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 your CVEs
  - More powerful pattern finder
  - Regression Tests
- ClientDependency Massacre
- Conclusion
Brief introduction to CodeQL

CodeQL’s variant analysis and powerful analyzers
How Semmle QL works

**Analysis Overview**

The database schema is (source) language specific, as are queries and libraries. Multi-language code bases are analyzed one language at a time.
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

• from VariableAccess va

  where va.getTarget().getName().regexpMatch(".*pass(wd|word|code).*")

  select va.getTarget()
import javascript

class SuspiciousExpr extendsInvokeExpr {

    SuspiciousExpr() {
        exists(StringLiteral s |
            getCalleeName() = "RegExp" and
            s.getStringValue().matches("%.*") and
            getEnclosingStmt() = s.getEnclosingStmt()
        )
        or
        exists(RegExpLiteral regex |
            regex.getValue().matches("%.*") and
            getEnclosingStmt() = regex.getEnclosingStmt()
        )
    }

    from SuspiciousExpr s

    select s
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
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,

  ```javascript
  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 possibly 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 uses \. instead of \\ in these two regular expression
Keybase hostname-validation regular expression
Let’s model this flaw

**Step 1: Find all occurrence**

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

  select c

**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
Electron 1.2.2 - 4.2.12

Regular expression failure upon checking a website’s URL to activate the webExtension
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=L3VtYnJhY28vbGliL2pzdWVyeS9qcXVlcnkubWluLmpz&t=Css&cdv=1`
  - `/umbraco/lib/jquery/jquery.min.js`
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, March 17th: [https://umbraco.com/blog/security-advisory-17th-of-march-patch-for-your-site-is-now-available/](https://umbraco.com/blog/security-advisory-17th-of-march-patch-for-your-site-is-now-available/)

**Umbraco Forms**

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

- 2016, January 27th: [https://umbraco.com/blog/umbraco-forms-security-notice/](https://umbraco.com/blog/umbraco-forms-security-notice/)
Umbraco CMS Local File Inclusion

GET /DependencyHandler.axd?t=Css&cdv=1 HTTP/1.1 200 OK
Cache-Control: public, max-age=86400, s-maxage=86400
Content-Type: text/xml
Expires: Wed, 10 May 2017 23:17:01 GMT
Last-Modified: Sun, 30 Apr 2017 23:17:01 GMT
ETag: "9af8f5be3895a83f13553ef7f8e80"
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[urlrewritesSection,UrlRewritingNet.UrlRewriter" />
<section name="microsoft.scripting"
on,Microsoft.Scripting,Version=1.0.0.0,Culture
eanutral,PublicKeyToken=31bf3656ad334e35" requirePermission="false" />
</section

GET /DependencyHandler.axd?s=aHR8cDovL3VyYXhhbXBsZS5jb20vd2V1LmNvbmxZpZw== 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

呜哇他bling bling的！
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
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

```java
class WebIHttpHandler extends Method {
    WebIHttpHandler() {
        getDeclaringType().getA.BaseType().hasQualifiedName("System.
        Web.IHttpHandler") and
        isSourceDeclaration()
    }
}
```
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
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
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
- CVE-2018-1000861
- RCE exists in the Stapler web framework used by Jenkins
- Stapler staplers most objects to URLs

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

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

- Use UML to find a good gadget to jump into the RCE chain
• CVE-2018-1000861

• RCE exists in the Stapler web framework used by Jenkins

UML Class Diagram
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
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
Client-side URL redirect

Client-side URL redirection based on unvalidated user input may cause redirection to malicious web sites.
Untrusted XML is read insecurely

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

```java
class BeanValidationConfig extends TaintTracking::Configuration {
    BeanValidationConfig() { this = "BeanValidationConfig" }

    override predicate isSource(Node source) { source instanceof InsecureBeanValidationSource }

    override predicate isSink(Node sink) { sink instanceof BuildConstraintViolationWithTemplateSink }
}
from BeanValidationConfig val, PathNode source, PathNode sink
where val.hasFlowPath(source, sink)
select source, source, sink, "instances new objects"
```
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
<table>
<thead>
<tr>
<th>FORUMS</th>
<th>LATEST</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Moderated Forum</td>
<td>MAR 20, 2020 09:22 PM</td>
<td>Use this moderated forum to test the forum system - posts will be occasionally deleted. Testing moderated forum by JunieB</td>
</tr>
</tbody>
</table>

**.NET LANGUAGES**

<table>
<thead>
<tr>
<th>FORUMS</th>
<th>LATEST</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>IronPython, IronRuby, and Other Languages</td>
<td>APR 02, 2020 04:43 AM</td>
<td>Questions about IronPython, IronRuby, and other languages for ASP.NET Re: # vs C# in Performance by Rion Williams</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FORUMS</th>
<th>LATEST</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>C#</td>
<td>AUG 25, 2020 07:18 AM</td>
<td>Questions about using C# for ASP.NET development Re: how can i display ip site is by XAPK Installer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FORUMS</th>
<th>LATEST</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual Basic .NET</td>
<td>JUL 20, 2020 11:09 AM</td>
<td>Questions about using Visual Basic .NET for ASP.NET development Re: Create a text delimited string of content a... by Sean Fang</td>
</tr>
</tbody>
</table>

**ABOUT THIS SITE**

<table>
<thead>
<tr>
<th>FORUMS</th>
<th>LATEST</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>What's New</td>
<td>FEB 07, 2019 09:31 PM</td>
<td>Here we let you know what is going on with the <a href="http://www.asp.net">www.asp.net</a> website, including site updates, maintenance windows, etc. ASP.NET Website Updates - February 6, 2019 by tmorton</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FORUMS</th>
<th>LATEST</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feedback on this website</td>
<td>AUG 24, 2020 05:36 PM</td>
<td>Have a problem with the website, or a new feature suggestion? Let us know! Re: read and unread threads are now styled th... by tmorton</td>
</tr>
</tbody>
</table>

**RETired FORUMS**
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 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 own to support web farms.

Knowledge of a the validation key allows an authenticated user with a mailbox to parse arbitrary objects to be deserialized into the web application, which runs as SYSTEM.

The security update addresses the vulnerability by correcting how Microsoft Exchange creates the keys during install.
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
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
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