Burp Suite: Extensions

Introduction

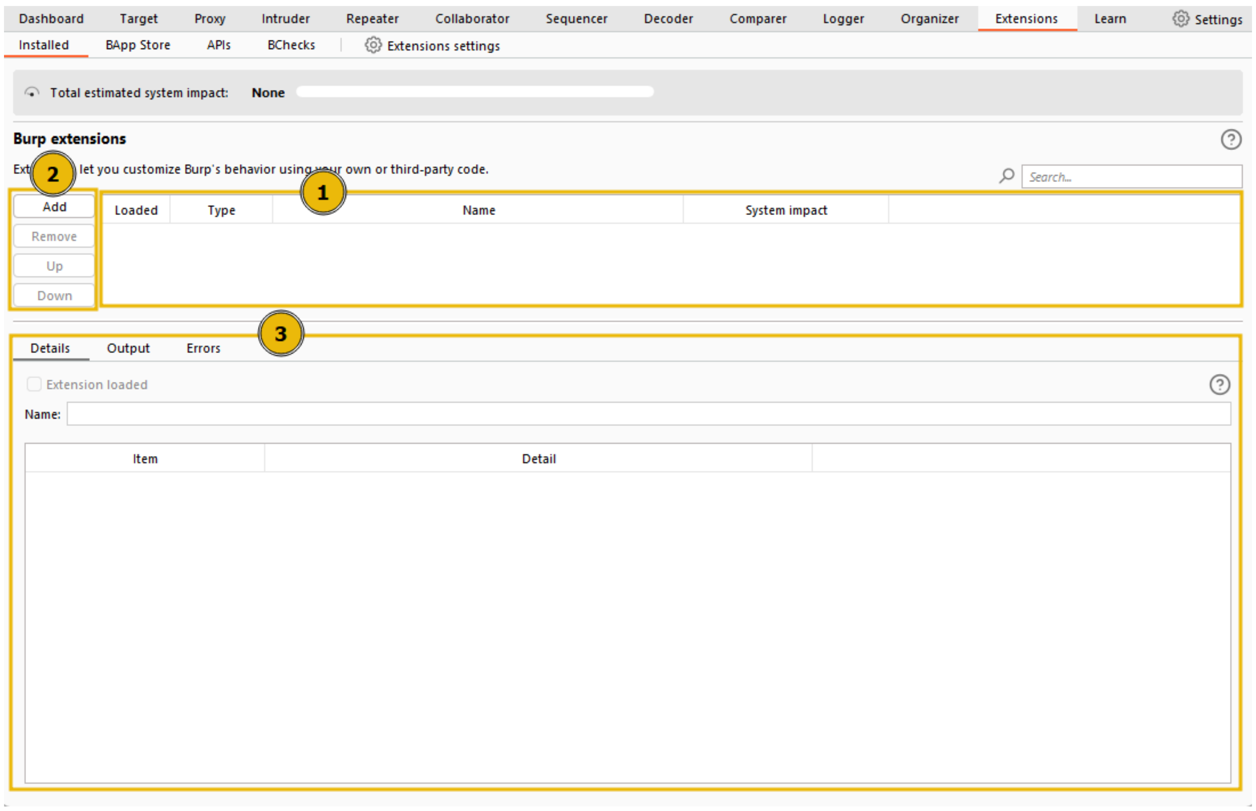
Welcome to the Burp Suite Extensions room!
In this room, we will explore the modular aspects of Burp Suite, focusing on its exposed functionality that allows developers to create additional modules for the framework.

While coding Burp modules is beyond the scope of this module, we will briefly examine the API documentation and discuss the typical process of adding new modules using the Burp Suite BApp Store.

To complete this room, you will not need a target machine, but make sure you have access to a copy of Burp Suite. If you are using the AttackBox, ensure that it is started now by pressing the blue Start AttackBox button at the top-right of the room. Let's dive into the exciting world of Burp Suite extensions and extensibility!

The Extensions Interface

The Extensions interface in Burp Suite provides an overview of the extensions loaded into the tool. Let's take a look at the different components of this interface:



1. Extensions List: The top box displays a list of the extensions that are currently installed in Burp Suite for the current project. It allows you to activate or deactivate individual extensions.
2. Managing Extensions: On the left side of the Extensions interface, there are options to manage extensions:
   - Add: You can use this button to install new extensions from files on your disk. These files can be custom-coded modules or modules obtained from external sources that are not available in the official BApp store.
   - Remove: This button allows you to uninstall selected extensions from Burp Suite.
   - Up/Down: These buttons control the order in which installed extensions are listed. The order determines the sequence in which extensions are invoked when processing traffic. Extensions are applied in descending order, starting from the top of the list and moving down. The order is essential, especially when dealing with extensions that modify requests, as some may conflict or interfere with others.
3. Details, Output, and Errors: Towards the bottom of the window, there are sections for the currently selected extension:
   - Details: This section provides information about the selected extension, such as its name, version, and description.

- Output: Extensions can produce output during their execution, and this section displays any relevant output or results.
- Errors: If an extension encounters any errors during execution, they will be shown in this section. This can be useful for debugging and troubleshooting extension issues.

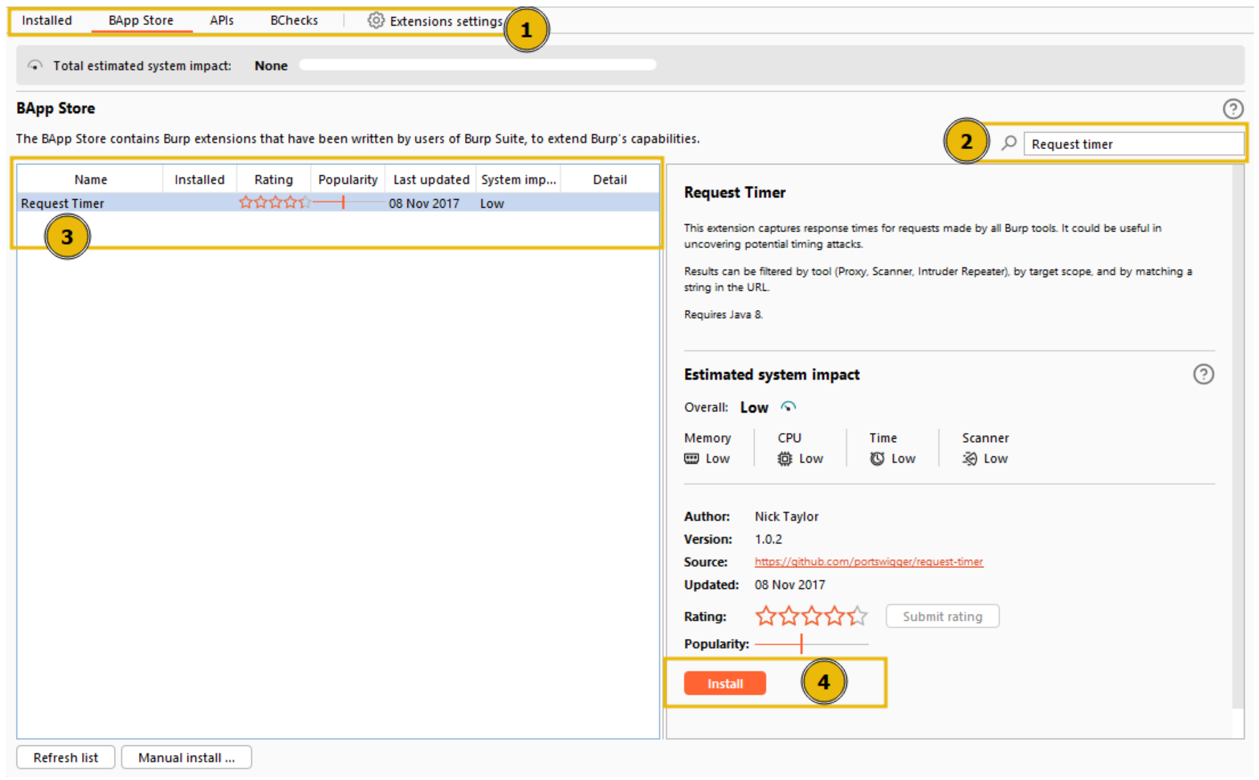
In summary, the Extensions interface in Burp Suite allows users to manage and monitor the installed extensions, activate or deactivate them for specific projects, and view important details, output, and errors related to each extension. By using extensions, Burp Suite becomes a powerful and customizable platform for various security testing and web application assessment tasks.
****
Answer the questions below:

Are extensions invoked in ascending (A) or descending (D) order?
Answer: D
****

The BApp Store
In Burp Suite, the BApp Store (Burp App Store) allows us to easily discover and integrate official extensions seamlessly into the tool. Extensions can be written in various languages, with Java and Python being the most common choices. Java extensions integrate automatically with the Burp Suite framework, while Python extensions require the Jython interpreter.

To get a feel for the BApp store and install a Java extension, let's install the Request Timer extension, written by Nick Taylor. The Request Timer extension enables us to log the time it takes for each request to receive a response. This functionality is particularly useful for identifying and exploiting time-based vulnerabilities. For instance, if a login form takes an extra second to process requests with valid usernames compared to invalid ones, we can use the time differences to determine which usernames are valid.

Follow these steps to install the Request Timer extension from the BApp store:
1. Switch to the BApp Store sub-tab in Burp Suite.
2. Use the search function to find Request Timer. There should only be one result for this extension.
3. Click on the returned extension to view more details.
4. Click the Install button to install the Request Timer extension.



The BApp Store contains Burp extensions that have been written by users of Burp Suite, to extend Burp's capabilities.

Name	Installed	Rating	Popularity	Last updated	System imp...	Detail
Request Timer		☆☆☆☆☆		08 Nov 2017	Low	

Request Timer

This extension captures response times for requests made by all Burp tools. It could be useful in uncovering potential timing attacks.

Results can be filtered by tool (Proxy, Scanner, Intruder Repeater), by target scope, and by matching a string in the URL.

Requires Java 8.

Estimated system impact

Overall: **Low**

Memory	CPU	Time	Scanner
Low	Low	Low	Low

Author:	Nick Taylor
Version:	1.0.2
Source:	https://github.com/portswigger/request-timer
Updated:	08 Nov 2017
Rating:	☆☆☆☆☆
Popularity:	

After successfully installing the extension, you will notice a new tab appearing in the main menu at the top of the Burp Suite interface. Different extensions may have varying behaviours. Some may add new items to right-click context menus, while others create entirely new tabs in the main menu bar.

As this installation is simply an example of using the BApp store, we won't cover how to use the Request Timer here. However, I highly recommend switching to the new tab and exploring the extension to understand its functionalities better. The Request Timer can be valuable in various scenarios, especially when assessing web application security and identifying potential time-based vulnerabilities.
****

Answer the questions below:

Click me to proceed to the next task.
No Answer Needed
****

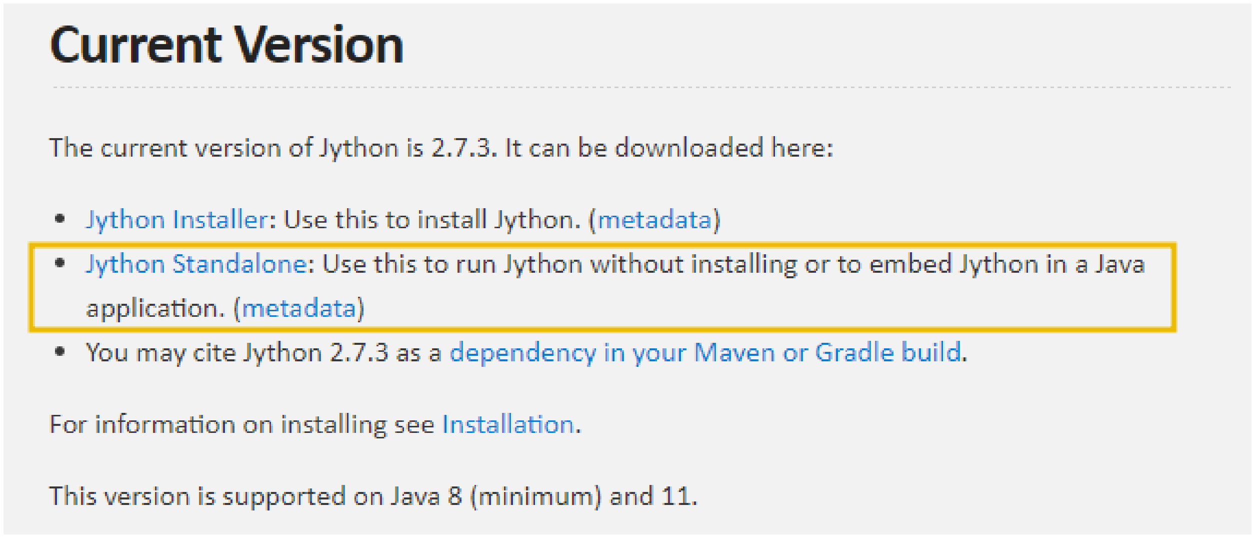
Jython

Note: If you are using the AttackBox, Jython integration has already been done for you, so you can skip this task.

To use Python modules in Burp Suite, we need to include the Jython Interpreter JAR file, which is a Java implementation of Python. The Jython Interpreter enables us to run Python-based extensions within Burp Suite.

Follow these steps to integrate Jython into Burp Suite on your local machine:

1. Download Jython JAR: Visit the [Jython website] and download the standalone JAR archive. Look for the Jython Standalone option. Save the JAR file to a location on your disk.



Current Version

The current version of Jython is 2.7.3. It can be downloaded here:

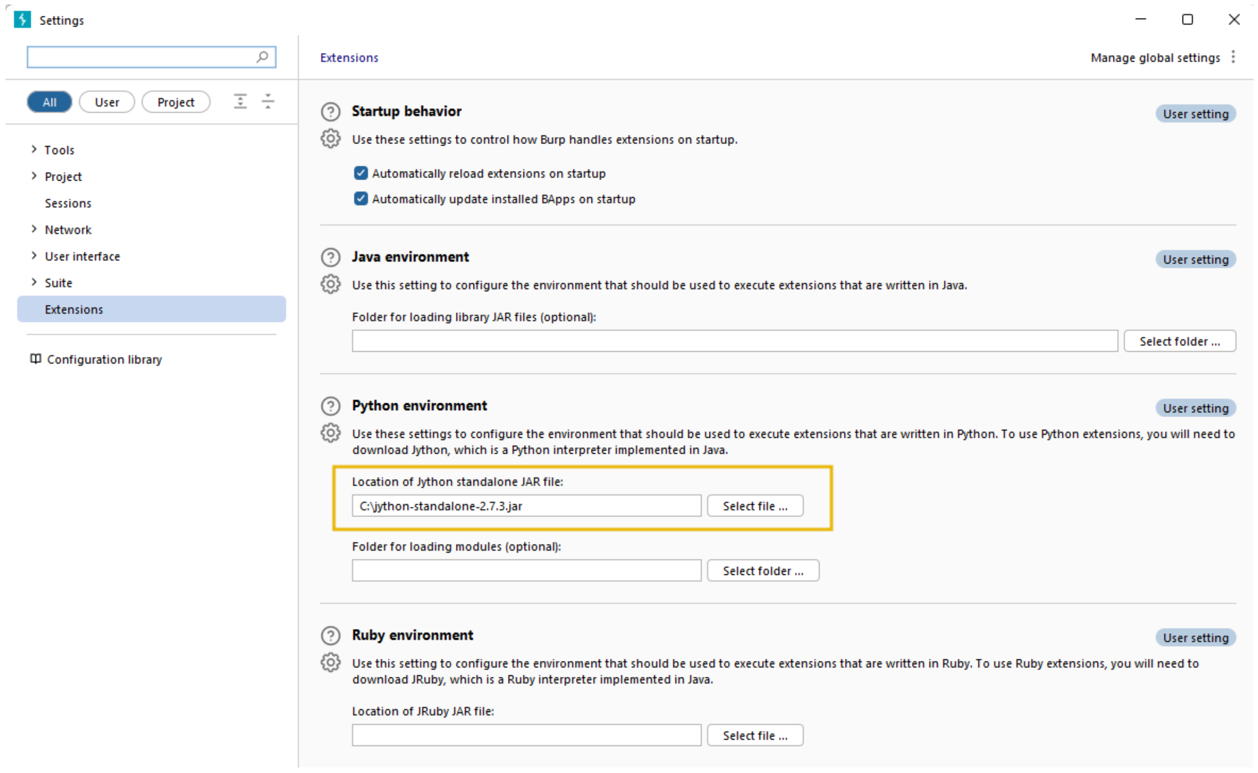
- Jython Installer: Use this to install Jython. (metadata)
- Jython Standalone: Use this to run Jython without installing or to embed Jython in a Java application. (metadata)
- You may cite Jython 2.7.3 as a dependency in your Maven or Gradle build.

For information on installing see Installation.

This version is supported on Java 8 (minimum) and 11.

2. Configure Jython in Burp Suite: Open Burp Suite and switch to the Extensions module. Then, go to the Extensions settings sub-tab.
3. Python Environment: Scroll down to the "Python environment" section.

4. Set Jython JAR Location: In the "Location of Jython standalone JAR file" field, set the path to the downloaded Jython JAR file.



Startup behavior
User setting
Use these settings to control how Burp handles extensions on startup.
- [✓] Automatically reload extensions on startup
- [✓] Automatically update installed BApps on startup

Java environment
User setting
Use this setting to configure the environment that should be used to execute extensions that are written in Java.
Folder for loading library JAR files (optional):
Select folder ...

Python environment
User setting
Use these settings to configure the environment that should be used to execute extensions that are written in Python. To use Python extensions, you will need to download Jython, which is a Python interpreter implemented in Java.
Location of Jython standalone JAR file:
C:\jython-standalone-2.7.3.jar
Select file ...
Folder for loading modules (optional):
Select folder ...

Ruby environment
User setting
Use this setting to configure the environment that should be used to execute extensions that are written in Ruby. To use Ruby extensions, you will need to download JRuby, which is a Ruby interpreter implemented in Java.
Location of JRuby JAR file:
Select file ...

Once you have completed these steps, Jython will be integrated with Burp Suite, allowing you to use Python modules in the tool. This integration significantly increases the number of available extensions and enhances your capabilities in performing various security testing and web application assessment tasks.

Note: The process of adding Jython to Burp Suite is the same for all operating systems, as Java is a multi-platform technology.
****

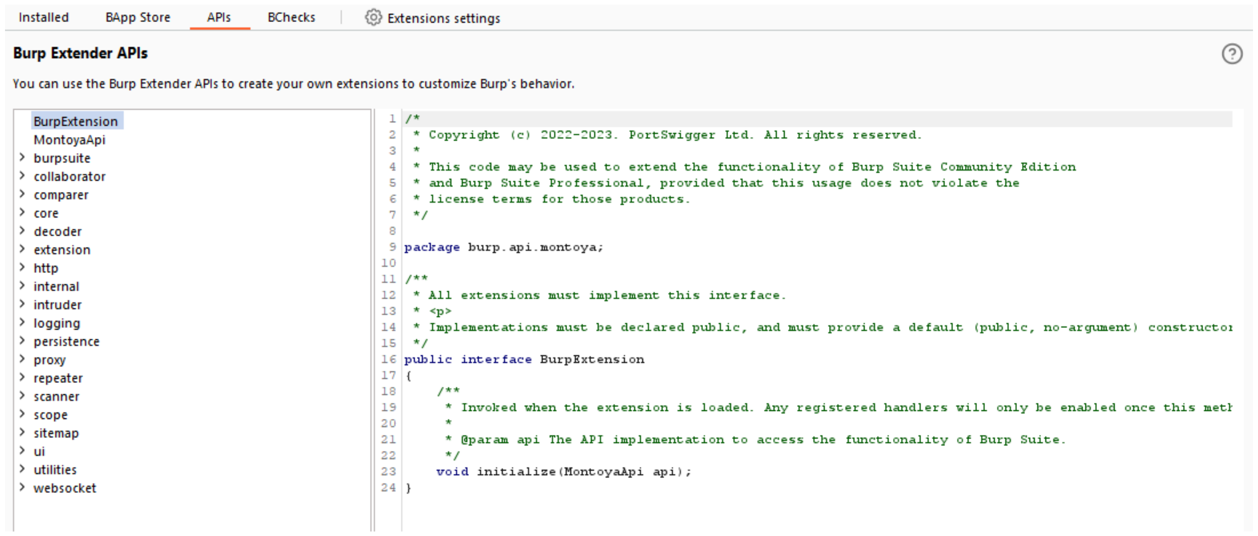
Answer the questions below:

Click me to proceed to the next task.
No Answer Needed
****

The Burp Suite API

In the Burp Suite Extensions module, you have access to a wide range of API endpoints that allow you to create and integrate your modules with Burp Suite. These APIs expose various functionalities, enabling you to extend the capabilities of Burp Suite to suit your specific needs.

To view the available API endpoints, navigate to the APIs sub-tab within the Extensions module. Each item listed in the left-hand panel represents a different API endpoint that can be accessed from within extensions.



Burp Extender APIs

You can use the Burp Extender APIs to create your own extensions to customize Burp's behavior.

- BurpExtension
- MontoyaApi
- burpsuite
- collaborator
- comparer
- core
- decoder
- extension
- http
- internal
- intruder
- logging
- persistence
- proxy
- repeater
- scanner
- scope
- sitemap
- ui
- utilities
- websocket

```
/*
 * Copyright (c) 2022-2023. PortSwigger Ltd. All rights reserved.
 *
 * This code may be used to extend the functionality of Burp Suite Community Edition
 * and Burp Suite Professional, provided that this usage does not violate the
 * license terms for those products.
 */

package burp.api.montoya;

/**
 * All extensions must implement this interface.
 * <p>
 * Implementations must be declared public, and must provide a default (public, no-argument) constructo
 */
public interface BurpExtension
{
    /**
     * Invoked when the extension is loaded. Any registered handlers will only be enabled once this meth
     *
     * @param api The API implementation to access the functionality of Burp Suite.
     */
    void initialize(MontoyaApi api);
}
```

The Extensions APIs give developers significant power and flexibility when writing custom extensions. You can use these APIs to seamlessly interact with Burp Suite's existing functionality and tailor your extensions to perform specific tasks.

Burp Suite supports multiple languages for writing extensions, such as:

- Java (Natively): You can directly use Java to write extensions for Burp Suite, taking advantage of the powerful APIs available.
- Python (via Jython): If you prefer Python as your programming language, you can utilize Jython, which is a Java implementation of Python to create Burp Suite extensions.
- Ruby (via JRuby): Developers familiar with Ruby can leverage JRuby, a Java implementation of Ruby, to build Burp Suite extensions.

It's important to note that coding your extensions for Burp Suite can be a complex task, and it goes beyond the scope of this module. However, suppose you are interested in exploring this area further and creating custom extensions. In that case, PortSwigger provides a comprehensive reference that is an excellent resource for developing Burp Suite extensions.

To learn more about Burp Suite extension development and to access the detailed reference, you can visit PortSwigger's official documentation. This resource will provide the information and guidance needed to create powerful and customized extensions that enhance your experience with Burp Suite.

****
Answer the questions below:

Click me to proceed to the next task.
No Answer Needed
****

Conclusion

Congratulations! You have completed the Burp Suite Extensions Room. By going through this module, you have gained valuable insights into how to use Extensions to enhance the functionality of Burp Suite.

You now have a solid understanding of how to:

1. Install and manage extensions from the BApp store to extend Burp Suite's capabilities.
2. Integrate Jython to leverage Python modules within Burp Suite.
3. Explore the APIs allowing you to develop custom Java, Python, or Ruby modules.

With these skills, you can utilize the Burp Suite Extensions module to its fullest potential in your web application security testing and vulnerability assessments.