

burpsuite 之 Scanner

1、简介

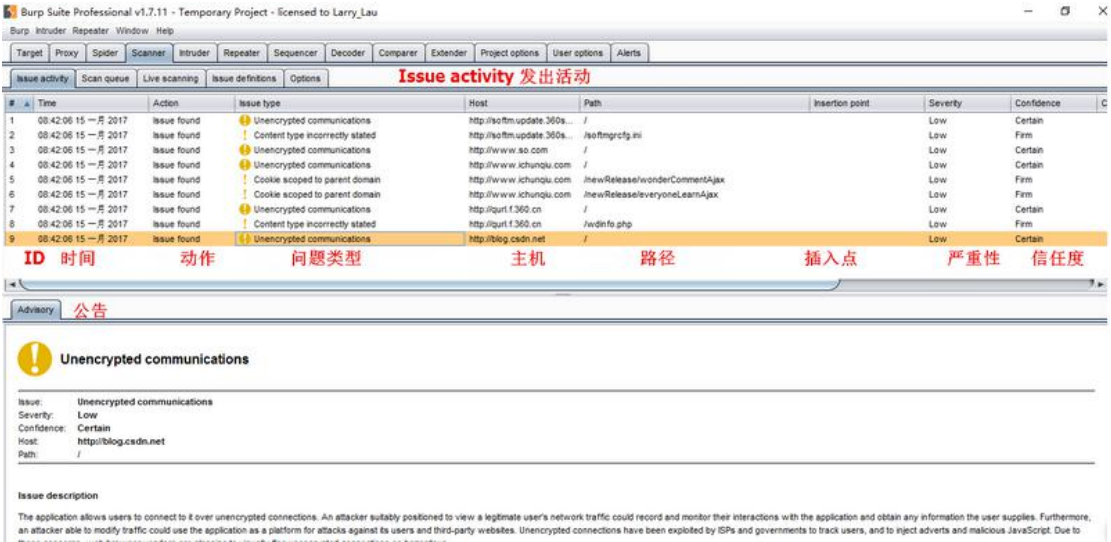
Burp Scanner 是一个进行自动发现 web 应用程序的安全漏洞的工具。它是为渗透测试人员设计的, 并且它和你现有的手动执行进行的 web 应用程序半自动渗透测试的技术方法很相似。

使用的大多数的 web 扫描器都是单独运行的: 你提供了一个开始 URL, 单击||go||,然后注视着进度条的更新直到扫描结束, 最后产生一个报告。Burp Scanner 和这完全不同, 在攻击一个应用程序时它和你执行的操作紧紧的结合在一起。让你细微控制着每一个扫描的请求, 并直接反馈回结果。

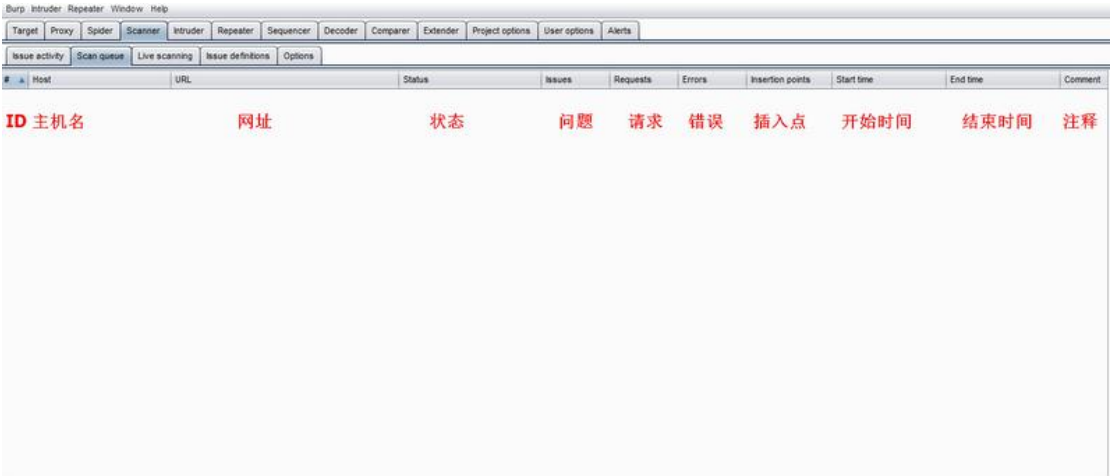
Burp Scanner 可以执行两种扫描类型: 主动扫描(Active scanning), 被动扫描(Passive scanning)。

2、模块说明

2.1：Issue activity



2.2：Scan queue 扫描队列，这里将显示扫描队列的状态 进度 结果等。



主要包含以下内容：

1. 索引号的项目，反映该项目的添加顺序。
2. 目的地协议，主机和 URL 。
3. 该项目的当前状态，包括完成百分比。

4. 项目扫描问题的数量（这是根据所附的最严重问题的重要性和彩色化）。
5. 在扫描项目的请求数量进行。
6. 网络错误的数目遇到的问题。
7. 为项目创建的插入点的数量。

2.3: Live scanning

实时扫描可让您决定哪些内容通过使用浏览器的目标应用，通过 BurpProxy 服务器进行扫描。您可以实时主动扫描设定 live active scanning(积极扫描)和 live passive（被动扫描）两种扫描模式。

Live Active Scanning：积极扫描。当浏览时自动发送漏洞利用代码。



Live Passive Scanning：被动扫描。只分析流量不发送任何请求。



2.4: Issue Definitions

漏洞列表，列出了 burp 可以扫描到的漏洞详情

Target	Proxy	Spider	Scanner	Intruder	Repeater	Sequencer	Decoder	Comparer	Extender	Project options	User options	Alerts
Issue activity	Scan queue	Live scanning	Issue definitions	Options								

Issue Definitions 漏洞列表

This listing contains the definitions of all issues that can be detected by Burp Scanner. 此列表包含了所有的问题，可以通过Burp扫描仪检测的定义

Name	Typical severity	Type index
OS command injection	High	0x00100100
SQL injection	High	0x00100200
SQL injection (second order)	High	0x00100210
ASP.NET tracing enabled	High	0x00100202
File path traversal	High	0x00100300
XML external entity injection	High	0x00100400
LDAP injection	High	0x00100500
SPPath injection	High	0x00100600
XML injection	Medium	0x00100700
ASP.NET debugging enabled	Medium	0x00100800
HTTP PUT method is enabled	High	0x00100900
Out-of-band resource load (HTTP)	High	0x00100a00
File path manipulation	High	0x00100b00
PHP code injection	High	0x00100c00
Server-side JavaScript code injection	High	0x00100d00
Perl code injection	High	0x00100e00
Ruby code injection	High	0x00100f00
Python code injection	High	0x00100f10
Expression Language injection	High	0x00100f02
Unidentified code injection	High	0x00101000
Server-side template injection	High	0x00101002
SSI injection	High	0x00101100
Cross-site scripting (stored)	High	0x00200100
HTTP response header injection	High	0x00200200
Cross-site scripting (reflected)	High	0x00200300
Client-side template injection	High	0x00200302
Cross-site scripting (DOM-based)	High	0x00200310
Cross-site scripting (reflected DOM-based)	High	0x00200311

OS command injection

Description:

Operating system command injection vulnerabilities arise when an application incorporates user-controllable data into a command that is processed by a shell command interpreter. If the user data is not strictly validated, an attacker can use shell metacharacters to modify the command that is executed, and inject arbitrary further commands that will be executed by the server.

OS command injection vulnerabilities are usually very serious and may lead to compromise of the server hosting the application, or of the application's own data and functionality. It may also be possible to use the server as a platform for attacks against other systems. The exact potential for exploitation depends upon the security context in which the command is executed, and the privileges that this context has regarding sensitive resources on the server.

Remediation

If possible, applications should avoid incorporating user-controllable data into operating system commands. In almost every situation, there are safer alternative methods of performing server-level tasks, which cannot be manipulated to perform additional commands than the one intended.

If it is considered unavoidable to incorporate user-supplied data into operating system commands, the following two layers of defense should be used to prevent attacks.

- The user data should be strictly validated. Ideally, a whitelist of specific accepted values should be used. Otherwise, only short, alphanumeric strings should be accepted. Input containing any other data, including any conceivable shell metacharacter or whitespace, should be rejected.
- The application should use command APIs that launch a specific process via its name and command-line parameters, rather than passing a command string to a shell interpreter that supports command chaining and redirection. For example, the Java API Runtime.exec and the ASP.NET API Process.Start do not support shell metacharacters. This defense can mitigate the impact of an attack even in the event that an attacker circumvents the input validation defenses.

Typical severity

High

3: Options

包含 Burp 扫描选项进行攻击的插入点，主动扫描引擎，主动扫描优化，主动扫描区和被动扫描区域。

3.1: Attack Insertion Points

Attack Insertion Points 攻击的插入点

Place attacks into the following locations within requests:

将攻击由放置到下列位置

- ☒ URL parameter values - 将攻击代码放在URL中。
- ☒ Body parameter values - 将攻击代码放在正文中，包括标准形式生成的参数参数值，属性的多重编码的参数，如上传的文件名，XML参数值和属性，和JSON值。
- ☒ Cookie parameter values - 将攻击代码放在HTTP Cookie的值中。
- ☒ Parameter name - 任意添加的参数名称。
- ☒ HTTP headers - 在引用和用户代理请求头的值。测试这些插入点通常可以检测如SQL注入或跨站脚本持续在日志记录功能的问题。
- ☒ Entire body (for relevant content types) - 整个身体（相关内容类型）
- ☒ Client-side template injection - 在引用和用户代理请求头的值。
- ☒ AMF string parameters - 内AMF编码的参数的任何字符串参数数据的值。
- ☒ URL path filename - URL路径文件名
- ☒ URL path folders - URL路径文件夹

Change parameter locations (causes many more scan requests) - 修改参数的位置（会产生更多的扫描请求），可以一些防火墙或过滤器。

Use nested insertion points - 使用嵌套插入点。嵌套的插入点，会使用一个插入点的思维包含可识别的格式的数据。例如，一个URL参数可能包含Base64编码数据，并且解码后的值可能又包含JSON或XML数据。与使用启用嵌套插入点的选项，Burp会为输入在每个嵌套级别中的每个单独的项目组合的插入点。

Limit maximum request size (per host/request) - 限制每个请求的最大插入点。

Step server-side injection tests for these parameters - 跳过列表中参数的测试。设定让指定请求参数的Burp应该跳过某些测试。

Add	Remove	Parameter	Name	Match type	Expression
+	-	Cookie	name	Match regex	expression
+	-	Cookie	name	is	not-matched

启用 参数 选项 匹配类型 表达式

3.2: Active Scanning Engine

Active Scanning Engine 主动扫描引擎

These settings control the engine used for making HTTP requests when doing active scanning. 控制用来做主动扫描时发出HTTP请求的线程、时间间隔等

Number of threads: 10 - 线程。控制发送请求的线程。

Number of retries on network failure: 3 - 在遇到网络请求超时重新请求的次数。

Pause before retry (milliseconds): 2000 - 重试失败的请求的时间间隔。

Throttle between requests (milliseconds): 500 - 请求之间的时间间隔。

☐ Add random variations to throttle - 添加随机的变化到请求中。增加隐蔽性。

☒ Follow redirections where necessary - 是否必要时跟随重定向。因为某些应用程序的问题重定向到包含您提交的参数值的第三方网址，B

3.3: Active Scanning Optimization

Active Scanning Optimization 主动扫描优化

These settings let you control the behavior of the active scanning engine to reflect the objectives of the scan and the nature of the target application. See the detailed help for more information about each option.

Scan speed: **Normal**

Scan accuracy: **Normal**

Use intelligent attack selection

这些设置允许您控制活动扫描逻辑的行为以反映扫描的对象和target应用程序的性质。有关每个选项的详细信息，请参阅详细帮助。

Scan speed: 扫描速度。
fast: 快速扫描。发送请求较少，扫描一些基本漏洞。
normal: 正常扫描。适用于大部分扫描。
thorough: 深入扫描。发起更多请求，检查更多的衍生类型的漏洞
Scan accuracy: 扫描准确性。
Minimize false negatives: 进行重试较少，因此更可能报告假阳性的问题，但也不太可能会错过由于不一致的应用程序行为的真正问题。
normal: 正常扫描。适用于大部分扫描。
Minimize false positives: 进行更多的试，所以是不太可能报告假阳性的问题，但可能会因此错误地错过了一些真正的问题，因为有些重试请求可能只是碰巧不返回结果是测试。
Use intelligent attack selection: 使用智能攻击选择。

3.4: Active Scanning Areas

Active Scanning Areas 活动扫描区域

These settings control the types of checks performed during active scanning.

☒ SQL injection

☒ Error-based

☒ Time-delay checks

☒ Boolean condition checks

☒ OS command injection

☒ Informed

☒ Blind

☒ Server-side code injection

☒ Server-side template injection (requires reflected XSS)

☒ Reflected XSS

☒ Stored XSS

☒ Reflected DOM issues

☒ Stored DOM issues

☒ File path traversal / manipulation

☒ External / out-of-band interaction

☒ HTTP header injection

☒ XML / SOAP injection

☒ LDAP injection

☒ Cross-site request forgery

☒ Open redirection

☒ Header manipulation

☒ Server-level issues

☐ Input returned in responses (reflected)

☐ Input returned in responses (stored)

☒ MSSQL-specific checks

☒ Oracle-specific checks

☒ MySQL-specific checks

Select all

Select none

设定主动扫描的范围。设置在扫描过程中需要检测的漏洞类型。

SQL injection: SQL注入
error-based: 基于错误的SQL注入
mssql-specific tests: mssql数据库的SQL注入
time-delay tests: 基于延时的SQL注入
oracle-spcofic tests: oracle数据库SQL注入
boolean condition tests: 基于布尔的SQL注入
mysql-spcofic tests: mysql数据库的注入
OS command injection: 操作系统命令注入执行
informed
blind
Server side code injection : 服务器端代码注入
Server-side template injection(reuires reflected XSS):服务端的注入（反射型xss）
Reflected XSS: 跨站点脚本
Stored XSS: 存储型的跨站点脚本
File path traversal/manipulation: 文件路径遍历/可编辑
External/out-of-band interaction : 外部/带外交互
HTTP header injection : HTTP头注入
XML/SOAP injection : XML/SOAP注射
LDAP injectionDAP 注入

3.5: Passive Scanning Areas

Passive Scanning Areas 被动扫描区

These settings control the types of checks performed during passive scanning.

☒ Headers

☒ Forms

☒ Links

☒ Parameters

☒ Cookies

☒ Server-level issues

☒ MIME type

☒ Caching

☒ Information disclosure

☒ Frameable responses ("Clickjacking")

☒ ASP.NET ViewState

Select all

Select none

这些设置控制被动扫描期间执行的检查类型

Headers:头
Forms:表格
Links:链接
Parameters:参数
Cookies:Cookie
Server-level issues:服务端漏洞
MIME type: MIME类型（多用途互联网邮件扩展类型）
Caching:缓存
Information disclosure:信息泄露
Frameable responses:耐燃反应（“点击劫持”）
ASP.NET ViewState:ASP.NET视图

3.6: Static Code Analysis

?

Static Code Analysis 静态代码分析

⚙

These settings control the types of scanning that will include static analysis of executable code. Note that static analysis can consume large amounts of memory and processing, and so it may be desirable to restrict static analysis to key targets of interest.

☒ Active scanning only

☐ Active and passive scanning

☐ Don't perform static code analysis

只进行主动扫描

主动扫描+被动扫描

不进行静态分析

Maximum analysis time per item (seconds):

每个项目分析所使用的最大时间（单位：秒）

这些设置控制将包括可执行代码的静态分析的扫描类型，注意静态分析可以消耗大量的存储器和处理，因此可能希望将静态分析限制到感兴趣的关键目标