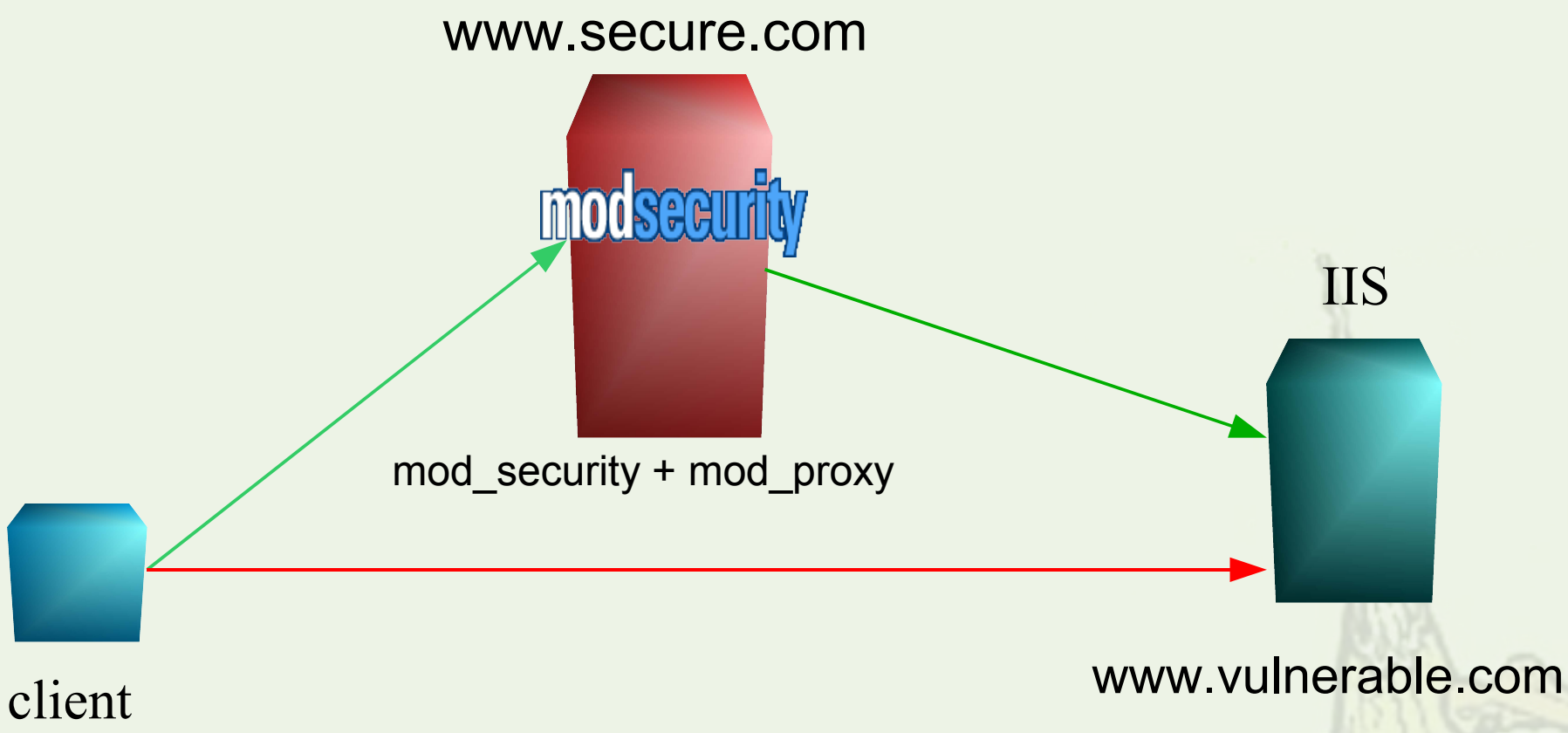
Open Proxy Honeypots (II)

- The honeypot systems will conduct real-time analysis on the HTTP traffic to categorize the requests into threat classifications outlined by the Web Security Threat Classification and report all logging data to a centralized location.

Examples

- SQL Injection
- XSS Scripting
- Fine grained checks
- Buffer Overflows
- Positive Model
- File extensions
- Output filtering, errors.
- Resource location prediction
- Strip comment code
- Blog/Forums spam protection

Example Scenario



SQL Injection

- **Vulnerable parameter:** Login
- **Test String:**

  ' or 1=1;--

- **Modsecurity rule:**

  Secfilter '.+-- redirect:http://www.secure.com/error123.html

- **Positive way:**

  SecfilterSelective ARG_login !^[a-zA-Z]+$

XSS Prevention

- **Vulnerable parameter:** pagina
- **Test String:**

  <script>alert('Next time I will eat your cookies ');</script>
- **Hex Encoded:**

  %3C%73%63%72%69%70%74%3E%61%6C%65%72%74%28
  %27%4E%65%78%74%20%74%69%6D%65%20%49%20%77
  %69%6C%6C%20%65%61%74%20%79%6F%75%72%20%63
  %6F%6F%6B%69%65%73%20%27%29%3B%3C%2F%73%63
  %72%69%70%74%3E
- **Modsecurity rule:**

  Secfilter “<(.|\n)+>” redirect:http://www.secure.com/error.html

Fine Grained (I) Positive Model

- **Vulnerable parameter:** login
- **Test String:**

  Test12-
  121212
- **Modsecurity rule:**

  SecfilterSelective ARG_login !^[a-zA-Z]+$

  Look at the server version!

File Extension Protection

- **File extension to protect:** *.txt
- **Requested file**: admin/password.txt
- **Modsecurity rule**:

  SecfilterSelective SCRIPT_FILENAME .+\.txt$

  Positive way: SCRIPT_FILENAME !.+\.asp$

If we use Link Signing we don't need to worry about this rules.

Output Errors Protection

- **Vulnerable parameter**: password
- **Test String**:

  ' (single quote) and 1 char in the login because one of the rules before.

- **Modsecurity rule**:

  SecfilterSelective OUTPUT “Microsoft OLE DB Provider” “redirect:http://www.secure.com/default.asp”

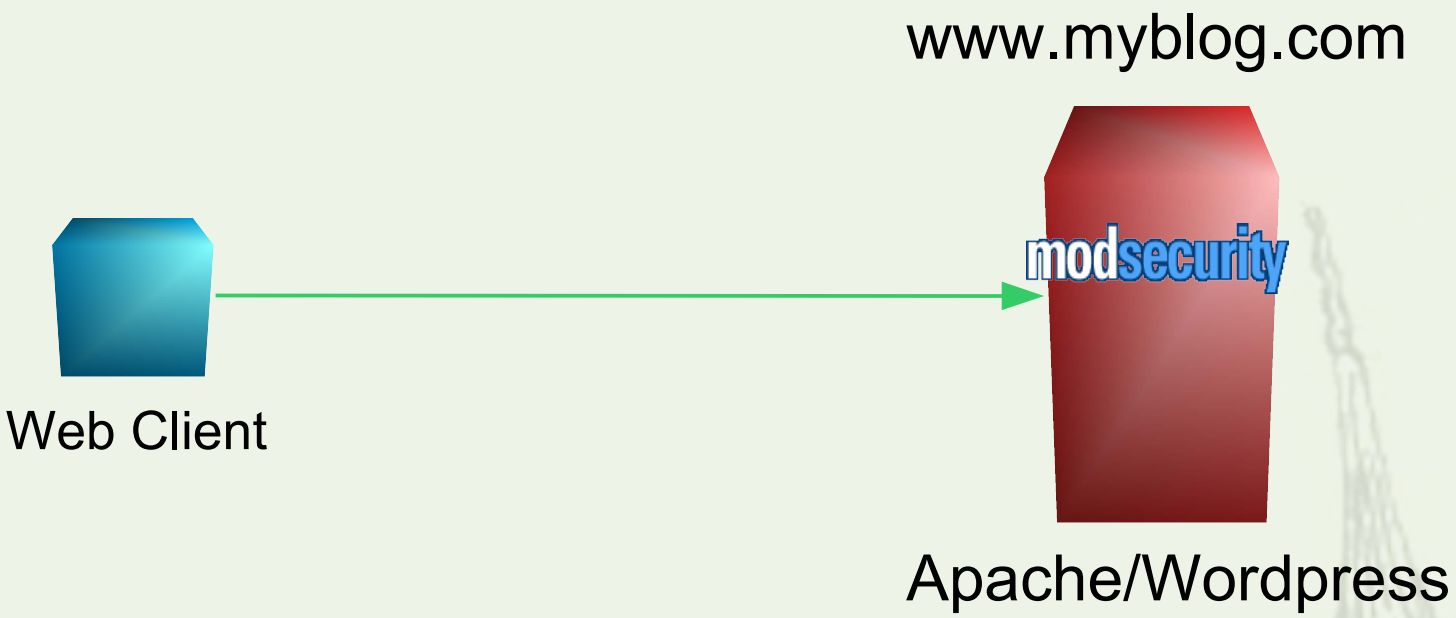
Strip Comment Code

- **Vulnerable page:** help.asp
- **Commented code:**

  <!-- remember that admin zone is in /admin/ -->
- **Modsecurity option:**

  SecStripCommenCode On

Blog Spam Scenario, Wordpress



Here Modsecurity is working in the same server that is protecting.

Blog/Forums Spam Protection

- **Vulnerable parameter: comments**
- **Spam message:**

  “Cheap viagra”

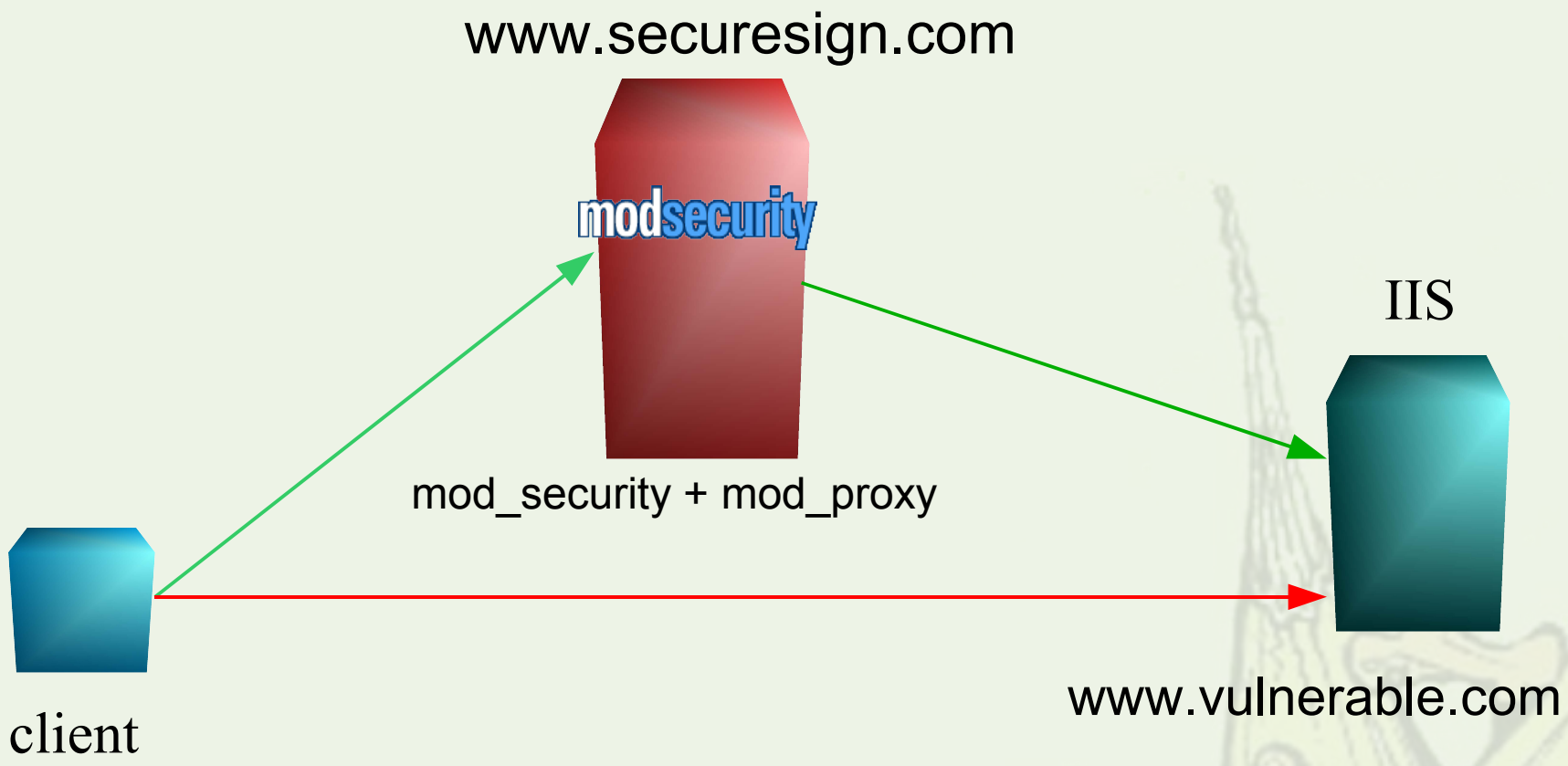
  “Cheap vLagra”

  “Cheap v1agra”

- **Modsecurity Rule:**

  Secfilter ARGS “v[iIL1]agra”
  redirect:http://www.myblog.com/spam.swf

Link Signing Activated



Resource Location Prediction

- **Attack tool:** Nikto
- **Attack options:**
  - nikto -host www.vulnerable.com
  - nikto -host www.securesign.com

- **Modsecurity options:**
  - SecSignLinks On
  - SecEncriptionPassword “our_Pass*Word-$”

Resource Location using Nikto (I)

```
[redacted]@salvacion: /home/[redacted]/tools/cgiscanners/nikto-1.34
[redacted]@salvacion:~/tools/cgiscanners/nikto-1.34$ perl nikto.pl -h www.vulnerable.c
om
---------------------------------------------------------------------------
- Nikto 1.34/1.29     -     www.cirt.net
+ Target IP:        10.10.0.114
+ Target Hostname:  www.vulnerable.com
+ Target Port:      80
+ Start Time:       Tue Jul 26 11:44:17 2005
---------------------------------------------------------------------------
- Scan is dependent on "Server" string which can be faked, use -g to override
+ Server: Microsoft-IIS/6.0
- Server did not understand HTTP 1.1, switching to HTTP 1.0
+ Server does not respond with '404' for error messages (uses '400').
+     This may increase false-positives.
- Retrieved X-Powered-By header: ASP.NET
+ Allowed HTTP Methods: OPTIONS, TRACE, GET, HEAD
+ HTTP method 'TRACE' is typically only used for debugging. It should be disabled.
+ Microsoft-IIS/6.0 appears to be outdated (4.0 for NT 4, 5.0 for Win2k)
+ //admin/aindex.htm - FlexWATCH firmware 2.2 is vulnerable to authentication bypass
 by prepending an extra '/'. http://packetstorm.linuxsecurity.com/0310-exploits/Flex
WATCH.txt (GET)
+ /admin/ - This might be interesting... (GET)
+ 2646 items checked - 2 item(s) found on remote host(s)
+ End Time:         Tue Jul 26 11:45:15 2005 (58 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
[redacted]@salvacion:~/tools/cgiscanners/nikto-1.34$
```

Resource Location using Nikto (II)

```
[redacted]@salvacion: /home/[redacted]/tools/cgiscanners/nikto-1.34
[redacted]@salvacion:~/tools/cgiscanners/nikto-1.34$ perl nikto.pl -h www.securesign.c
om
---------------------------------------------------------------------------
- Nikto 1.34/1.29     -     www.cirt.net
+ Target IP:        10.10.0.120
+ Target Hostname:  www.securesign.com
+ Target Port:      80
+ Start Time:       Tue Jul 26 11:50:32 2005
---------------------------------------------------------------------------
- Scan is dependent on "Server" string which can be faked, use -g to override
+ Server: Microsoft II$
- Server did not understand HTTP 1.1, switching to HTTP 1.0
+ Server does not respond with '404' for error messages (uses '400').
+     This may increase false-positives.
+ All CGI directories 'found', use '-C none' to test none
- Retrieved X-Powered-By header: ASP.NET
+ 2329 items checked - 0 item(s) found on remote host(s)
+ End Time:         Tue Jul 26 11:51:30 2005 (58 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
[redacted]@salvacion:~/tools/cgiscanners/nikto-1.34$
```

XSS with cripto-sign links enabled

- **Vulnerable parameter:** pagina
- **Test String:**

  <script>alert('Next time I will eat your cookies ');</script>
- **Modsecurity rule:**

  Secfilter “<(.|\n)+>”

Same for all other type of injection and variable manipulation, that involves a link, or direct access to URL.

File Extensions with Sign Links On

- **Vulnerable file extension:** *.txt
- **Requested file**: /admin/password.txt
- **Modsecurity rule**:

SecfilterSelective SCRIPT_FILENAME .+\.txt$
Positive way: SCRIPT_FILENAME !.+\.asp$

We don't need to worry about this rules anymore!

Conclusions

- Modsecurity is a great choice for protecting your web applications
- Easy to configure
- Very effective
- Remember that Modsecurity is an extra layer in our protection scheme, we have to secure our applications whenever we can.
- There is still much work to do to improve this features, they are an alpha version.

Question, doubts?

References

Download Modsecurity:
www.modsecurity.org

Mailing List:
mod-security-users@lists.sourceforge.net

Cryptlib:
http://www.cs.auckland.ac.nz/~pgut001/cryptlib/

Libxml2:
http://xmlsoft.org/

Thank you!

modsecurity

Advanced web application defense with Modsecurity

Daniel Fernandez Bleda
dfernandez@isecauditors.com

Christian Martorella
cmartorella@isecauditors.com

Coding is no easy
But programmers are patient