Web Application Firewall Technology Insight

Joakim Sandström
Role: OWASP Prospect

OWASP
22/2-2007

The OWASP Foundation
http://www.owasp.org

Web Application Firewalls (WAF)

- Introduction
  - Essentials
- Protection mechanisms
  - White- & Blacklisting
  - Virtualisation
  - other..
- mod_security
  - Configuration
  - Whitelisting
  - XML schema & dtd validation
- WAF Definition & Addvalue etc.
- Evaluating criterias
- Benefits
- Drawbacks
- Other..

Introduction

- Web Application Firewalls because..
  - "Most application deployed today are insecure because the average developer is still not trained enough."

    Ivan Ristic

Introduction

- Web Application Firewalls
  - Interpose themselves between the "web server" and the user-side client.
  - Hereby intercept all http queries between the client and server.
  - Analyze the traffic based on both blacklisting & white listing rules. – hereby blocks the "bad" requests AND **responses**.

Introduction

- Like IDS/IPS/FW's exactly...
  - Three types of implementations
    - Host based (mod_security – can do more also)
    - Inline / network
    - Reverse proxies (most commercial products)

Introduction

- Just to straighten things up..
  - Application firewalls are no substitute for good programming practices.
  - Relying on an application firewall to protect bad software is doomed to the eventual catastrophic failure of the application
    - Blacklisting = known threats in know code
    - Whitelisting = ”unknown threats” in ”unknown code”
    - Whats inbetween?

Essentials

- Complete support for HTTP
  - Now that means everything and in every aspect

    headers, fields, 1.0 and 1.1, responses and requests

  - Anti anti ids & ips functionality
    - Normalisation & enforcing encoding schemes and such.

Essential Protection Mechanisms

- Two ”main” protection mechanisms
  - Blacklisting
    - Look for bad stuff
  - Whitelisting
    - Verify that input is correct
    - Learning how application works over time..

+ Defining what functionality you wish to be visible from your webserver (methods, headers etc..)

Protections other.

- High level of virtualisation
  - session data
  - cookies
  - application state
    - links
    - request flow
  - ”certain fields & data”
- Brute-force protection

Protections other.

- Different level of whitelisting (honestly don't know what to call this)
  - ”client aware” whitelisting (dynamic)
  - Links virtualization ^ signing
  - Request flow enforcement

- XML
  - schema & dtd validations ”made for you”

Protections other.

- And :/
  - Hardening your server configuration
    - Methods, headers (in and out)
    - protecting file uploads
  - Protecting your web server
    - validating http (whitelisting according to rfc =))

- DOS protection as well.

Things that cause problems..

- masked parameter names
- /dir/SESSIONID123123123/index.php
- Ajax, amfphp, applets – rpc etc..

Mod_security

- Audit logging
- Provides access to requests and responses
- Flexible regular expression-based rule engine.
- Rules can be combined
- External logic can be invoked
- well.. flexible =)

Mod_security

- ”waf” built on Apache

```
$sudo apt-get install libapache2-mod-security
$sudo  a2enmod mod-security
$sudo  /etc/init.d/apache2 force-reload
```

mod_security – sample rules

- Configuring (emacs /etc/apache2/conf.d/mod_security)

mod_security - basic

- SecRule REQUEST_URI|QUERY_STRING dirty
  - Rejects a request which contains the word ”dirty” in the querystrings or uri.
- SecRule ARGS:p dirty
  - parameter p cannot contain word dirty
  - Different types:
    - SecRule ARGS|!ARGS:z dirty (z can contain dirty)
    - SecRule ARGS:/^id_/ dirty (radio buttons and such -> which transform into arrays kinda)
- SecRule REQUEST_FILENAME "^/cgi-bin/login\.php$“ "chain,log,deny,status:403,phase:2“
- SecRule **ARGS_COMBINED_SIZE** "@gt 25"
  - Prevent buffer overflows?? :D
- SecRule REQUEST_FILENAME "/index.php" "chain,log,deny,status:403,phase:2"
- SecRule **ARGS_NAMES** "!^(p|a)$"
  - Whitelisting allowed parameters (p and a only allowed)
- SecServerSignature “MESHUGGAH WEB SERVER 1.0"
  - Web server type is now: norwegian black metal

mod_security - basic

- SecRule **HTTP_REFERER** "!^www.mysite.com$"
  - CSRF attacks prevented? (ye sure referers can be faked, but anyway)
- SecRule **RESPONSE_BODY** "ODBC Error Code"
  - Limiting what the web server talks back to the client.
- Session "evil scoring"
  - Blocking sessions based on score system.
- SecRule REQUEST_HEADERS:User-Agent "nikto" log,**deny**,msg:'Nikto Scanners Identified"
  - Filtering base on user agent.
- SecRule REQUEST_URI "^/cgi-bin/script\.pl" "log,**exec:/usr/local/apache/bin/test.sh**,phase:1"
  - Executes external scripts

mod_security - xss

- XSS (as presented in manual 1.9.x)
  - SecFilter "<script"
  - SecFilter "<.+>"
- OR

```
<Location /cms/article-update.php>
      SecFilterInheritance Off
      SecFilterSelective "ARGS|!ARG_body" "<.+>"
</Location>
```

mod_security

- mod_security and XML

SecRule REQUEST_HEADERS:Content-Type ^text/xml$
    phase:1,t:lowercase,nolog,pass,ctl:requestBodyProcessor=XML

SecRule REQBODY_PROCESSOR "!^XML$" nolog,pass,skip:1

SecRule XML "**@validateSchema /path/to/apache2/conf/xml.xsd**"

mod_security

- Whitelisting

Unfortunately, most of the mod_security samples and documentation doesn't realy guide you towards complete whitelisting.

ie. parameter "x" -> A-z0-9 etc. (SecFilterSelective ARG_recipient "![a-zA-Z0-9]+@webkreator\.com$">)

Sample core configuration contain stuff like →

mod_security

- Ok it requires some regexp skills to write your whitelist.. but what about the blacklist (core samples)

SecRule REQUEST_FILENAME|ARGS|ARGS_NAMES|REQUEST_HEADERS "(?:\b(?:on(?:(?:mo(?:use(?:o(?:ver|ut)|down|move|up)|ve)|key(?:press|down|up)|c(?:hange|lick)|s(?:elec|ubmi)t|(?:un)?load|dragdrop|resize|focus|blur)\b\W*?=|abort\b)|(?:l(?:owsrc\b\W*?\b(?:(?:java|vb)script|shell)|ivescript)|(?:href|url)\b\W*?\b(?:(?:java|vb)script|shell)|mocha):|type\b\W*?\b(?:text\b(?:\W*?\b(?:j(?:ava)?|ecma)script\b|[vbscript])|application\b\W*?\bx-(?:java|vb)script\b)|s(?:(?:tyle\b\W*=.*\bexpression\b\W*|ettimeout\b\W*?)\(|rc\b\W*?\b(?:(?:java|vb)script|shell|http):)|(?:c(?:opyparentfolder|reatetextrange)|get(?:special|parent)folder|background-image:|@import)\b|a(?:ctivexobject\b|lert\b\W*?\())|<(?:(?:body\b.*?\b(?:backgroun|onloa)d|input\b.*?\\btype\b\W*?\bimage)\b|!\[CDATA\[|script|meta)|.(?:(?:execscrip|addimpor)t|(?:fromcharcod|cooki)e|innerhtml)\b)" \ "log,id:950004,severity:2,msg:Attack name here'"

Who does what..

- Blacklisting
  - snort, ids/ips vendors
  - VA tools ”find” the same things
- Configuration hardening
  - Should be standard installation procedure..
- Whitelisting
  - WAF!!!!!!
  - and..
    - Load balancing
    - SSL termination and acceleration
    - Caching and transparent compression
    - ftp -> sftp
    - Web SSO
    - etc..etc.

What I'm interested in..

- That grey area in-between..
  - Detecting anomalities in user behavor!
  - Reporting!!!!!
- More virtualisation
  - links & sessions etc.
- Easy whitelisting
  - Implementation issues need to be solved
  - Configuration & management

Benefits

- Another layer of security (benefit?)
- Specialized security knowledge
  - Covering the unknown.
  - Developers don't always know what to protect against.
- Specialized application knowledge
  - In f.ex. "Xml" firewalls
- "Flexible" policy enforcement
  - Centralized policy on approved behavior (requires good co-operation between it-security and application developers)
- Intrusion attempt detection & logging in general
  - Most people have hardly logs on what happens in their web applications

Drawbacks

- Configuration
  - You often must teach the firewall to understand positive behavior.
- Single point of failure
- Performance
- Complexity
- Passing the buck
- Blacklisting
- Default deny
- Incompatibility

OSS vs. Commercial

OSS vs. Commercial (2)

- Open Source:
  - Do not have all the features of commercial offerings, but have the ones that are really important.
  - No nice GUIs yet - you have to get your hands dirty, understand how it works, and know the components well.

Web Intrusion Detection with ModSecurity 34 / 50

Evaluation Criteria

- Evaluating Web Application Firewalls
  - According to WAFEC
    - Deployment Architecture
    - HTTP and HTML Support
    - Detection Techniques
    - Prevention Techniques
    - Logging
    - Reporting
    - Management
    - Performance
    - XML

http://www.webappsec.org/projects/wafec