Burp Extension Writing

Workshop - OWASP Bay Area

THANKS

/me: Sanoop Thomas @s4n7h0

- Chapter Moderator at Null Singapore
- 8+ years in Information Security
- Created Halcyon IDE – an IDE for Nmap Script Developers
  - https://halcyon-ide.org/
- Host 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

Key Points to Learning

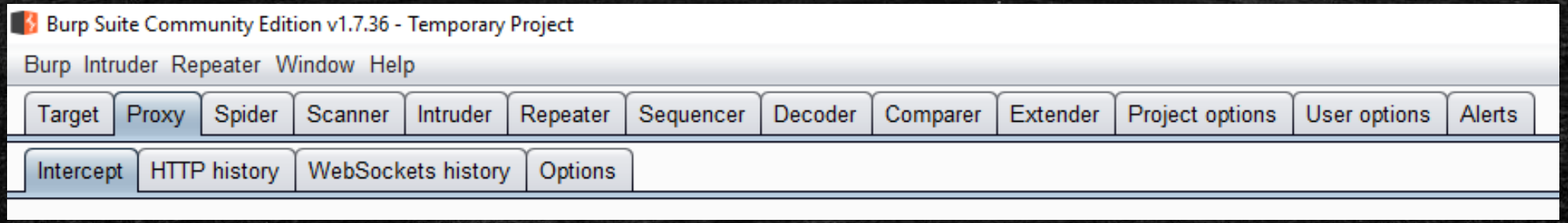
- Ask questions when in doubt
- Experiment with test cases
- Continue writing code after this session

Let’s get started..

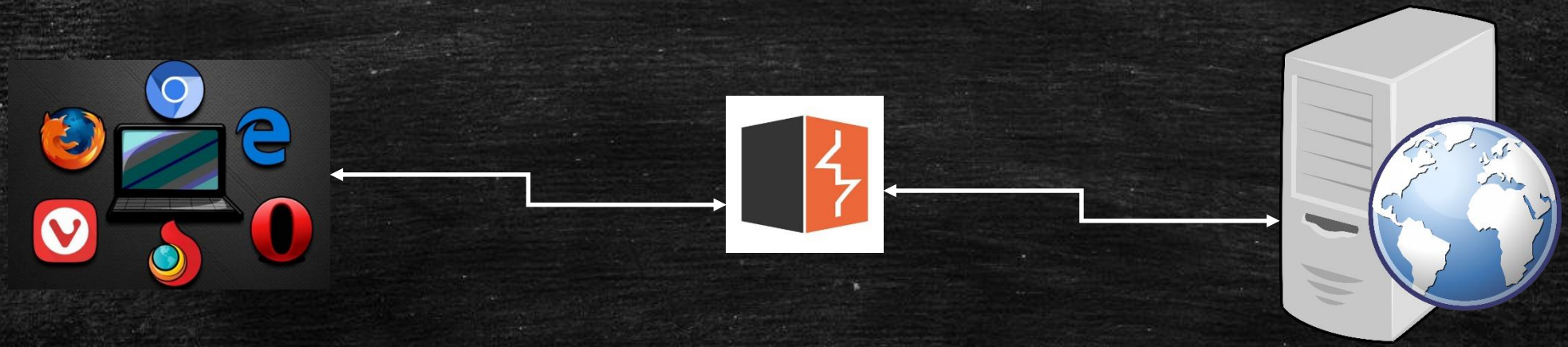
- Quick intro to Burp Suite
- Know the Features
- Understand extender APIs
- Write extensions
  - Java
  - Python

Burp Introduction

- One of the de-facto tool for AppSec
- Proxy at first, but compliments a bunch of other capabilities



How Burp Works



Practice Time

Exercise #: get started with burp

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.

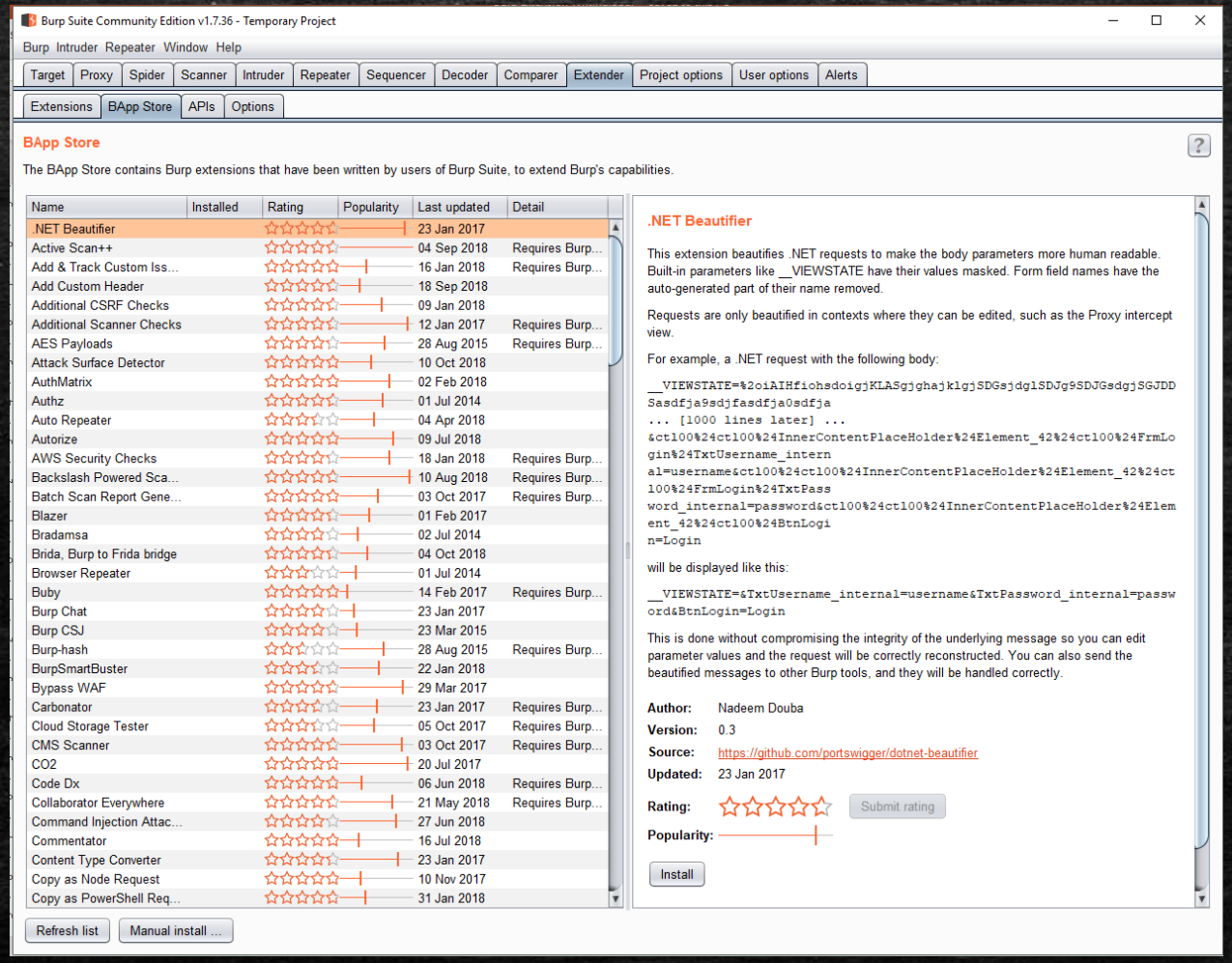
Extender Capabilities (cont..)

- Query and update the Suite-wide target scope.
- Query and update the session handling cookie jar.
- Implement custom session handling actions.
- Add custom tabs and context menu items to Burp's user interface.
- Use Burp's native HTTP message editor within your own user interface.

Extender Capabilities (cont..)

- 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

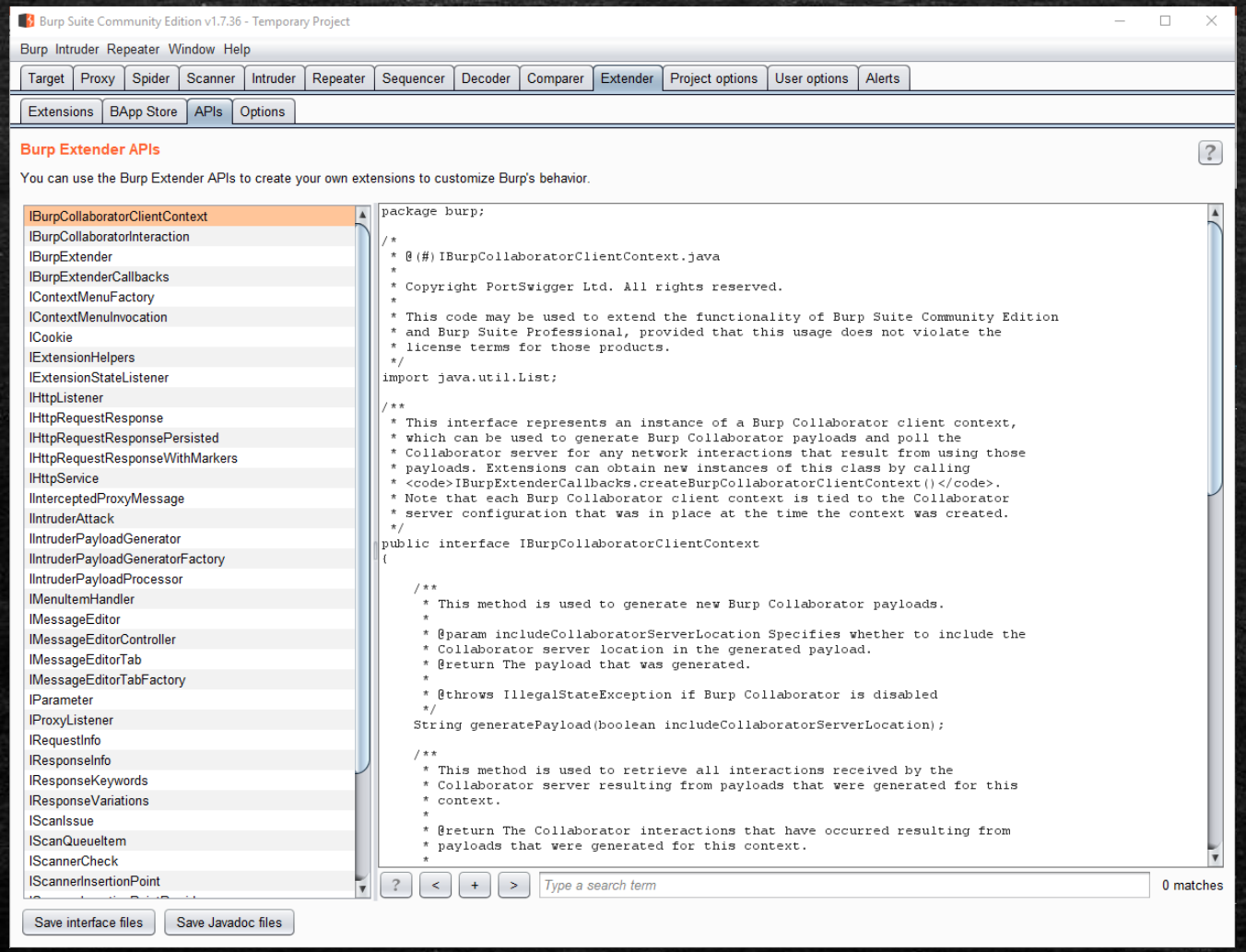


https://portswigger.net/bappstore

Lab Setup

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

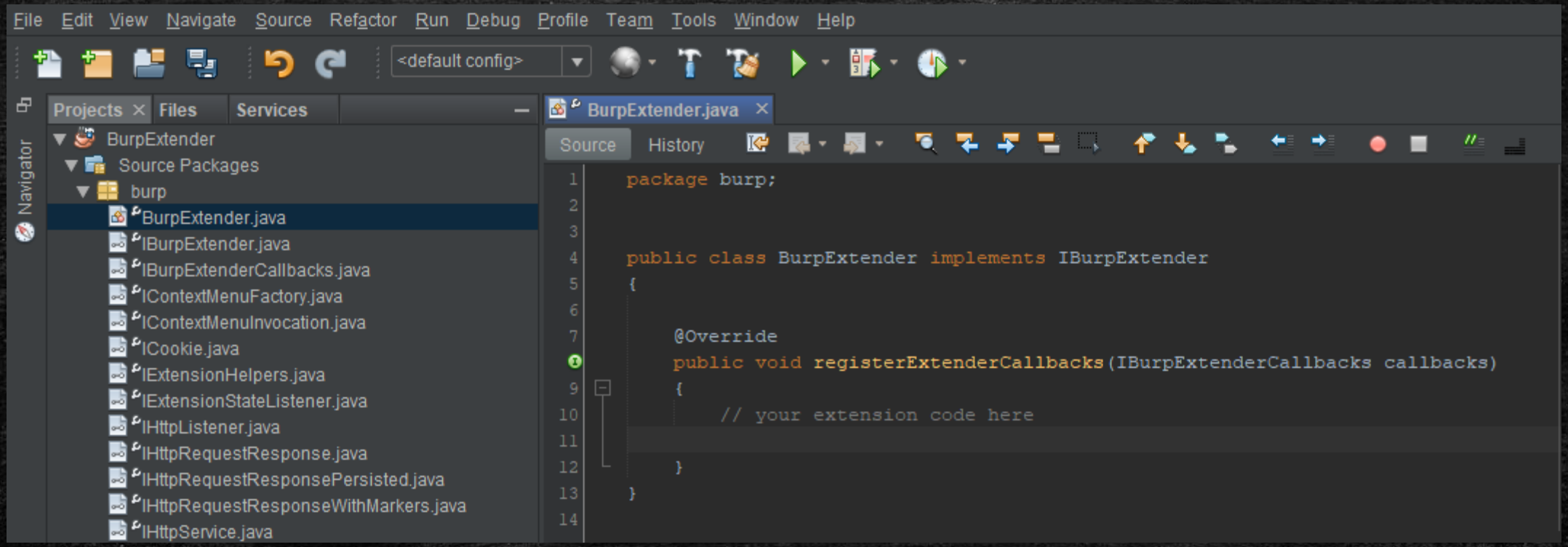
Extender API



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


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.java
IHttpRequestResponseWithMarkers.java
IHttpService.java
BurpExtender.java
Source
History
1 package burp;
2
3
4 public class BurpExtender implements IBurpExtender
5 {
6
7 @Override
8 public void registerExtenderCallbacks(IBurpExtenderCallbacks callbacks)
9 {
10 // your extension code here
11
12 }
13 }
14

Setup for Python

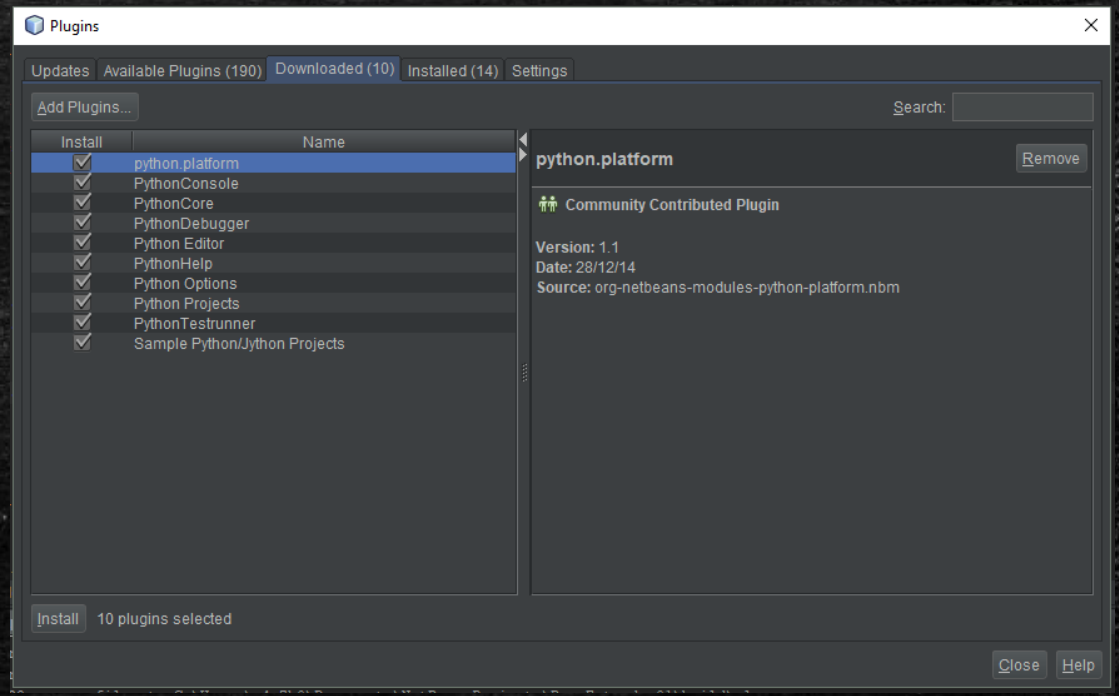
- Download Jython
  - http://www.jython.org/downloads.html
- Python Plugin for NetBeans IDE
  - http://plugins.netbeans.org/plugin/56795

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
```



Practice Time

Exercise #: setup development environment

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()`

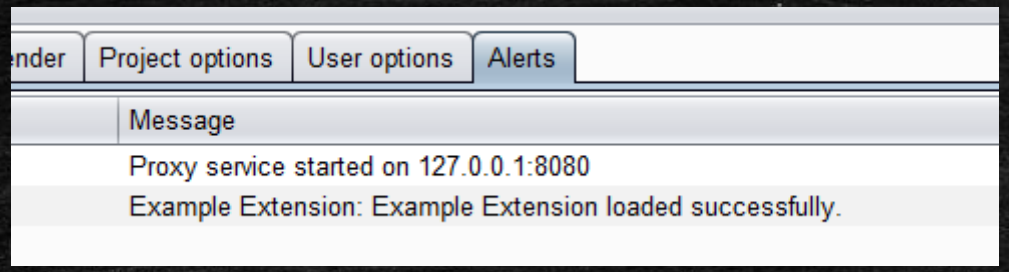
```
private IBurpExtenderCallbacks callbacks;
private PrintWriter stdErr, stdOut;

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

}
```

Extension Settings

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



Practice Time

Exercise #: use stdOut/stdErr for extension with alert on loading

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();

Practice Time

Exercise #: analyse server response for match

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`

Thanks

s4n7h0@infoseccampus.com
https://twitter.com/s4n7h0