Burp Suite Extension Writing Workshop

DevSecCon Singapore

March 01, 2019

Sanoop Thomas & Samandeep Singh

Thank you for showing up.

Thank You

About Us

- Sanoop Thomas
  - Organizer – BSides Singapore
    - https://bsidessg.org/
  - 9+ years in Information Security
  - Created Halcyon IDE – an IDE for Nmap Script Developers
    - https://halcyon-ide.org/
  - Hosting SecTools Podcast at Infosec Campus
    - https://infoseccampus.com/
  - Barely get time to write blogpost, but presentations/talks are updated
    - https://devilslab.in/
  - Presented at OWASP India, Nullcon Goa, HITBGSEC, ROOTCON, BlackHat Arsenal (USA, Asia) and Defcon
  - Tweet me at - **@s4n7h0**

About Us

- Samandeep Singh
  - Principal Security Consultant at SEC Consult Singapore
  - OSCE, OSCP, CRT
  - Co-Organizer – BSides Singapore
  - 6+ years in Information Security
  - Author of XVWA – a WebSec learning app
  - Interested in SDR, Windows Exploitation, Mainframe Hacking
  - Licensed scuba and sky diver
  - Adventure sports and traveller in free time
  - Tweet me at - **@samanL33T**

Agenda

- Introduction to Burp Suite
- Burp features:
  - scoping, proxy settings, repeater mode and other options
- Extender Capabilities
- Setting up development environment
- Important/useful methods
- Debugging tips
- Packaging tips
- Extensions will be Developed in Java and Python

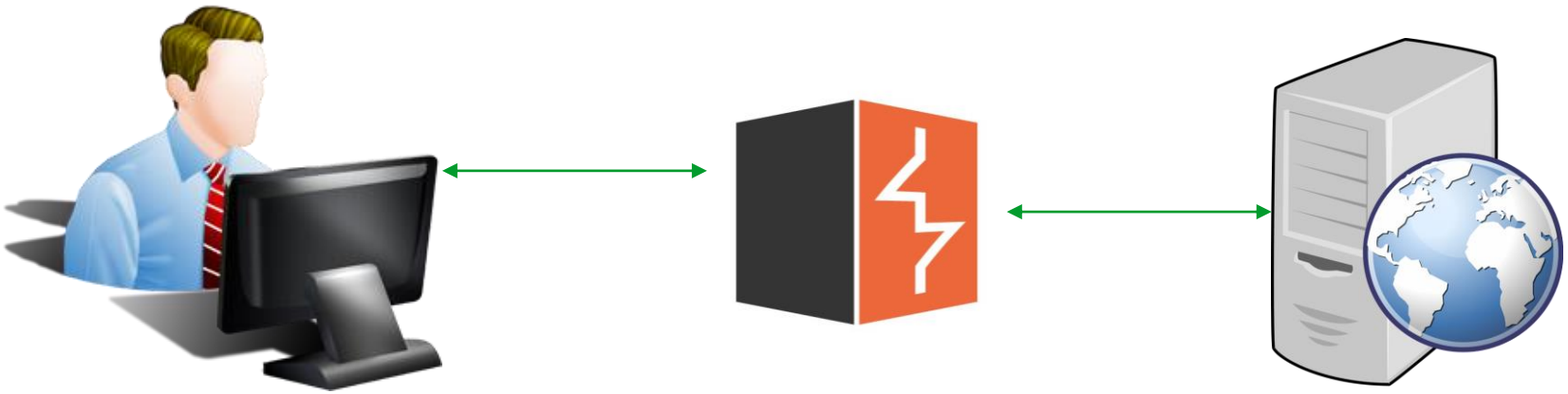
Schedule
10:20 – 12:45
1 hour – lunch break
13:45 – 15:00

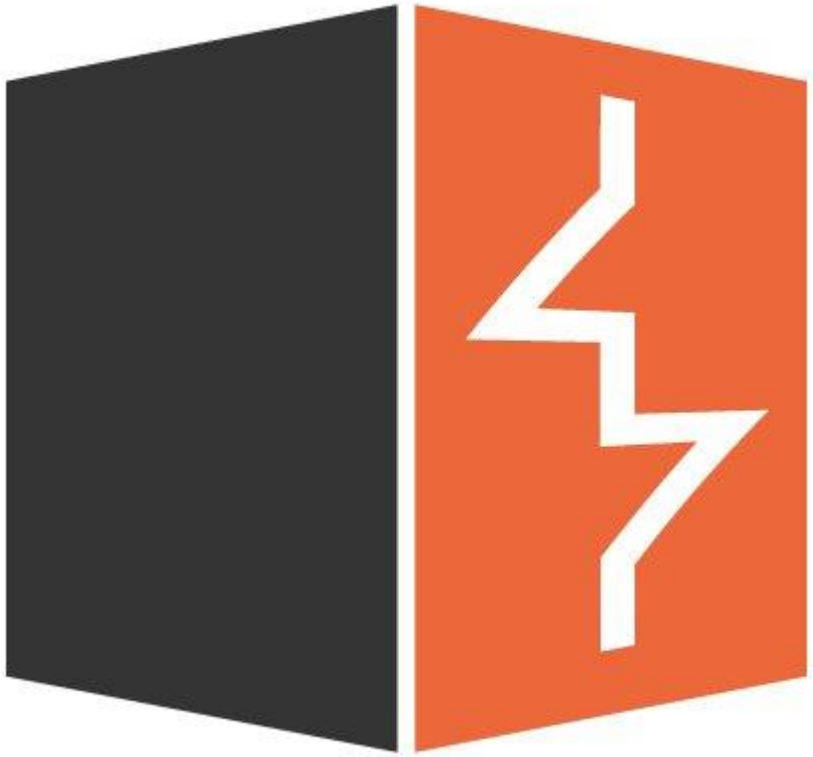
About the workshop

- Kickstarter to burp extension development; not a java or python course
- This is a trimmed version of our 2 day extensive development workshop
- Hands-on development. i.e., we code and debug together here
- Improve your code if you see a chance
- Mistakes are okay. We learn from this more
- Questions are welcome at any given time. Feel free to interrupt

Introduction to Burp Suite

- The de-facto tool for application security testers
- Application proxy with a bunch of additional capabilities
- Burp features
  - Proxy
  - Scanner (Pro)
  - Intruder
  - Repeater
  - Sequencer
  - Decoder
  - Extender





Exercise #1:

Burp Suite crash course

Objective:

- Getting started with burp
- Setting up browser
- Working with intercept options, repeater and more
- Know what to automate using burp extensions
- Analyzing request/response pairs

Burp Extensions

- API to Burp functionalities
- Extensions can be written in:
  - Java
  - Python (JPython)
  - Ruby (JRuby)

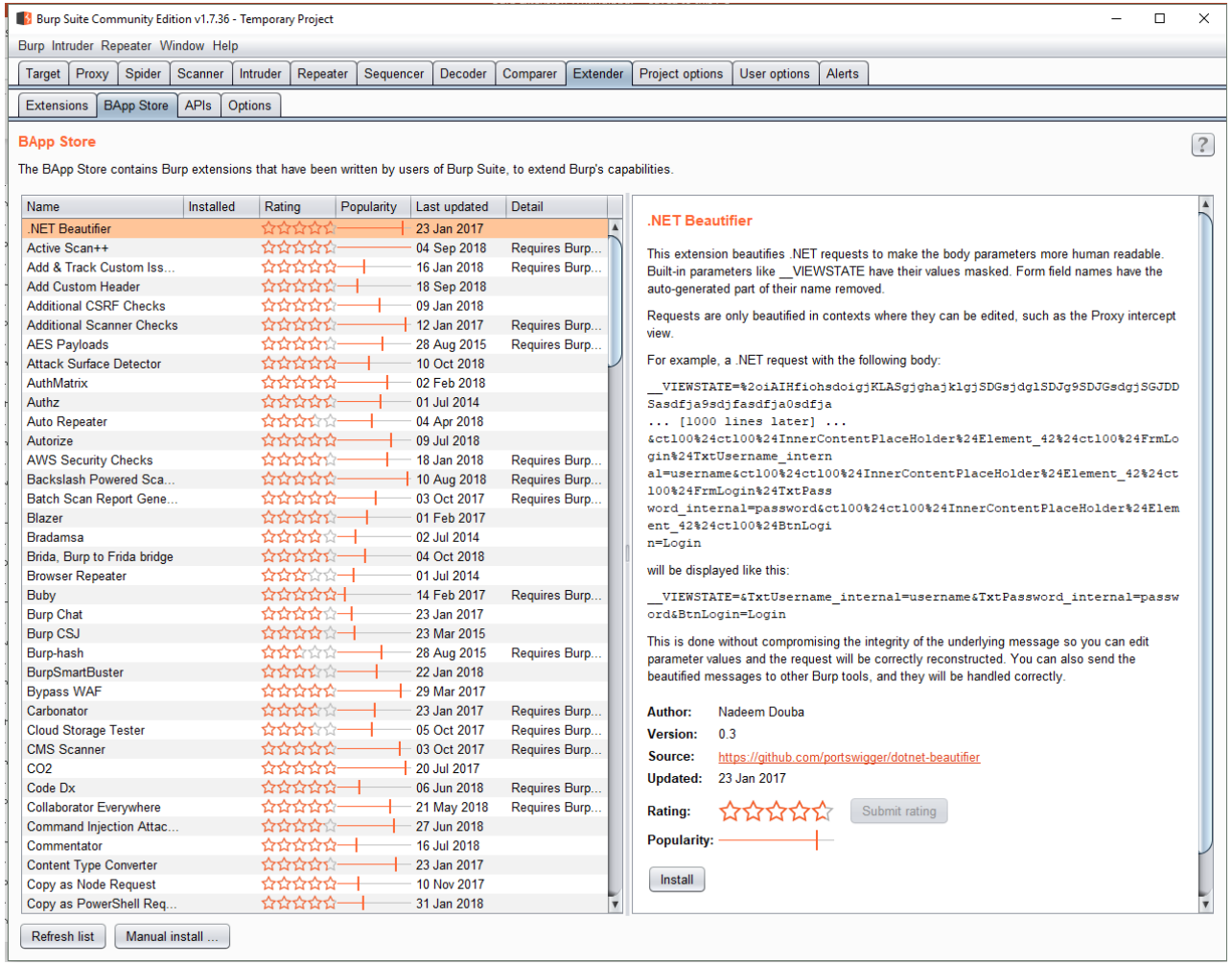
Extender Capabilities

- Process and modify HTTP requests and responses for all Burp tools.
- Access key runtime data, such as the Proxy history, target site map, and Scanner issues.
- Initiate actions like scanning and spidering.
- Implement custom scan checks and register scan issues.
- Customize the placement of attack insertion points within scanned requests.
- Provide custom Intruder payloads and payload processors.
- Query and update the Suite-wide target scope.
- Query and update the session handling cookie jar.
- Implement custom session handling actions.

Extender Capabilities (cont..)

- Add custom tabs and context menu items to Burp's user interface.
- Use Burp's native HTTP message editor within your own user interface.
- Customize Burp's HTTP message editor to handle data formats that Burp does not natively support.
- Analyze HTTP requests and responses to obtain headers, parameters, cookies, etc.
- Build, modify and issue HTTP requests and retrieve responses.
- Read and modify Burp's configuration settings.
- Save and restore Burp's state.

BApp Store



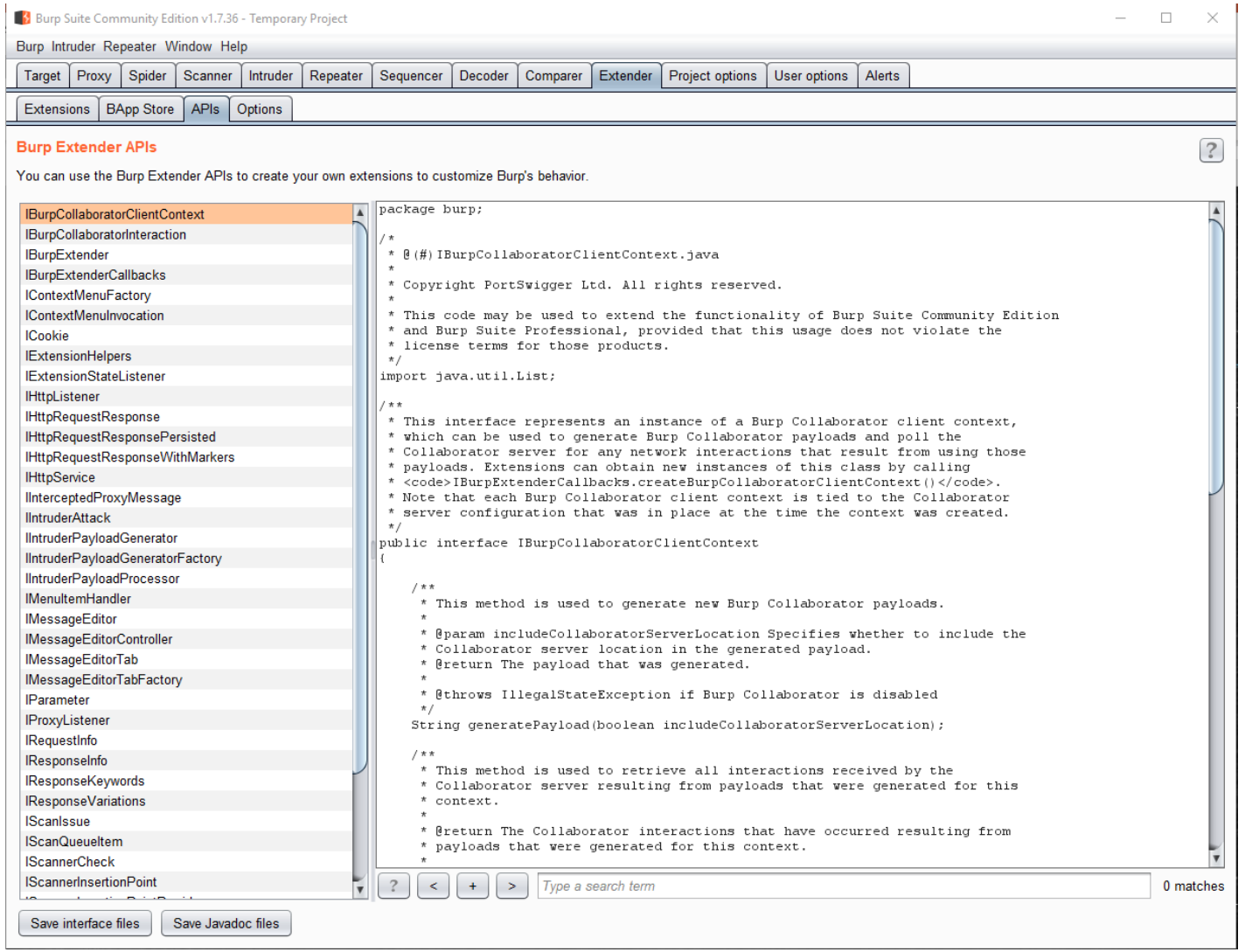
https://portswigger.net/bappstore

Lab Setup

- Burp Suite Community Edition
  - https://portswigger.net/burp/
- Extender APIs
- Test Application
- Development Environments
  - Java
  - Python

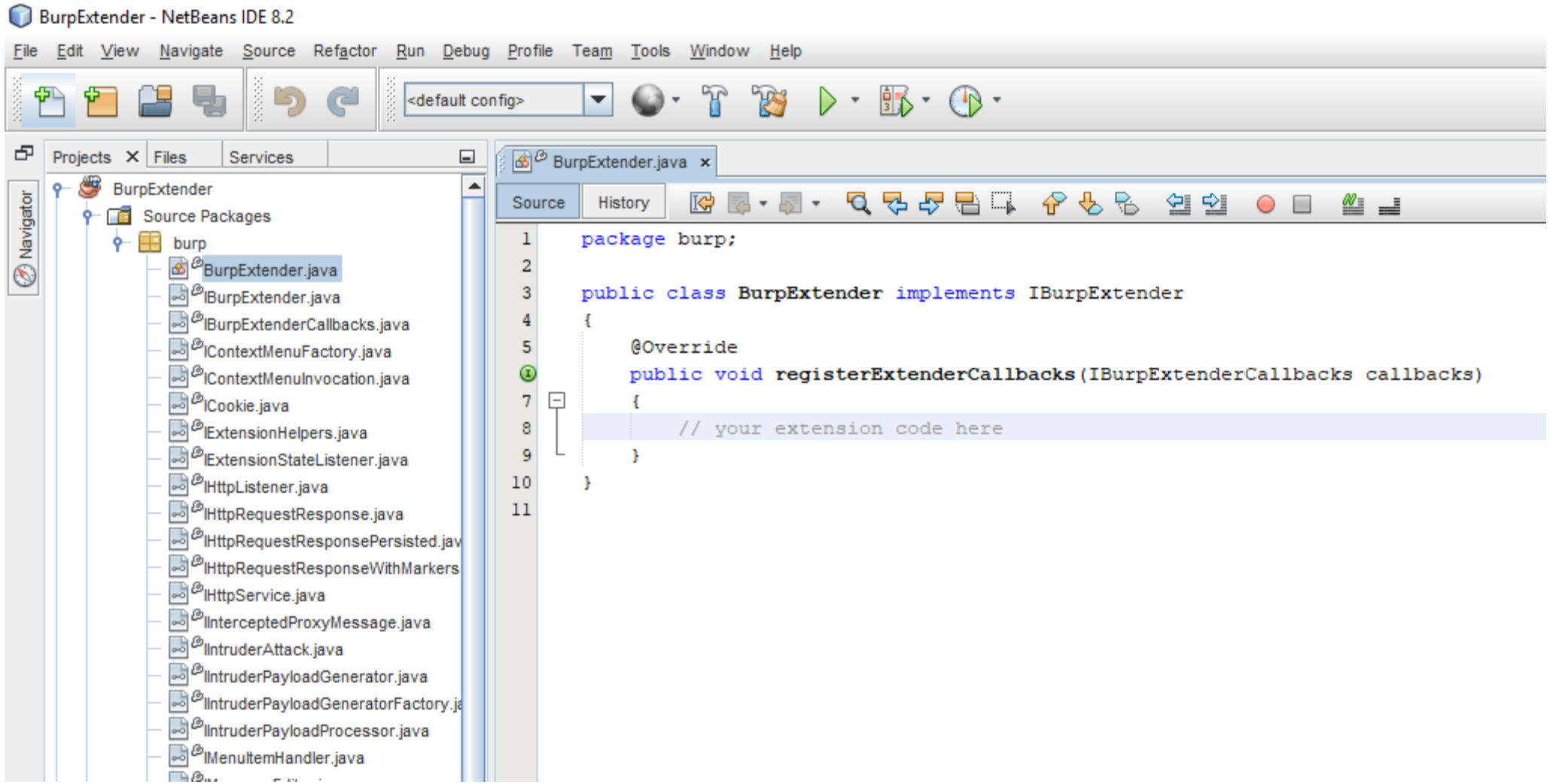
- Test application : http://10.191.35.228/xvwa

Extender APIs



Setup for Java

- Create a project in your favourite IDE
- Create a package "burp" and copy all Extender API files
- Create a java class "BurpExtender"
- Download NetBeans project
  - https://portswigger.net/burp/extender/examples/emptyextension.zip
- This workshop will be using NetBeans version 8.2


BurpExtender - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Projects Files Services
Navigator
BurpExtender
Source Packages
burp
BurpExtender.java
IBurpExtender.java
IBurpExtenderCallbacks.java
IContextMenuFactory.java
IContextMenuInvocation.java
ICookie.java
IExtensionHelpers.java
IExtensionStateListener.java
IHttpListener.java
IHttpRequestResponse.java
IHttpRequestResponsePersisted.jav
IHttpRequestResponseWithMarkers
IHttpService.java
IInterceptedProxyMessage.java
IIntruderAttack.java
IIntruderPayloadGenerator.java
IIntruderPayloadGeneratorFactory.ja
IIntruderPayloadProcessor.java
IMenuItemHandler.java
BurpExtender.java
Source History
1 package burp;
2
3 public class BurpExtender implements IBurpExtender
4 {
5 @Override
public void registerExtenderCallbacks(IBurpExtenderCallbacks callbacks)
7 {
8 // your extension code here
9 }
10 }
11

Setup for Python

- Download Jython
  - http://www.jython.org/downloads.html
- Sublime Text
  - https://www.sublimetext.com/

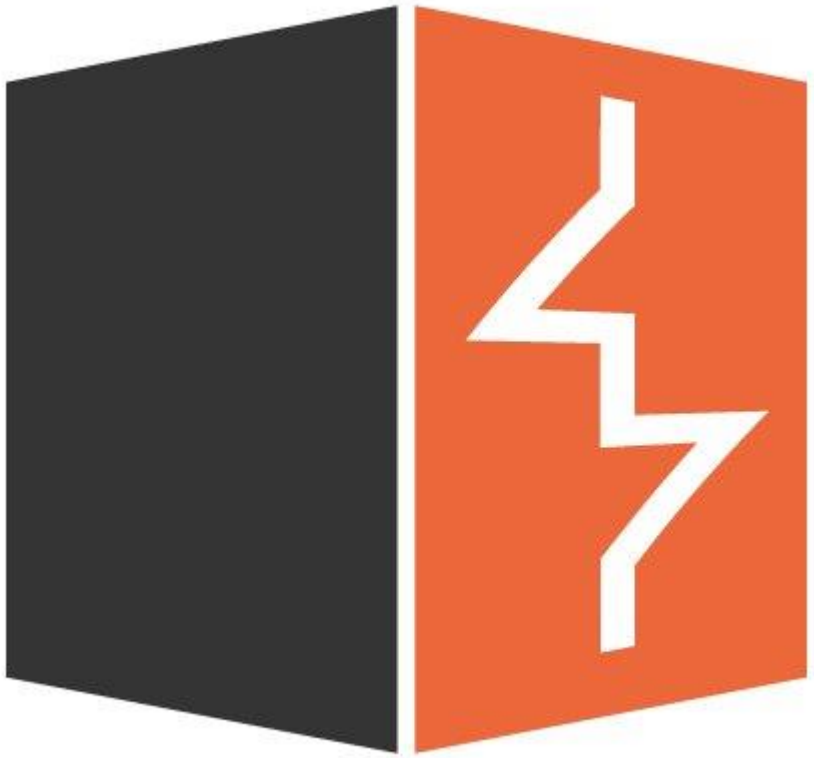
```
1  #Basic Burp Plugin- Python
2  from burp import IBurpExtender
3
4  class BurpExtender(IBurpExtender):
5      def registerExtenderCallbacks(self, callbacks):
6          return
7
```

Note: Because of the way in which Jython and JRuby dynamically generate Java classes, you may encounter memory problems if you load several different Python or Ruby extensions, or if you unload and reload an extension multiple times. If this happens, you will see an error like:

```
java.lang.OutOfMemoryError: PermGen space
```

You can avoid this problem by configuring Java to allocate more PermGen storage, by adding a `-XX:MaxPermSize` option to the command line when starting Burp. For example:

```
java -XX:MaxPermSize=1G -jar burp.jar
```



Exercise #2:

Setting up development environment

Objective:

- Setup environment for Java
- Setup environment for Python

IBurpExtender

- All extensions must implement this interface.
- BurpExtender must implement this interface with one method
  - `void registerExtenderCallbacks(IBurpExtenderCallbacks callbacks)`
- Burp invokes registerExtenderCallbacks()method when an extension is loaded

IBurpExtenderCallbacks

- Consists a set of callback methods for extensions to perform various actions within Burp
- Used to register extension settings
- This interface can handles stdout and stderr

IExtensionHelpers

- Build, analyse and modify HTTP requests and responses
- Toggle request's method between GET and POST
- Do encoding/decoding to request/response data
- Constructs scanner insertion point

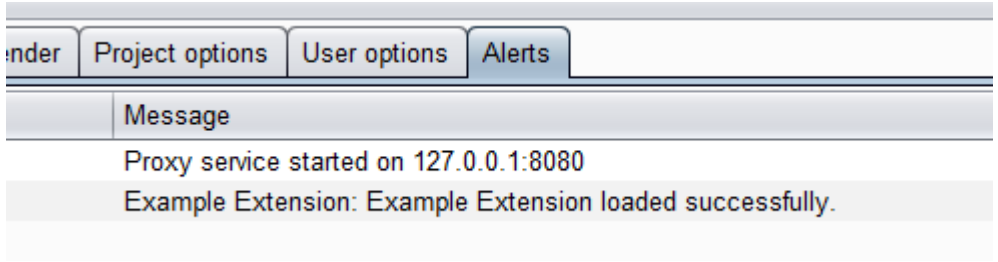
Handling stdout/stderr

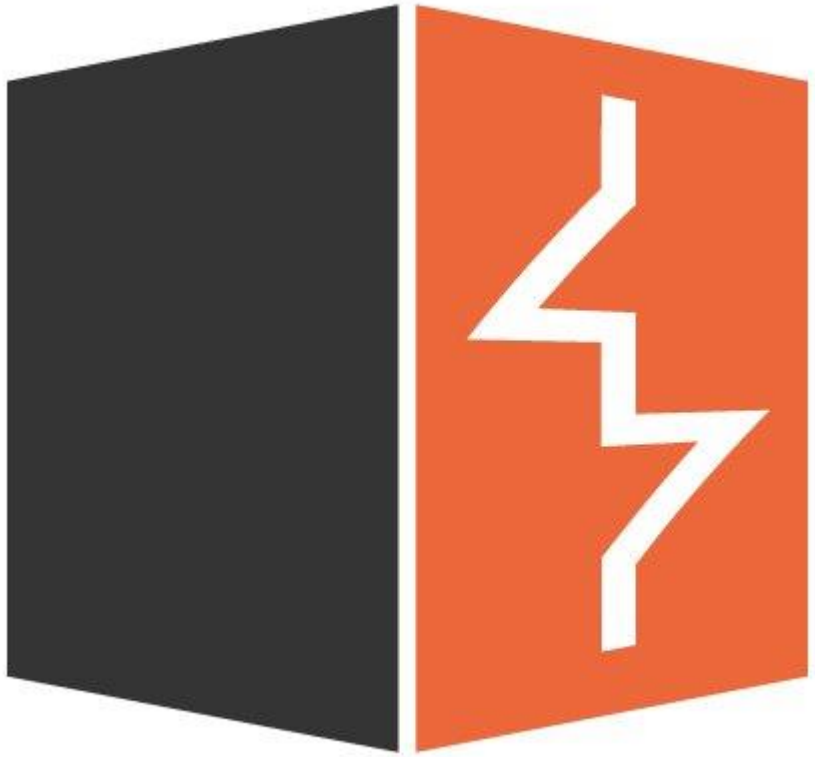
- `callbacks.getStderr()`
- `callbacks.getStdout()`

```
private IExtensionHelpers helpers;
private PrintWriter stdOut, StdErr;

@Override
public void registerExtenderCallbacks(IBurpExtenderCallbacks callbacks)
{
    this.callbacks = callbacks;
    this.helpers = callbacks.getHelpers();
    stdOut = new PrintWriter(callbacks.getStdout(), true);
    StdErr = new PrintWriter(callbacks.getStderr(), true);

}
```

Extension Settings

- `callbacks.setExtensionName(String extName)`
- `callbacks.issueAlert(String msg)`





Exercise #3:

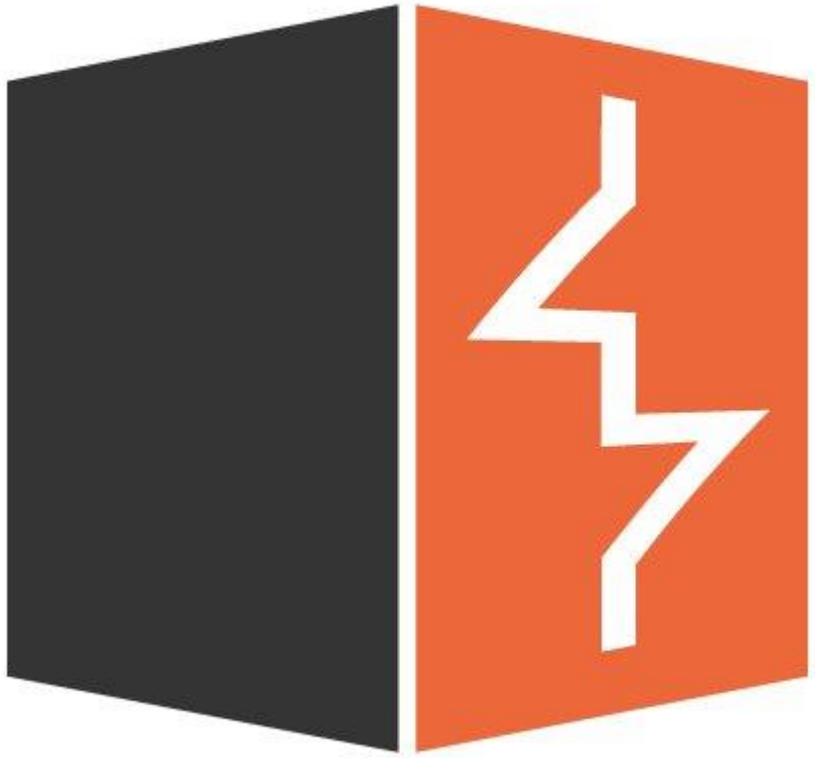
Using stdOut/stdErr for extension with alert on loading

Objective:

- Create extension with name
- Use stdOut and stdErr to show messages
- Issue alert on load

Some useful methods

- IBurpExtenderCallbacks
  - `IHttpRequestResponse makeHttpRequest(IHttpService httpService, byte[] request);`
  - `void registerHttpListener(IHttpListener listener);`
  - `void registerProxyListener(IProxyListener listener);`
- IExtensionHelper
  - `IRequestInfo analyzeRequest(byte[] request);`
  - `IResponseInfo analyzeResponse(byte[] response);`
  - `IParameter getRequestParameter(byte[] request, String parameterName);`
  - `byte[] addParameter(byte[] request, IParameter parameter);`
  - `byte[] toggleRequestMethod(byte[] request);`
  - `byte[] buildHttpMessage(List headers, byte[] body);`
- IRequestInfo
  - `List<String> getHeaders();`
  - `List<IParameter> getParameters();`

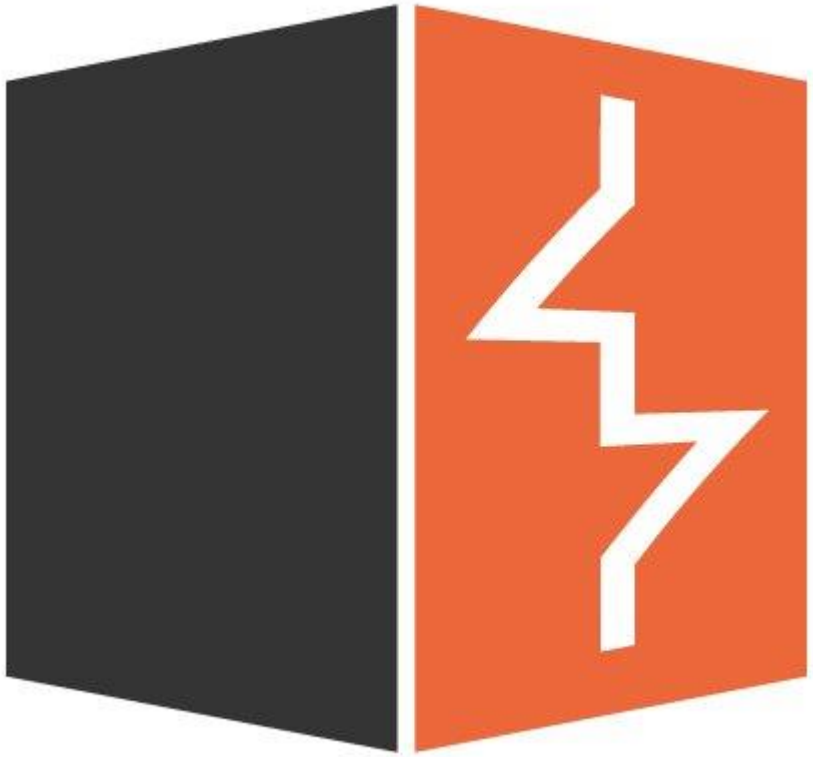


Exercise #4:

Analyze and match from server response

Objective:

- Analyse test application for request and responses
- Find server header in the responses.
- Trigger message if server header found
- Parse server header value and show in messages.



Exercise #5:

Using iMessageEditorTab to analyze/parse HTTP content

Objective:

- Implement iMessageEditorTab
- Show tab if request or response contain specific keyword
- Parse and print messages in iMessageEditorTab

Debugging Tips

- Version updates may change APIs too. Use the latest
- Run burp from terminal
- Memory problems with Jython and Jruby
  - `java.lang.OutOfMemoryError: PermGen space`
- In case of above error, allocate more PermGen storage
  - `java -XX:MaxPermSize=1G -jar burp.jar`

Packaging Tips

```
<target name="package-for-release" depends="jar">
        <property name="store.jar.name" value="YourProjectName"/>
        <property name="store.dir" value="store"/>
        <property name="store.jar" value="${store.dir}/${store.jar.name}.jar"/>
        <echo message="Packaging ${application.title} into a single JAR at ${store.jar}"/>
        <delete dir="${store.dir}"/>
        <mkdir dir="${store.dir}"/>
        <jar destfile="${store.dir}/temp_final.jar" filesetmanifest="skip">
            <zipgroupfileset dir="dist" includes="*.jar"/>
            <zipgroupfileset dir="dist/lib" includes="*.jar"/>
            <manifest>
                <attribute name="Main-Class" value="${main.class}"/>
            </manifest>
        </jar>
        <zip destfile="${store.jar}">
            <zipfileset src="${store.dir}/temp_final.jar"
            excludes="META-INF/*.SF, META-INF/*.DSA, META-INF/*.RSA"/>
        </zip>
        <delete file="${store.dir}/temp_final.jar"/>
    </target>
```

build.xml > Run target >
Other Targets > package-for-release

Thank You

Reach us at:

Sanoop Thomas
s4n7h0@infoseccampus.com
twitter.com/s4n7h0

Samandeep Singh
s.singh@sec-consult.com
twitter.com/samanL33T