Semmle™

No More Whack-a-Mole:

How to Find and Prevent Entire Classes of Security Vulnerabilities

Sam Lanning

@samlanning @samlanning

A story of many bugs (CVE-2017-8046)

7 September 2017
Mo privately discloses vulnerability and exploit in Spring Framework

21 September 2017
Pivotal publish a patch, and make an announcement.

22 September 2017
Mo checks patch, sees it's incomplete sends updated exploit to Pivotal

26 September 2017
Pivotal sends Mo details of second attempt at fix

27 September 2017
Mo checks patch, sees it's ***still*** incomplete sends updated exploit to Pivotal

25 October 2017
Pivotal publishes a complete refactor of relevant code to hopefully prevent further occurrences.

https://lgtm.com/blog/spring_data_rest_CVE-2017-8046_gl

A story of many bugs 2

27 April 2016
S2-032 / CVE-2016-3081
RCE in Apache Struts 2 via OGNL
Nike Zheng

12 May 2016
S2-033 / CVE-2016-3087
RCE in Apache Struts 2 via OGNL
Alvaro Munoz

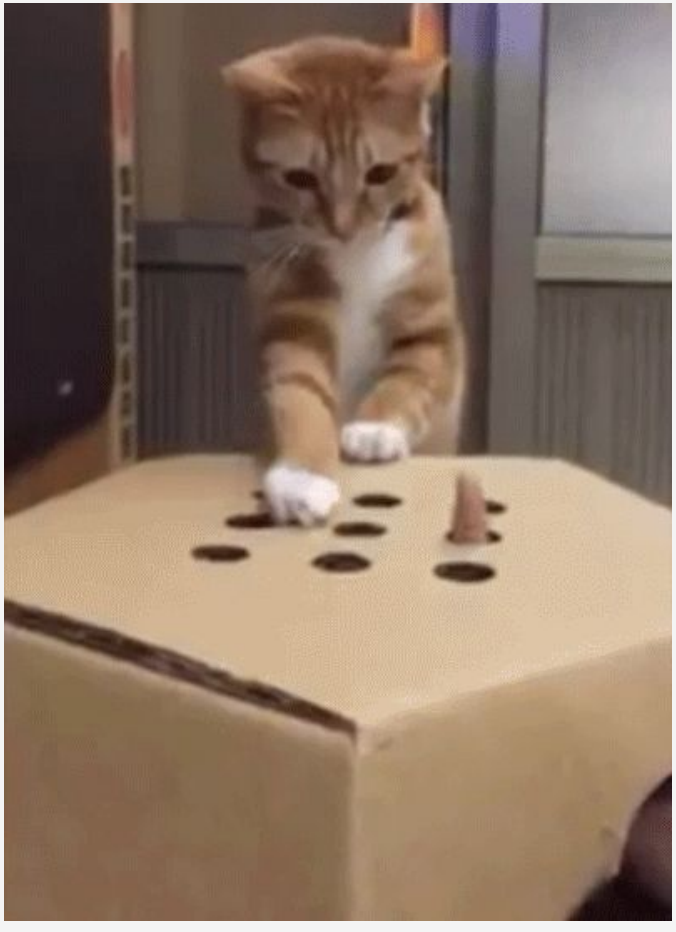
20 June 2016
S2-037 / CVE-2016-4438
RCE in Apache Struts 2 via OGNL
Chao Jack, Shinsaku Nomura

19 March 2017
S2-045 / CVE-2017-5638
RCE in Apache Struts 2 via OGNL
Nike Zheng

22 September 2017
S2-046 / CVE-2017-5638
RCE in Apache Struts 2 via OGNL
Chris Frohoff, Nike Zheng, Alvaro Munoz

24 September 2018
S2-057 / CVE-2018-11776
RCE in Apache Struts 2 via OGNL
Man Yue Mo

See Also: CVE-2012-0394, CVE-2013-1966, CVE-2012-0391, CVE-2013-2115, CVE-2012-0393



https://www.reddit.com/r/gifs/comments/2nyeb1/arcade_game_for_cats/

Solution:

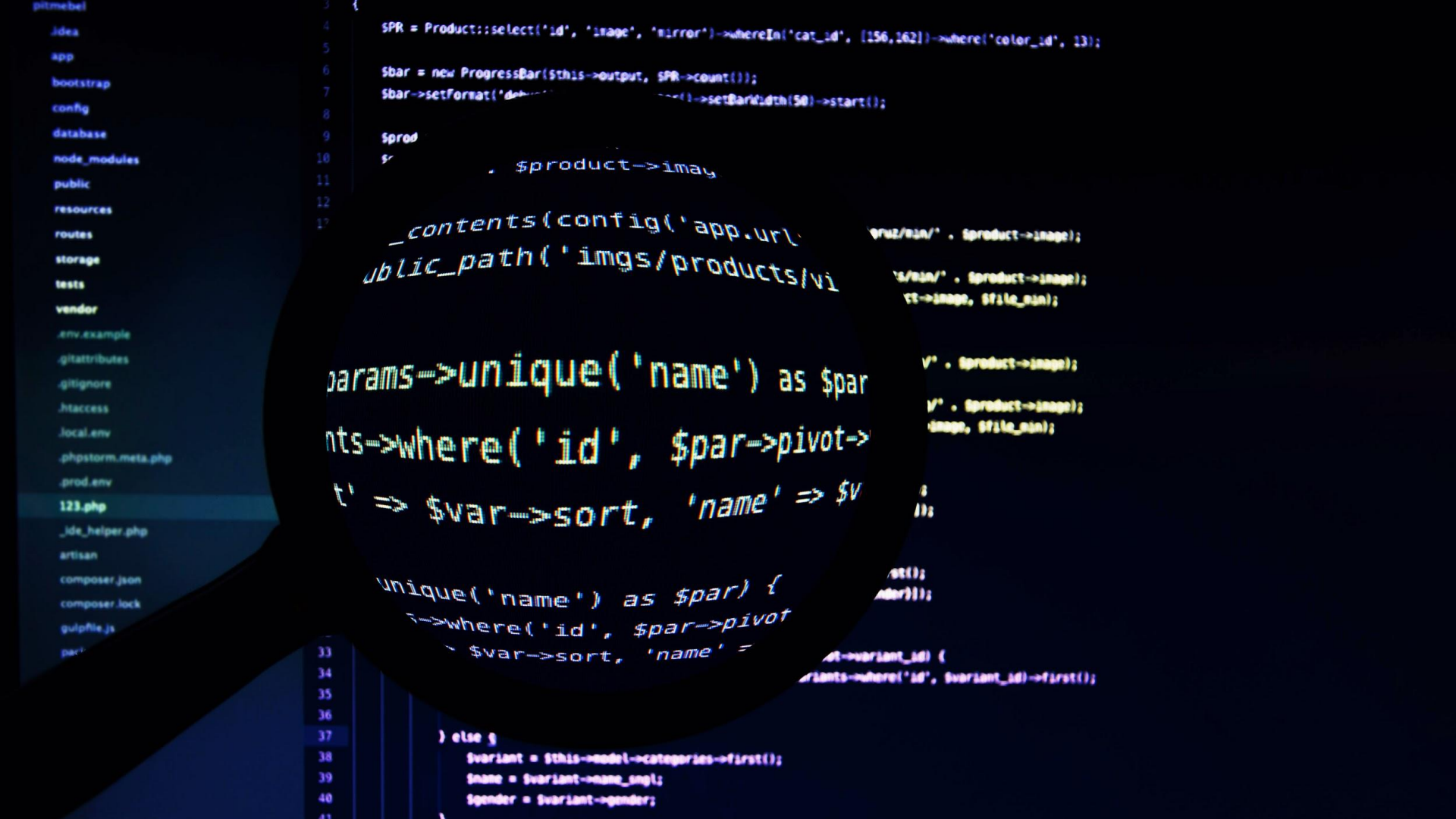
When a new mistake is discovered, try and find similar mistakes across your code base

Variant Analysis?

"After doing this [root cause analysis], our next step is **variant analysis**: finding and investigating any variants of the vulnerability. It's important that we find all such variants and patch them simultaneously, otherwise we bear the risk of these being exploited in the wild."

- Steven Hunter, MSRC Vulnerabilities & Mitigations team

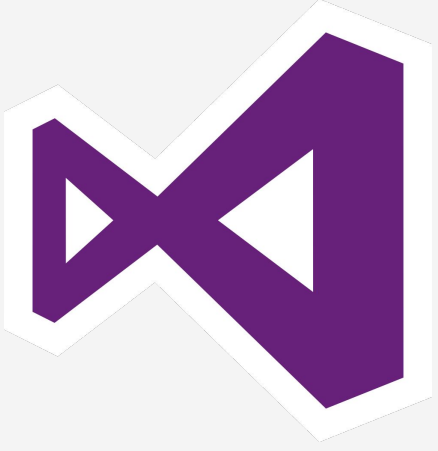
https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/


pitmebel
.idea
app
bootstrap
config
database
node_modules
public
resources
routes
storage
tests
vendor
.env.example
.gitattributes
.gitignore
.htaccess
.local.env
.phpstorm.meta.php
.prod.env
123.php
_ide_helper.php
artisan
composer.json
composer.lock
gulpfile.js
$PR = Product::select('id', 'image', 'mirror')->whereIn('cat_id', [156,162])->where('color_id', 13);
$bar = new ProgressBar($this->output, $PR->count());
$product->imag
_contents(config('app.url
ublic_path('imgs/products/vi
params->unique('name') as $par
nts->where('id', $par->pivot->
t' => $var->sort, 'name' => $v
unique('name') as $par) {
->where('id', $par->pivot
$var->sort, 'name'
} else {
$variant = $this->model->categories->first();
$name = $variant->name_sngl;
$gender = $variant->gender;

Code Navigation / IDE

sourcegraph

eclipse



Automating Variant Analysis

Could we describe a mistake in a way that lets us automatically find other instances?

An Example: Chakra

```
1 HRESULT SomeClass::vulnerableFunction(Var* args, UINT argCount, Var* retVal)
2 {
3     // get first argument -
4     // from Chakra, acquire pointer to array
5     BYTE* pBuffer;
6     UINT bufferSize;
7     hr = Jscript::GetTypedArrayBuffer(args[1], &pBuffer, &bufferSize);
8
9     // get second argument –
10    // from Chakra, obtain an integer value
11    int someValue;
12    hr = Jscript::VarToInt(args[2], &someValue);
13
14    // perform some operation on the array acquired previously
15    doSomething(pBuffer, bufferSize);
16 }
```

https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/

An Example: Chakra

```
1 HRESULT SomeClass::vulnerableFunction(Var* args, UINT argCount, Var* retVal)
2 {
3     // get first argument -
4     // from Chakra, acquire pointer to array
5     BYTE* pBuffer;
6     UINT bufferSize;
7     hr = Jscript::GetTypedArrayBuffer(args[1], &pBuffer, &bufferSize);
8
9     // get second argument -
10    // from Chakra, obtain an integer value
11    int someValue;
12    hr = Jscript::VarToInt(args[2], &someValue);
13
14    // perform some operation on the array acquired previously
15    doSomething(pBuffer, bufferSize);
16 }
```

https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/

An Example: Chakra

```
1 HRESULT SomeClass::vulnerableFunction(Var* args, UINT argCount, Var* retVal)
2 {
3     // get first argument -
4     // from Chakra, acquire pointer to array
5     BYTE* pBuffer;
6     UINT bufferSize;
7     hr = Jscript::GetTypedArrayBuffer(args[1], &pBuffer, &bufferSize);
8
9     // get second argument -
10    // from Chakra, obtain an integer value
11    int someValue;
12    hr = Jscript::VarToInt(args[2], &someValue);
13
14    // perform some operation on the array acquired previously
15    doSomething(pBuffer, bufferSize);
16 }
```

https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/

An Example: Chakra

```
1 HRESULT SomeClass::vulnerableFunction(Var* args, UINT argCount, Var* retVal)
2 {
3     // get first argument -
4     // from Chakra, acquire pointer to array
5     BYTE* pBuffer;
6     UINT bufferSize;
7     hr = Jscript::GetTypedArrayBuffer(args[1], &pBuffer, &bufferSize);
8
9     // get second argument –
10    // from Chakra, obtain an integer value
11    int someValue;
12    hr = Jscript::VarToInt(args[2], &someValue);
13
14    // perform some operation on the array acquired previously
15    doSomething(pBuffer, bufferSize);
16 }
```

https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/

An Example: Chakra

```
1 HRESULT SomeClass::vulnerableFunction(Var* args, UINT argCount, Var* retVal)
2 {
3     // get first argument -
4     // from Chakra, acquire pointer to array
5     BYTE* pBuffer;
6     UINT bufferSize;
7     hr = Jscript::GetTypedArrayBuffer(args[1], &pBuffer, &bufferSize);
8
9     // get second argument -
10    // from Chakra, obtain an integer value
11    int someValue;
12    hr = Jscript::VarToInt(args[2], &someValue);
13
14    // perform some operation on the array acquired previously
15    doSomething(pBuffer, bufferSize);
16 }
```

https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/

An Example: Chakra

```
var buf = new ArrayBuffer(length);
var arr = new Uint8Array(buf);

var param = {}
param.valueOf = function() {
    /* free `buf`
       (code to actually do this would be defined elsewhere) */
    neuter(buf);  // neuter `buf` by e.g. posting it to a web worker
    gc();         // trigger garbage collection
    return 0;
};

vulnerableFunction(arr, param);
```

https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/

An Example: Chakra

```
1 from Variable v, TypedArrayBufferPointer def, FunctionCall call, VariableAccess use
2 where
3   // variable is defined at "def", and used at "use"
4   v.getAnAccess() = def and v.getAnAccess() = use
5   // "call" may eventually call a function that calls back into JS code
6   and exists(Function g | call.getTarget().calls+(g) and g.hasName("MethodCallToPrimitive"))
7   // "call" happens after "def"
8   and def.getASuccessor+() = call
9   // "use" happens after "call"
10  and call.getASuccessor+() = use
11 select def, call, use
```

*slightly modified query from:

https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/

An Example: Chakra

```
1 from Variable v, TypedArrayBufferPointer def, FunctionCall call, VariableAccess use
2 where
3   // variable is defined at "def", and used at "use"
4   v.getAnAccess() = def and v.getAnAccess() = use
5   // "call" may eventually call a function that calls back into JS code
6   and exists(Function g | call.getTarget().calls+(g) and g.hasName("MethodCallToPrimitive"))
7   // "call" happens after "def"
8   and def.getASuccessor+() = call
9   // "use" happens after "call"
10  and call.getASuccessor+() = use
11 select def, call, use
```

*slightly modified query from:

https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/

An Example: Chakra

```
from Variable v, TypedArrayBufferPointer def, FunctionCall call, VariableAccess use
where
  // variable is defined at "def", and used at "use"
  v.getAnAccess() = def and v.getAnAccess() = use
  // "call" may eventually call a function that calls back into JS code
  and exists(Function g | call.getTarget().calls+(g) and g.hasName("MethodCallToPrimitive"))
  // "call" happens after "def"
  and def.getASuccessor+() = call
  // "use" happens after "call"
  and call.getASuccessor+() = use
select def, call, use
```

*slightly modified query from:

https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/

An Example: Chakra

```
from Variable v, TypedArrayBufferPointer def, FunctionCall call, VariableAccess use
where
  // variable is defined at "def", and used at "use"
  v.getAnAccess() = def and v.getAnAccess() = use
  // "call" may eventually call a function that calls back into JS code
  and exists(Function g | call.getTarget().calls+(g) and g.hasName("MethodCallToPrimitive"))
  // "call" happens after "def"
  and def.getASuccessor+() = call
  // "use" happens after "call"
  and call.getASuccessor+() = use
select def, call, use
```

*slightly modified query from:

https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/

An Example: Chakra

```
from Variable v, TypedArrayBufferPointer def, FunctionCall call, VariableAccess use
where
  // variable is defined at "def", and used at "use"
  v.getAnAccess() = def and v.getAnAccess() = use
  // "call" may eventually call a function that calls back into JS code
  and exists(Function g | call.getTarget().calls+(g) and g.hasName("MethodCallToPrimitive"))
  // "call" happens after "def"
  and def.getASuccessor+() = call
  // "use" happens after "call"
  and call.getASuccessor+() = use
select def, call, use
```

*slightly modified query from:

https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/

An Example: Chakra

```
1 HRESULT SomeClass::vulnerableFunction(Var* args, UINT argCount, Var* retVal)
2 {
3     // get first argument -
4     // from Chakra, acquire pointer to array
5     BYTE* pBuffer;
6     UINT bufferSize;
7     hr = Jscript::GetTypedArrayBuffer(args[1], &pBuffer, &bufferSize);
8
9     // get second argument -
10    // from Chakra, obtain an integer value
11    int someValue;
12    hr = Jscript::VarToInt(args[2], &someValue);
13
14    // perform some operation on the array acquired previously
15    doSomething(pBuffer, bufferSize);
16 }
```

def

use

https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/

An Example: Chakra

```
from Variable v, TypedArrayBufferPointer def, FunctionCall call, VariableAccess use
where
  // variable is defined at "def", and used at "use"
  v.getAnAccess() = def and v.getAnAccess() = use
  // "call" may eventually call a function that calls back into JS code
  and exists(Function g | call.getTarget().calls+(g) and g.hasName("MethodCallToPrimitive"))
  // "call" happens after "def"
  and def.getASuccessor+() = call
  // "use" happens after "call"
  and call.getASuccessor+() = use
select def, call, use
```

*slightly modified query from:

https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/

An Example: Chakra

```
1 HRESULT SomeClass::vulnerableFunction(Var* args, UINT argCount, Var* retVal)
2 {
3     // get first argument -
4     // from Chakra, acquire pointer to array
5     BYTE* pBuffer;
6     UINT bufferSize;
7     hr = Jscript::GetTypedArrayBuffer(args[1], &pBuffer, &bufferSize);
8
9     // get second argument –
10    // from Chakra, obtain an integer value
11    int someValue;
12    hr = Jscript::VarToInt(args[2], &someValue);
13                                                    call
14    // perform some operation on the array acquired previously
15    doSomething(pBuffer, bufferSize);
16 }
```

https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/

An Example: Chakra

```
from Variable v, TypedArrayBufferPointer def, FunctionCall call, VariableAccess use
where
  // variable is defined at "def", and used at "use"
  v.getAnAccess() = def and v.getAnAccess() = use
  // "call" may eventually call a function that calls back into JS code
  and exists(Function g | call.getTarget().calls+(g) and g.hasName("MethodCallToPrimitive"))
  // "call" happens after "def"
  and def.getASuccessor+() = call
  // "use" happens after "call"
  and call.getASuccessor+() = use
select def, call, use
```

*slightly modified query from:

https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/

An Example: Chakra

```
1 HRESULT SomeClass::vulnerableFunction(Var* args, UINT argCount, Var* retVal)
2 {
3     // get first argument -
4     // from Chakra, acquire pointer to array
5     BYTE* pBuffer;
6     UINT bufferSize;                                    1st
7     hr = Jscript::GetTypedArrayBuffer(args[1], &pBuffer, &bufferSize);
8                                                          def
9     // get second argument –
10    // from Chakra, obtain an integer value
11    int someValue;  2nd
12    hr = Jscript::VarToInt(args[2], &someValue);
13                                                        call
14    // perform some operation on the array acquired previously
15    doSomething(pBuffer, bufferSize);
16 }
```

https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/

An Example: Chakra

```
from Variable v, TypedArrayBufferPointer def, FunctionCall call, VariableAccess use
where
  // variable is defined at "def", and used at "use"
  v.getAnAccess() = def and v.getAnAccess() = use
  // "call" may eventually call a function that calls back into JS code
  and exists(Function g | call.getTarget().calls+(g) and g.hasName("MethodCallToPrimitive"))
  // "call" happens after "def"
  and def.getASuccessor+() = call
  // "use" happens after "call"
  and call.getASuccessor+() = use
select def, call, use
```

*slightly modified query from:

https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/

An Example: Chakra

```
1  HRESULT SomeClass::vulnerableFunction(Var* args, UINT argCount, Var* retVal)
2  {
3      // get first argument -
4      // from Chakra, acquire pointer to array
5      BYTE* pBuffer;
6      UINT bufferSize;
7      hr = Jscript::GetTypedArrayBuffer(args[1], &pBuffer, &bufferSize);
8
9      // get second argument –
10     // from Chakra, obtain an integer value
11     int someValue;
12     hr = Jscript::VarToInt(args[2], &someValue);
13
14     // perform some operation on the array acquired previously
15     doSomething(pBuffer, bufferSize);
16 }
```

1st

call

2nd use

https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/

An Example: Chakra

```
1 from Variable v, TypedArrayBufferPointer def, FunctionCall call, VariableAccess use
2 where
3   // variable is defined at "def", and used at "use"
4   v.getAnAccess() = def and v.getAnAccess() = use
5   // "call" may eventually call a function that calls back into JS code
6   and exists(Function g | call.getTarget().calls+(g) and g.hasName("MethodCallToPrimitive"))
7   // "call" happens after "def"
8   and def.getASuccessor+() = call
9   // "use" happens after "call"
10  and call.getASuccessor+() = use
11 select def, call, use
```

*slightly modified query from:

https://blogs.technet.microsoft.com/srd/2018/08/16/vulnerability-hunting-with-semmle-ql-part-1/

Beyond your own code

- ➔ Make your (general-purpose) mistake descriptions open source!
- ➔ Use external mistake descriptions!

ZipSlip

https://snyk.io/research/zip-slip-vulnerability

ZipSlip

```
../../../.bashrc

../../../../../../../../../etc/crontab
```

https://snyk.io/research/zip-slip-vulnerability

ZipSlip

```
1 Enumeration<ZipEntry> entries = zip.getEntries();
2 while (entries.hasMoreElements()) {
3    ZipEntry e = entries.nextElement();
4    File f = new File(destinationDir, e.getName());
5    InputStream input = zip.getInputStream(e);
6    IOUtils.copy(input, write(f));
7 }
```

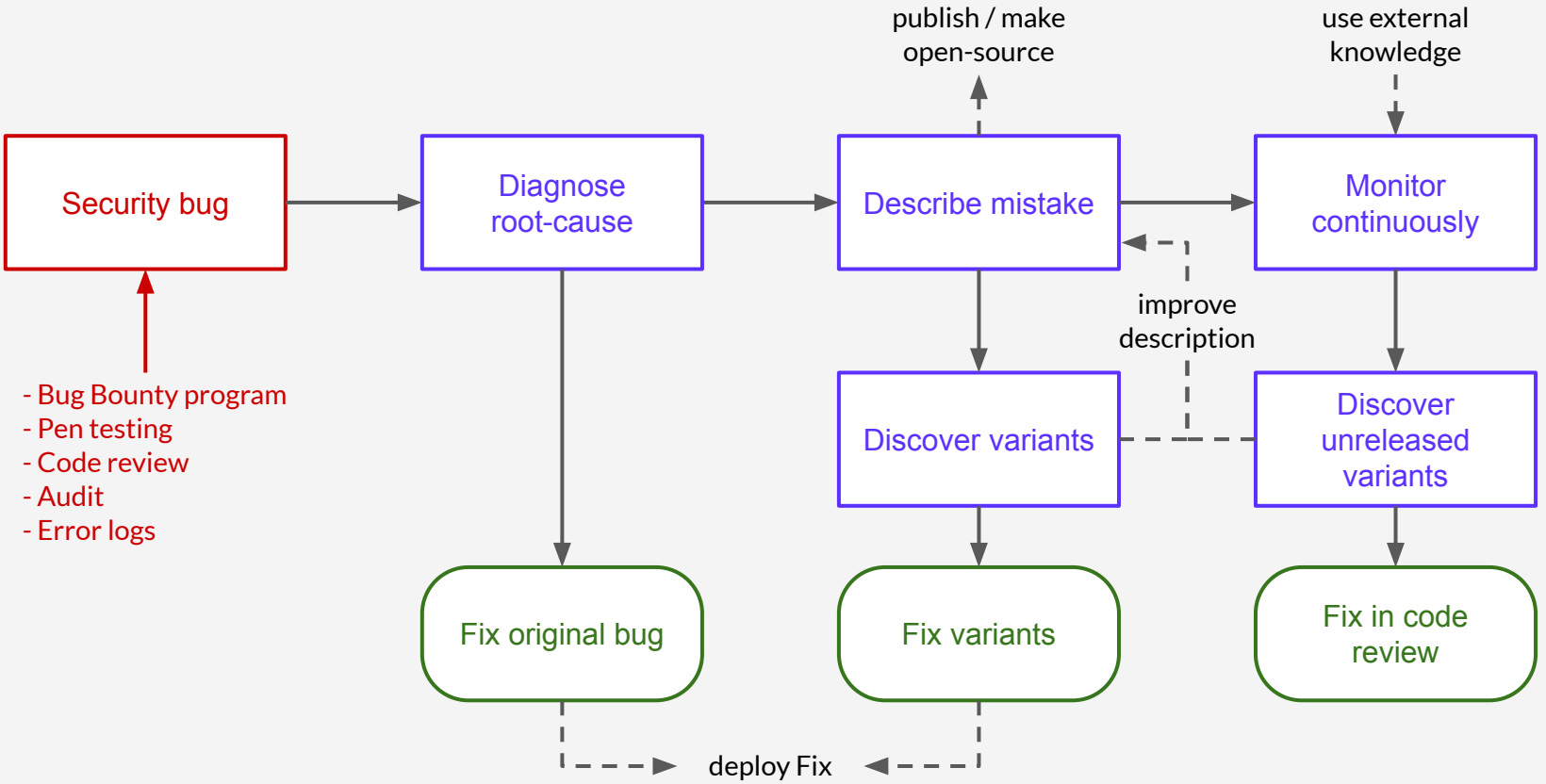
https://snyk.io/research/zip-slip-vulnerability

ZipSlip

```
1 Enumeration<ZipEntry> entries = zip.getEntries();
2 while (entries.hasMoreElements()) {
3    ZipEntry e = entries.nextElement();
4    File f = new File(destinationDir, e.getName());
5    InputStream input = zip.getInputStream(e);
6    IOUtils.copy(input, write(f));
7 }
```

https://snyk.io/research/zip-slip-vulnerability

Fitting in to your workflow



No vulnerability response process?

What variant analysis is NOT

- ➔ A replacement for good security architecture, a way to avoid large refactors
- ➔ A replacement for exploit mitigation
- ➔ A replacement for other security processes
- ➔ Something that automatically fixes bugs / vulnerabilities.

Recap

- ➔ You should do variant analysis
- ➔ Better yet, you should do ***automated*** variant analysis
- ➔ Use and contribute to the shared knowledge / checks
- ➔ Checks should be run continuously, not once-off!
- ➔ VA compliments (not replaces) other security practices

Thank You

Semmle™

http://**semmle.com**

@Semmle

Sam Lanning

@samlanning

@samlanning