An Introduction to ModSecurity

Securing your Apache Web Applications

A. Crowell J. St. John

iSEC Partners

LinuxFest NW 2013

iSECpartners
part of nccgroup

Outline

iSECpartners part of nccgroup

History of ModSecurity

- Created in 2004
- Originally for Apache, now additionally for nginx and IIS
- Stable releases for Apache/IIS, release candidate for nginx

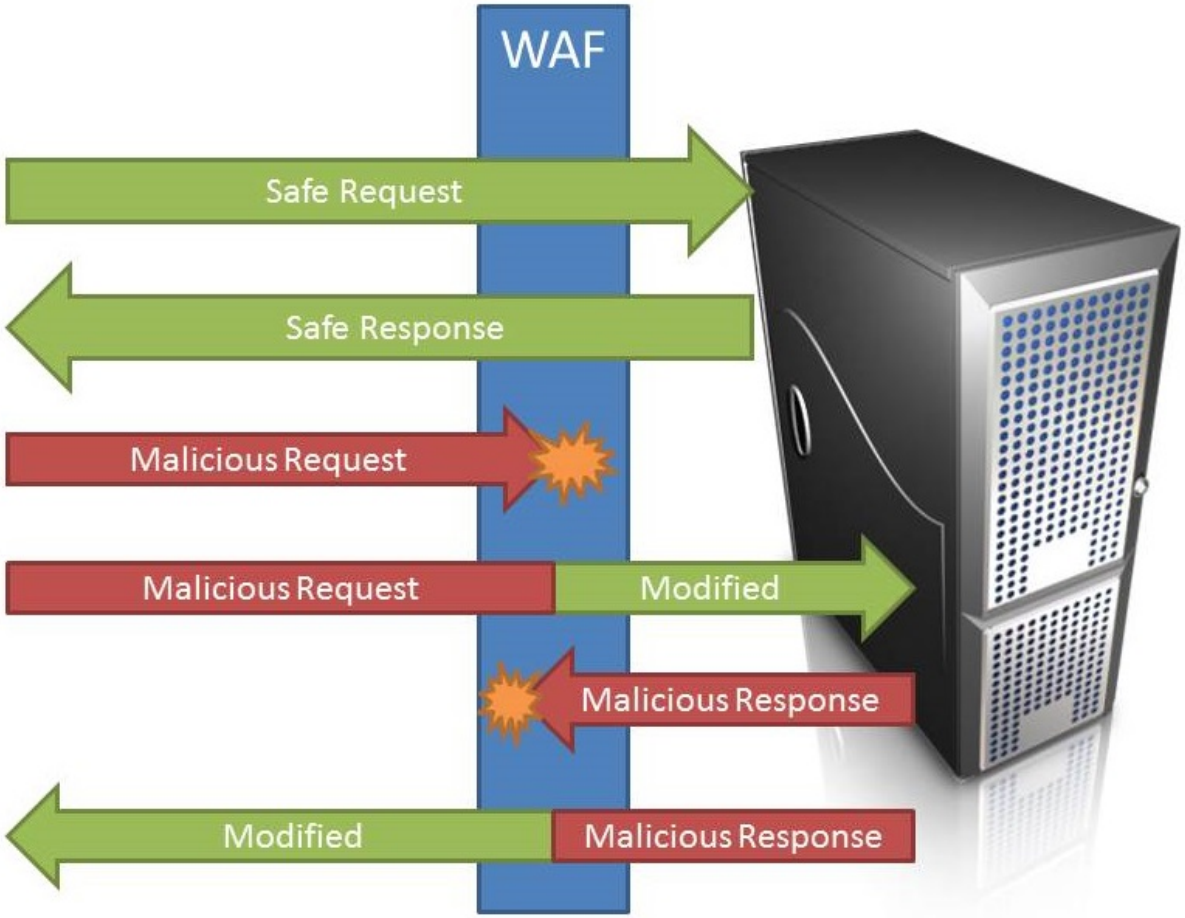
iSECpartners
part of nccgroup

What is a web application firewall?

- Layer of protection between web server and outside world
- Generally intercepts web server traffic and can
  - prevent attacks by denying or transforming malicious content
  - perform logging to identify attackers
  - use local storage and scripts to perform more sophisticated tasks

iSECpartners
part of nccgroup

What is a web application firewall?



iSECpartners
part of nccgroup

What does ModSecurity provide?

- A powerful rule language
- Selective filtering and transformation capabilities
- Extensive logging options
- Embedded and reverse proxy modes of operation
- But... no protection on its own

- Flexibility is double-edged sword
  - Extremely powerful
  - But hard to do correctly

iSECpartners
part of nccgroup

What does ModSecurity provide?

- A powerful rule language
- Selective filtering and transformation capabilities
- Extensive logging options
- Embedded and reverse proxy modes of operation
- But... no protection on its own

- Flexibility is double-edged sword
  - Extremely powerful
  - But hard to do correctly

iSECpartners part of nccgroup

General Setup Information

- Can be installed by
  - Package managers in Debian/Redhat based Linux distros
  - Binary installer for IIS on Windows
  - Third party binary packages (available on ModSecurity website)
  - Source

iSECpartners part of nccgroup

Ubuntu Installation Guide

via Package Manager

- Easy: apt-get install libapache2-modsecurity
- Done!

iSECpartners
part of nccgroup

Fedora Installation Guide

via Package Manager

- yum install httpd mod_security
- edit /etc/httpd/conf/httpd.conf by adding line:
  - LoadModule security2_module modules/mod_security2.so
- sudo service httpd restart

iSECpartners part of nccgroup

Installation from Source

Dependencies:

- Apache 2.0.x
- mod_uniqueid
- libapr and libapr-util
- libpcre
- libxml2

Optional libraries:

- liblua 5.1.x - for ModSecurity Lua engine
- libcurl 7.15.1+ - if using ModSecurity Log Collector

iSECpartners part of nccgroup

Adding ModSecurity to Apache

Add to apache.conf:

- LoadFile /usr/lib/libxml2.so
- LoadFile /usr/lib/liblua5.1.so
- LoadModule security2_module modules/mod_security.so
- Include /opt/modsecurity/etc/modsecurity.conf

iSECpartners part of nccgroup

Enabling ModSecurity

Define rule locations in modsecurity.conf:

```
<IfModule mod_security2.c>
  Include /opt/modsecurity/etc/<your-rule-1>.conf
  Include /opt/modsecurity/etc/<your-rule-2>.conf
  ...
</IfModule>
```

Finally, enable ModSecurity in modsecurity.conf:

- SecRuleEngine Enabled

iSECpartners part of nccgroup

Rules & Traffic

- Rules made up of 4 parts:
  - variables
  - operators
  - transformations
  - actions
- Traffic has 5 phases of processing, different data available
  - phase 1 -- request headers
  - phase 2 -- request body
  - phase 3 -- response headers
  - phase 4 -- response body
  - phase 5 -- logging
- Rules specify in which phase they act

iSECpartners part of nccgroup

Rules & Traffic

- Rules made up of 4 parts:
  - variables
  - operators
  - transformations
  - actions
- Traffic has 5 phases of processing, different data available
  - phase 1 -- request headers
  - phase 2 -- request body
  - phase 3 -- response headers
  - phase 4 -- response body
  - phase 5 -- logging
- Rules specify in which phase they act

iSECpartners part of nccgroup

Rules & Traffic

- Rules made up of 4 parts:
  - variables
  - operators
  - transformations
  - actions
- Traffic has 5 phases of processing, different data available
  - phase 1 -- request headers
  - phase 2 -- request body
  - phase 3 -- response headers
  - phase 4 -- response body
  - phase 5 -- logging
- Rules specify in which phase they act

iSECpartners part of nccgroup

Variables

```
SecRule ARGS "<script>" t:lowercase log,deny,status:403
```

- Identify pieces of the transaction for the rule to work with
- Made available by ModSecurity
- Examples:
  - REMOTE_ADDR
  - ARGS
  - FILES
  - REQUEST_BODY, REQUEST_COOKIES, REQUEST_METHOD
  - RESPONSE_BODY, RESPONSE_HEADER, RESPONSE_STATUS

iSECpartners part of nccgroup

Operators

```
SecRule ARGS "<script>" t:lowercase log,deny,status:403
```

- Specify how variables are analyzed
- Most commonly regular expressions
- Examples:
  - string matching (`@beginsWith`, `@rsub`, `@rx`)
  - numerical (`@eq`, `@ge`, `@gt`)
  - validation (`@validateByteRange`, `@validateSchema`, `@validateUrlEncoding`)
  - miscellaneous (`@geoLookup`, `@verifyCC`, `@ipMatch`)

iSECpartners part of nccgroup

Transformations

```
SecRule ARGS "<script>" t:lowercase log,deny,status:403
```

- Can transform (modify) variable before the operator runs
- Examples:
  - `base64decode`, `base64encode`
  - `length`
  - `lowercase`
  - `sha1`, `md5`

iSECpartners part of nccgroup

Actions

```
SecRule ARGS "<script>" t:lowercase log,deny,status:403
```

- specify what happens when a rule matches
- have different properties:
  - are disruptive (`allow`, `block`, `deny`, `drop`, `proxy`, `pass`, `redirect`)
  - affect rule flow (`chain`, `skip`, `skipAfter`)
  - affect metadata (`id`, `phase`, `msg`, `rev`, `severity` `tag`)
  - affect variables (`capture`, `deprecatevar`, `setvar`, `setuid`)

iSECpartners part of nccgroup

Simple Blacklist Entry

- `SecRule ARGS "@contains <script>"`
- But what about <ScRiPt>? <SCRIPT >? etc?
- Enter transformations:

```
SecRule ARGS "@contains <script>" t:lowercase,t:removeWhitespace
```

iSECpartners part of nccgroup

Simple Blacklist Entry

- `SecRule ARGS "@contains <script>"`
- But what about <ScRiPt>? <SCRIPT >? etc?
- Enter transformations:

```
SecRule ARGS "@contains <script>" t:lowercase,t:removeWhitespace
```

iSECpartners part of nccgroup

Blacklist Evasion

- Never-ending problem
- As the web standard evolves, we get new injection vectors
- Methodology itself is flawed
- But we can try...

iSECpartners part of nccgroup

Blacklist Evasion

- Never-ending problem
- As the web standard evolves, we get new injection vectors
- Methodology itself is flawed
- But we can try...

iSECpartners part of nccgroup

ModSecurity Core Rule Set

- Managed by the folks at OWASP
- Fairly easy to install via package manager
- Rules designed to cover:
  - Cross-site scripting
  - SQL Injection
  - Much, much more

iSECpartners part of nccgroup

ModSecurity Example Rule

```
SecRule ARGS "(?i)(<script[^>]*>[\s\S]*?<\/script[^>]*>|<script[^>]*>[\s\S]*?<\/
    script[[\s\S]]*[\s\S]|<script[^>]*>[\s\S]*?<\/script[\s]*[\s]|<script
    [^>]*>[\s\S]*?<\/script|<script[^>]*>[\s\S]*?)" "id:'973336',phase:2,rev
    :'1',ver:'OWASP_CRS/2.2.7',maturity:'1',accuracy:'8',t:none,t:urlDecodeUni,
    t:htmlEntityDecode,t:jsDecode,t:cssDecode,log,capture,msg:'XSS Filter -
    Category 1: Script Tag Vector',tag:'OWASP_CRS/WEB_ATTACK/XSS',tag:'WASCTC/
    WASC-8',tag:'WASCTC/WASC-22',tag:'OWASP_TOP_10/A2',tag:'OWASP_AppSensor/IE1
    ',tag:'PCI/6.5.1',logdata:'Matched Data: %{TX.0} found within %{
    MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.msg
    }',setvar:tx.xss_score=+%{tx.critical_anomaly_score},setvar:tx.
    anomaly_score=+%{tx.critical_anomaly_score},setvar:tx.%{rule.id}-OWASP_CRS/
    WEB_ATTACK/XSS-%{matched_var_name}=%{tx.0}"
```

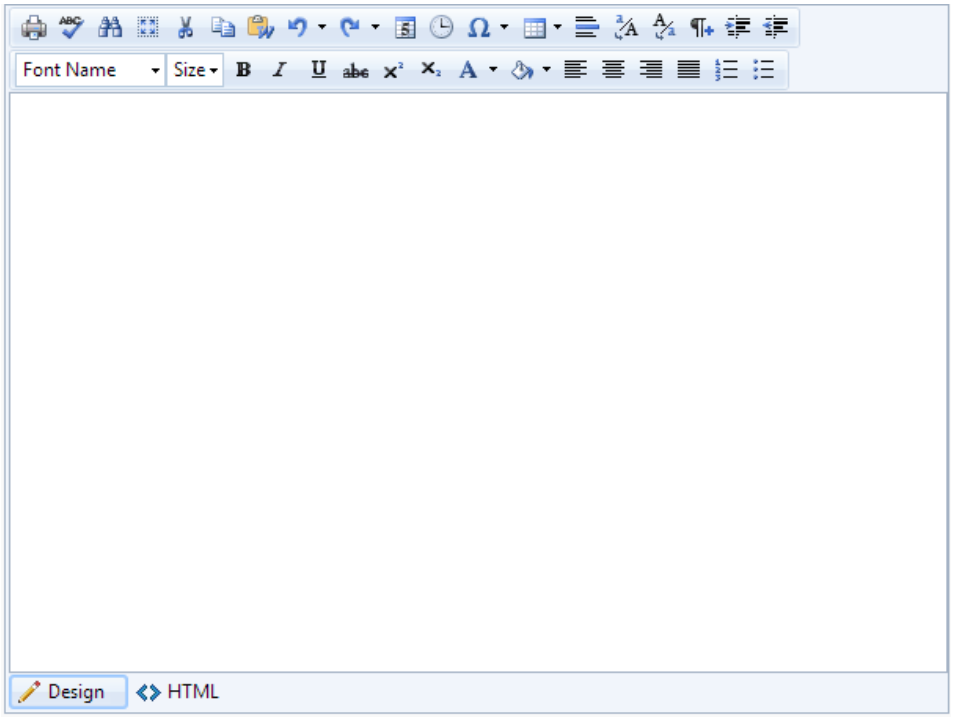
Whitelist Methodology

- Not the easiest in the short term
- More comprehensive
- New attack vectors less likely to break your configuration

iSECpartners part of nccgroup

Whitelist Example

HTML Editor



- A hard problem
- Don't always trust users, but want to allow some HTML content tags

iSECpartners part of nccgroup

Whitelist Example

HTML Editor

Input:

```
something <strong>bold</strong>
```

Submit

Output:

localhost/test4.html?code=something bold<%2Fstrong>

Code: something **bold**

iSECpartners part of nccgroup

Whitelist Example

HTML Editor

The obvious problem:

```
<img src=a onerror=alert(1)
```

Results in:

iSECpartners part of nccgroup

Whitelist Example

HTML Editor

ModSecurity to the rescue:

```
#Handle <img src=... differently. If we find a match, skip to the end and pass
SecRule ARGS:code "@rx <img src=([a-zA-Z0-9:/.\-\+]+)>" \
        "t:none,t:lowercase,t:compressWhitespace,skipAfter:whitelistMarker,pass"

#Do not allow any attributes on tags, restrict to <word
SecRule ARGS:code "@rx <(\w+)\ " "t:none,t:lowercase,t:compressWhitespace"

#Capture the word and match against several whitelist values
SecRule ARGS:code "@rx <(\w+)" "t:none,t:lowercase,t:compressWhitespace,capture,
     chain"
SecRule TX:1 "!@rx ^a$|^div$|^td$|^tr$|^br$|^b$|^strong$"

SecMarker whitelistMarker
```

Whitelist Example

HTML Editor

Now we get:

Logging Capabilities

- Debug Logging
  - Used to see how rules are behaving
  - 9 levels (*nothing* to *warnings* to *everything*)
  - Uses lots of storage (7KB per transaction)
- Audit Logging
  - Main goal - ability to log full transactions
  - Amount of data logged configurable
  - Serial or concurrent
- Remote Logging
  - Send logs to remote server
  - Is *secure*, *efficient*, *reliable*, & *buffered*

iSECpartners part of nccgroup

Logging Capabilities

- Debug Logging
  - Used to see how rules are behaving
  - 9 levels (*nothing* to *warnings* to *everything*)
  - Uses lots of storage (7KB per transaction)
- Audit Logging
  - Main goal - ability to log full transactions
  - Amount of data logged configurable
  - Serial or concurrent
- Remote Logging
  - Send logs to remote server
  - Is *secure*, *efficient*, *reliable*, & *buffered*

iSECpartners part of nccgroup

Logging Capabilities

- Debug Logging
  - Used to see how rules are behaving
  - 9 levels (*nothing* to *warnings* to *everything*)
  - Uses lots of storage (7KB per transaction)
- Audit Logging
  - Main goal - ability to log full transactions
  - Amount of data logged configurable
  - Serial or concurrent
- Remote Logging
  - Send logs to remote server
  - Is *secure*, *efficient*, *reliable*, & *buffered*

iSECpartners part of nccgroup

Logging in Rules

- Dynamically choose what to log (`auditLogParts=ABCDEFGH`)
- Can add audit parts based on severity (`HIGHEST_SEVERITY`)
- Sanitize sensitive data:
  - `sanitizeArg:password`
  - `sanitizeRequestHeader:Authorization`
  - `SecRule ARG_NAMES password "phase:5,nolog,pass,sanitizeMatched"`

iSECpartners part of nccgroup

Performance Implications

- Parsing - not much more than Apache
- Buffering - uses "a lot of" RAM
- Rule processing - will use CPU (fewer rules the better)
- Logging - performance wise, not much. Storage could be a lot if doing full audit logging.

iSECpartners part of nccgroup

Summary

- ModSecurity is great for handling attacks outside of app.
- The learning curve is steep.
- But...is a good resource when done correctly.

iSECpartners
part of nccgroup

Questions?

https://www.isecpartners.com

iSECpartners
part of nccgroup

For Further Reading I

I. Ristić
ModSecurity Handbook.
Feisty Duck Limited, 2012.

iSECpartners
part of nccgroup