Your Stack Traces are Leaking CVEs

Fingerprinting Java Stack Traces

Luc Gommans

Intro

```
$ id
```

Luc Gommans

IT Security Consultant at X41 D-Sec

This talk will be about extracting information from stack traces

Credits to my colleague Eric Sesterhenn

Why would you extract information from stack traces?

Why would you extract information from stack traces?

When doing a security test on a web app written in Java, you can often trigger a stack trace

- For example by adding a null byte in an input field

Why would you extract information from stack traces?

When doing a security test on a web app written in Java, you can often trigger a stack trace

- For example by adding a null byte in an input field

If that immediately reveals a lot of their technology stack...

BeanStack

We developed a stack trace fingerprinting tool

https://BeanStack.io

Input: Java stack trace
Output: CVEs

BeanStack

We developed a stack trace fingerprinting tool

https://BeanStack.io

Input: Java stack trace
Output: CVEs

How does it work?

Stack Traces

```
java.lang.IllegalStateException: On-the-fly migration has not been activate
    at com.continental.coremedia.migration.OnTheFlyMigrationController.reso
    at com.coremedia.objectserver.web.AbstractViewController.handleRequestI
    at org.springframework.web.servlet.mvc.AbstractController.handleRequest
    at org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter.h
    at org.springframework.web.servlet.DispatcherServlet.doDispatch(Dispatc
    at com.coremedia.objectserver.web.DispatcherServlet.doDispatch(Dispatch
    at org.springframework.web.servlet.DispatcherServlet.doService(Dispatch
    at org.springframework.web.servlet.FrameworkServlet.processRequest(Fram
    at org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServ
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:617)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:717)
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(App
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(Application
    at org.tuckey.web.filters.urlrewrite.RuleChain.handleRewrite(RuleChain.
```

Stack Traces

```
java.lang.IllegalStateException: On-the-fly migration has not been activate
    at com.continental.coremedia.migration.OnTheFlyMigrationController.reso
    at com.coremedia.objectserver.web.AbstractViewController.handleRequestI
    at org.springframework.web.servlet.mvc.AbstractController.handleRequest
    at org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter.h
    at org.springframework.web.servlet.DispatcherServlet.doDispatch(Dispatc
    at com.coremedia.objectserver.web.DispatcherServlet.doDispatch(Dispatch
    at org.springframework.web.servlet.DispatcherServlet.doService(Dispatch
    at org.springframework.web.servlet.FrameworkServlet.processRequest(Fram
    at org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServ
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:617)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:717)
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(App
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(Application
    at org.tuckey.web.filters.urlrewrite.RuleChain.handleRewrite(RuleChain.
```

Stack Traces

```
public class Car {
    public int car;

    public static void main(String[] args) {
        // Build a new car...
        Car myNewCar = new Car();
    }
```

Stack Traces

```
public class Car {
    public int car;

    public static void main(String[] args) {
        // Build a new car...
        Car myNewCar = new Car();
    }

    public Car() {
        this.car = getWheels() + getFrame() + getEngine() + getElectricWindows();
    }
```

Stack Traces

```
public class Car {
    public int car;

    public static void main(String[] args) {
        // Build a new car...
        Car myNewCar = new Car();
    }

    public Car() {
        this.car = getWheels() + getFrame() + getEngine() + getElectricWindows();
    }

    private int getWheels() {
        throw new java.lang.RuntimeException("Oops!");
    }
```

Stack Traces

```
Exception in thread "main":
java.lang.RuntimeException
  at Car.getWheels(Car.java:14)
  at Car.<init>(Car.java:10)
  at Car.main(Car.java:6)
```

```
 1 public class Car {
 2     public int car;
 3
 4     public static void main(String[]
 5         // Build a new car...
 6         Car myNewCar = new Car();
 7     }
 8
 9     public Car() {
10         this.car = getWheels() + get
11     }
12
13     private int getWheels() {
14         throw new java.lang.RuntimeE
15     }
```

Stack Traces

```
Exception in thread "main":
java.lang.RuntimeException
  at Car.getWheels(Car.java:14)
  at Car.<init>(Car.java:10)
  at Car.main(Car.java:6)
```

```
 1 public class Car {
 2     public int car;
 3
 4     public static void main(String[]
 5         // Build a new car...
 6         Car myNewCar = new Car();
 7     }
 8
 9     public Car() {
10         this.car = getWheels() + get
11     }
12
13     private int getWheels() {
14         throw new java.lang.RuntimeE
15     }
```

Stack Traces

```
Exception in thread "main":
java.lang.RuntimeException
  at Car.getWheels(Car.java:14)
  at Car.<init>(Car.java:10)
  at Car.main(Car.java:6)
```

```
 1 public class Car {
 2     public int car;
 3
 4     public static void main(String[]
 5         // Build a new car...
 6         Car myNewCar = new Car();
 7     }
 8
 9     public Car() {
10         this.car = getWheels() + get
11     }
12
13     private int getWheels() {
14         throw new java.lang.RuntimeE
15     }
```

Stack Traces

```
Exception in thread "main":
java.lang.RuntimeException
  at Car.getWheels(Car.java:14)
  at Car.<init>(Car.java:10)
  at Car.main(Car.java:6)
```

```
 1 public class Car {
 2     public int car;
 3
 4     public static void main(String[]
 5         // Build a new car...
 6         Car myNewCar = new Car();
 7     }
 8
 9     public Car() {
10         this.car = getWheels() + get
11     }
12
13     private int getWheels() {
14         throw new java.lang.RuntimeE
15     }
```

Matching Code to a Stack Trace

```
class Car {
  6: Car.<init>
10: Car.getWheels
10: Car.getFrame
10: Car.getEngine
10: Car.getElectricWindows
14: java.lang.RuntimeException
}
```

```
 1 public class Car {
 2     public int car;
 3
 4     public static void main(String[]
 5         // Build a new car...
 6         Car myNewCar = new Car();
 7     }
 8
 9     public Car() {
10         this.car = getWheels() + get
11     }
12
13     private int getWheels() {
14         throw new java.lang.RuntimeE
15     }
```

Matching Code to a Stack Trace

```
class Car {
  6: Car.<init>
10: Car.getWheels
10: Car.getFrame
10: Car.getEngine
10: Car.getElectricWindows
14: java.lang.RuntimeException
}
```

```
Exception in thread "main":
java.lang.RuntimeException
  at Car.getWheels(Car.java:14)
  at Car.<init>(Car.java:10)
  at Car.main(Car.java:6)
```

Matching Code to a Stack Trace

```
class Car {
  6: Car.<init>
10: Car.getWheels
10: Car.getFrame
10: Car.getEngine
10: Car.getElectricWindows
14: java.lang.RuntimeException
}
```

What if you edit the source code?
Add a line, remove a line?

Matching Code to a Stack Trace

```
class Car {
  6: Car.<init>
11: Car.getWheels
11: Car.getFrame
11: Car.getEngine
11: Car.getElectricWindows
15: java.lang.RuntimeException
}
```

What if you edit the source code?
Add a line, remove a line?

All line numbers below will change

Matching Code to a Stack Trace

```
class Car {
  6: Car.<init>
11: Car.getWheels
11: Car.getFrame
11: Car.getEngine
11: Car.getElectricWindows
15: java.lang.RuntimeException
}
```

What if you edit the source code?
Add a line, remove a line?

All line numbers below will change

Different versions will have
different line numbers

Matching Code to a Stack Trace

```
javax.servlet.http.HttpServlet.service(HttpServlet.java:717)
org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:377)
org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:313)
org.apache.jasper.servlet.JspServlet.service(JspServlet.java:260)
```

Combining data from multiple lines

Matching Code to a Stack Trace

```
javax.servlet.http.HttpServlet.service(HttpServlet.java:717)
org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:377)
org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:313)
org.apache.jasper.servlet.JspServlet.service(JspServlet.java:260)
```

Combining data from multiple lines

Which version of Apache Jasper:

- calls JspServlet.serviceJspFile() on line 260, *and*
- calls JspServletWrapper.service() on line 313, *and*
- calls HttpServlet.service() on line 377?

Extracting Line Numbers from (Compiled) Code

How to extract function calls from the source code?

Extracting Line Numbers from (Compiled) Code

How to extract function calls from the source code?

- Let the Java compiler do the heavy lifting
- Output: class files

Extracting Line Numbers from (Compiled) Code

How to extract function calls from the source code?

- Let the Java compiler do the heavy lifting
- Output: class files

How to extract function calls from the .jar files?

- JAR files are ZIP files, so unzip
- Output: class files

Extracting Line Numbers from Class Files

Class file:

- *Metadata* Magic string, version number
- *Constant pool* Method names
- …
- *Methods* All methods in this class
- *Attributes* LineNumberTable

Extracting Line Numbers from Class Files

Class file:

- *Metadata* Magic string, version number
- *Constant pool* Method names
- …
- *Methods* All methods in this class
- *Attributes* LineNumberTable => [{ offset: line_number }, ...]

Extracting Line Numbers from Class Files

1. Foreach method in the class
2. Foreach instruction in its bytecode

Extracting Line Numbers from Class Files

1. Foreach method in the class
2. Foreach instruction in its bytecode
3. If the instruction is *invokevirtual*, *invokeinterface*, or *invokespecial*
4. Take the bytecode offset and do a *LineNumberTable* lookup

Extracting Line Numbers from Class Files

1. Foreach method in the class
2. Foreach instruction in its bytecode
3. If the instruction is *invokevirtual*, *invokeinterface*, or *invokespecial*
4. Take the bytecode offset and do a *LineNumberTable* lookup
5. Look up the method name in the *constant_pool*
6. Store in database: class, method called, caller, line number

Extracting Line Numbers from Class Files

1. Foreach method in the class
2. Foreach instruction in its bytecode
3. If the instruction is *invokevirtual*, *invokeinterface*, or *invokespecial*
4. Take the bytecode offset and do a *LineNumberTable* lookup
5. Look up the method name in the *constant_pool*
6. Store in database: class, method called, caller, line number

```
at javax.servlet.http.HttpServlet.service(HttpServlet.java:617)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:717)
```

Extracting Line Numbers from Class Files

1. Foreach method in the class
2. Foreach instruction in its bytecode
3. If the instruction is *invokevirtual*, *invokeinterface*, or *invokespecial*
4. Take the bytecode offset and do a *LineNumberTable* lookup
5. Look up the method name in the *constant_pool*
6. Store in database: class, method called, caller, line number

```
at javax.servlet.http.HttpServlet.service(HttpServlet.java:617)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:717)
```

Extracting Line Numbers from Class Files

1. Foreach method in the class
2. Foreach instruction in its bytecode
3. If the instruction is *invokevirtual*, *invokeinterface*, or *invokespecial*
4. Take the bytecode offset and do a *LineNumberTable* lookup
5. Look up the method name in the *constant_pool*
6. Store in database: class, method called, caller, line number

```
at javax.servlet.http.HttpServlet.service(HttpServlet.java:617)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:717)
```

Extracting Line Numbers from Class Files

1. Foreach method in the class
2. Foreach instruction in its bytecode
3. If the instruction is *invokevirtual*, *invokeinterface*, or *invokespecial*
4. Take the bytecode offset and do a *LineNumberTable* lookup
5. Look up the method name in the *constant_pool*
6. Store in database: class, method called, caller, line number

```
at javax.servlet.http.HttpServlet.service(HttpServlet.java:617)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:717)
```

Extracting Line Numbers from Class Files

1. Foreach method in the class
2. Foreach instruction in its bytecode
3. If the instruction is *invokevirtual*, *invokeinterface*, or *invokespecial*
4. Take the bytecode offset and do a *LineNumberTable* lookup
5. Look up the method name in the *constant_pool*
6. Store in database: class, method called, caller, line number

```
at javax.servlet.http.HttpServlet.service(HttpServlet.java:617)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:717)
```

Extracting Line Numbers from Class Files

Demo: importing our Car class

Extracting Line Numbers from Class Files

Problem: this gets large pretty fast

Extracting Line Numbers from Class Files

Problem: this gets large pretty fast

So we host a database for you

Currently 51 million rows for 1 625 unique versions of 32 products, 36GB on disk

Extracting Line Numbers from Class Files

Problem: this gets large pretty fast

So we host a database for you

Currently 51 million rows for 1 625 unique versions of 32 products, 36GB on disk

Problem: privacy! Customer code! NDAs!

Hashed submissions

What about customer code?

```
com.valve.halflife3.updater.getUpdate(Updater.java:9001)
```

Hashed submissions

What about customer code? Not a problem.

```
com.valve.halflife3.updater.getUpdate(Updater.java:9001)
```

`96e28e87f0bda1b32da0952d1c3dfb66:`	class+method
`e53992bb5f7af6f51773ad8ea3033d66:`	class
`e31ee78976445cfae5ee6447c9240fce:`	method
`9001`	line number

How to convert traces to hashed traces?

Burp Suite Plug-in

How to convert traces to hashed traces?

Burp plug-in

- Automatically look up stack traces
- Hash stack traces before submission
- Optionally blacklist any private classes, such as com.valve.halflife3

Version to CVEs

We’ve extracted versions, but how do we determine CVEs?

Version to CVEs

We’ve extracted versions, but how do we determine CVEs?

NIST matches CVEs to products,
provides a data feed

Version to CVEs

We've extracted versions, but how do we determine CVEs?

NIST matches CVEs to products,
provides a data feed

We wrote a microservice that pulls the feed

(a story for another day)

Version to CVEs

We add the *cpe_name* when importing a new product

BeanStack queries the CVE microservice internally, using the *cpe_name* and version from the fingerprinting result

Recap

Stack traces contain line numbers

We built a database of function calls and line numbers

Matching the two yields reliable version information

API results enhanced with a CVE database lookup

Future work

C#?

- Stack traces more frequently hidden from users
- Compiled code usually doesn’t show line numbers

Future work

C#?

- Stack traces more frequently hidden from users
- Compiled code usually doesn’t show line numbers

JavaScript (nodejs)?

- Less common than Java, but still a frequent encounter
- Throws stack traces with line numbers

Future work: JavaScript

```
ReferenceError: notdefd is not defined
at /root/keystonetest/routes/index.js:4:42
at Layer.handle [as handle_request] (/root/keystonetest/node_modules/express/lib/route
at next (/root/keystonetest/node_modules/express/lib/router/route.js:137:13)
at Route.dispatch (/root/keystonetest/node_modules/express/lib/router/route.js:112:3)
at Layer.handle [as handle_request] (/root/keystonetest/node_modules/express/lib/route
at /root/keystonetest/node_modules/express/lib/router/index.js:281:22
at Function.process_params (/root/keystonetest/node_modules/express/lib/router/index.j
at next (/root/keystonetest/node_modules/express/lib/router/index.js:275:10)
at /root/keystonetest/node_modules/grappling-hook/index.js:198:10
at _combinedTickCallback (internal/process/next_tick.js:131:7)
```

Future work: JavaScript

```
at /root/keystonetest/node_modules/grappling-hook/index.js:198:10
at next (/root/keystonetest/node_modules/express/lib/router/index.js:275:10)
at Function.process_params (/root/keystonetest/node_modules/express/lib/rout
at Layer.handle [as handle_request] (/root/keystonetest/node_modules/express
```

Elements of a JavaScript stack trace:

- Whitespace followed by “at” and a space
- Optionally: “myClass.function” or maybe just “function”
- Optionally: “[as func]”
- “path/to/file.js:19:4” (path:row:column)
  - Optionally between parentheses

Future work: JavaScript

```
at /root/keystonetest/node_modules/grappling-hook/index.js:198:10
at next (/root/keystonetest/node_modules/express/lib/router/index.js:275:10)
at Function.process_params (/root/keystonetest/node_modules/express/lib/rout
at Layer.handle [as handle_request] (/root/keystonetest/node_modules/express
```

Elements of a JavaScript stack trace:

- Whitespace followed by “at” and a space
- Optionally: “myClass.function” or maybe just “function”
- Optionally: “[as func]”
- **“path/to/file.js:19:4”** (path:row:column) ← The only certain part
  - Optionally between parentheses ← (well, mostly)

Future work: JavaScript

```
at /root/keystonetest/node_modules/grappling-hook/index.js:198:10
at next (/root/keystonetest/node_modules/express/lib/router/index.js:275:10)
at Function.process_params (/root/keystonetest/node_modules/express/lib/rout
at Layer.handle [as handle_request] (/root/keystonetest/node_modules/express
```

The calling function can have:

Zero names	*at /.../index.js:123:45*
One name	*at next (/.../index.js:123:45)*
One full name	*at Function.process_params (/.../index.js:123:45)*
Two names	*at Layer.handle [as handle_request] (/.../index.js:123:45)*

Future work: JavaScript

How to go from source code to function call mappings?

- JavaScript is not precompiled, we can’t use the same trick again

Future work: JavaScript

How to go from source code to function call mappings?

- JavaScript is not precompiled, we can’t use the same trick again

Or is it?

Future work: JavaScript

How to go from source code to function call mappings?

JavaScript is not precompiled, we can't use the same trick again

Or is it?

```
!function(e,t){"use strict";"object"==typeof module&&"object"=
?module.exports=e.document?t(e,!0):function(e){if(!e.document)
y requires a window with a document");return t(e)}:t(e)}("unde
window:this,function(C,e){"use strict";var t=[],E=C.document,r
,s=t.slice,g=t.concat,u=t.push,i=t.indexOf,n={},o=n.toString,v
toString,l=a.call(Object),y={},m=function(e){return"function"=
ypeof e.nodeType},x=function(e){return null!=e&&e===e.window},
e:!0,noModule:!0};function b(e,t,n){var r,i,o=(n=n||E).createE
text=e,t)for(r in c)(i=t[r]||t.getAttribute&&t.getAttribute(r)
```

Future work: JavaScript

How to go from source code to function call mappings?

- JavaScript is not precompiled, we can't use the same trick again

Or is it?

```
!function(e,t){"use strict";"object"==typeof module&&"object"=
?module.exports=e.document?t(e,!0):function(e){if(!e.document)
y requires a window with a document");return t(e)}:t(e)}("unde
window:this,function(C,e){"use strict";var t=[],E=C.document,r
,s=t.slice,g=t.concat,u=t.push,i=t.indexOf,n={},o=n.toString,v
```

Developers wanted to map minified code back to the source.

Future work: JavaScript

Source maps map minified source code to the original.

```
Mapping {
  generatedLine: 1,
  generatedColumn: 0,
  lastGeneratedColumn: null,
  source: '0',
  originalLine: 1,
  originalColumn: 0,
  name: 'exports' }
```

Future work: JavaScript

Source maps map minified source code to the original.

So we can take

- Mapping.name
- Mapping.originalLine
- Mapping.originalColumn

```
Mapping {
  generatedLine: 1,
  generatedColumn: 0,
  lastGeneratedColumn: null,
  source: '0',
  originalLine: 1,
  originalColumn: 0,
  name: 'exports' }
```

Future work: JavaScript

Source maps map minified source code to the original.

So we can take

- Mapping.name
- Mapping.originalLine
- Mapping.originalColumn

And make the same call map as with Java

```
express/lib/router/route.js 211:6 this
express/lib/router/route.js 211:11 stack
express/lib/router/route.js 211:17 push
express/lib/router/route.js 211:22 layer
express/lib/router/route.js 214:4 null
express/lib/router/route.js 214:11 this
express/lib/router/layer.js 9:0 null
express/lib/router/layer.js 16:0 null
express/lib/router/layer.js 16:4 pathRegexp
express/lib/router/layer.js 16:17 require
```

Future work: JavaScript

Source maps map minified source code to the original.

So we can take

- Mapping.name
- Mapping.originalLine
- Mapping.originalColumn

And make the same call map as with Java

```
express/lib/router/route.js 211:6 this
express/lib/router/route.js 211:11 stack
express/lib/router/route.js 211:17 push
```

Even stack trace lines with zero names, we can see if the file:line:column exists in the source map

Future work: JavaScript

```
# js lookup-trace.js traceback.txt
We do not have this source file: internal/process/next_tick.js
We do not have this source file: internal/process/next_tick.js
Exact match for notdefd at keystonetest/routes/index.js:4:41
Fuzzy match at modules/express/lib/router/layer.js:95:4
   but the name is 'fn' instead of []
Exact match for handle_request at modules/express/lib/router/route.js:137:12
Exact match for next at modules/express/lib/router/route.js:112:2
Fuzzy match at modules/express/lib/router/layer.js:95:4
   but the name is 'fn' instead of [ 'Route.dispatch' ]
Exact match for handle_request at modules/express/lib/router/index.js:281:21
Fuzzy match at modules/express/lib/router/index.js:335:11
   but the name is 'done' instead of []
Exact match for process_params at modules/express/lib/router/index.js:275:9
Fuzzy match at modules/grappling-hook/index.js:198:9
   but the name is 'apply' instead of [ 'next' ]
```

Future work: JavaScript

To do:

- Putting the source map in a database
- Experiment with accuracy, especially of fuzzy matches
- Import a lot of modules
- Write another API endpoint
- Update the Burp client

Future work: JavaScript

To do:

- Putting the source map in a database
- Experiment with accuracy, especially of fuzzy matches
- Import a lot of modules
- Write another API endpoint
- Update the Burp client
- … basically, go from PoC to PROD :-)

Summary

BeanStack can match stack traces against any .jar file

JavaScript is a promising next target

Let us know what you think!

- Is JavaScript useful for you?
- Do you need us to import another Java library?

Thank you

Try it at https://beanstack.io
Slides at https://github.com/x41sec/slides

Built by Eric Sesterhenn and Luc Gommans

Contact: `luc.gommans@x41-dsec.de`