Netsparker® Help

Netsparker is the first false-positive free scanner. This document details the features of Netsparker, and shows you how to use and tweak them in order to get the best out of the product. If you can't find what you are looking for, please contact us (support@netsparker.com) for assistance.

Netsparker Ltd.

Finance House, 522 Uxbridge Rd.
Pinner. HA5 3PU / UK

+44 (0) 20 3411 8552

+44 (0) 20 3411 8553

Table of Contents

Introduction to Netsparker

Product Overview

Welcome to **Netsparker**, the next generation Web Application Security Scanner. Netsparker is a powerful web application security scanner, which can crawl, attack and identify vulnerabilities in all types of web application - whatever platform and technology it's built on. Netsparker can help you identify web application vulnerabilities such as **SQL Injection**, **Cross-site Scripting (XSS)**, **Remote Code Execution,** and many more with an easy-to-use and intuitive user interface.

Netsparker helps **web application developers** or **penetration testers** to secure web applications easily and with the minimum of fuss. With pre-set scan profiles, you can quickly start scanning your web applications and get instant, detailed reports on Netsparker's findings.

Feature List

False-Positive Free

Netsparker doesn't produce false positives, period.

All current web application security scanners report false-positives. That is, they report vulnerabilities that do not exist.

Netsparker is different; it will perform multiple tests to confirm any identified issues. If Netsparker can't confirm them, the issue will require manual inspection and verification – therefore Netsparker will inform you about a **potential issue** - generally prefixed as **[Possible]**.

This means that if Netsparker makes a positive confirmation, you can be sure that a real vulnerability has been found.

Netsparker confirms vulnerabilities by exploiting them in a safe manner. If a vulnerability is successfully exploited, **it can't be a false-positive.** Exploitation is carried out in a non-destructive way.

Please see False Positive Free Scanning on our website for more information about the technical details and overall technology used by Netsparker.

JavaScript / AJAX / Web 2.0 Support

Netsparker has a JavaScript engine which can parse, execute and analyse the output of JavaScript and VBScript used in web applications.

This allows Netsparker to successfully crawl and understand websites that use different AJAX frameworks, custom code or well-known frameworks such as jQuery.

SOAP Web Service Scanning Support

Netsparker parses WSDL (Web Services Definition Language) documents and creates SOAP (Simple Object Access Protocol) requests for each operation defined in the WSDL document. This allows Netsparker to attack web services successfully. You can either scan a single web service by entering its WSDL address or importing the WSDL file from disk. If you start a regular web site scan and Netsparker discovers WSDL documents on that site, Netsparker will automatically scan those web services too.

Detailed Issue Reporting

Netsparker reports vulnerabilities with the maximum available details to make the issue, and the impact, clear to the user.

For example, instead of simply reporting XSS (Cross-site Scripting), Netsparker will report one of the following issues:

- Reflective Cross-site Scripting
- Permanent Cross-site Scripting

- Cross-site Scripting via RFI (*Where the user can carry out an XSS attack via RFI but cannot execute code*)
- Cross-site Scripting via LFI (*Where it's possible to attack via LFI, however impact is limited. In this case Netsparker will try to identify Cross-site Scripting via this limited LFI vulnerability*)
- Limited Cross-site Scripting (*Attack only works on Internet Explorer*)

The same goes for many other types of vulnerability.

The impact and remediation of issues is also tailored based on these details. Therefore developers will know exactly what to do in order to correctly resolve the problem.

Automation

Netsparker provides a CLI (*Command Line Interface*) to help you to automate scans and integrate Netsparker into your automated scanning, reporting or development systems.

Logging

Netsparker supports logging of all HTTP Requests and responses, as well as all identified vulnerabilities and other scan-related data.

Reporting

Netsparker produces reports in several different formats:

- XML
- HTML
- PDF
- CSV

In addition, you can use Netsparker's Reporting API to generate custom reports. The Reporting API supports C# scripting, and Netsparker ships with a selection of sample report templates which you may use as models for your own custom reports.

DRM Free Licensing

Netsparker utilizes a user-friendly licensing system which also respects users' privacy. It's DRM free and you don't have to activate it every time you move your license. Also it doesn't require an internet connection to activate or work. It works instantly, without the need to login anywhere or get permission from us.

Integrated Exploitation Engine

Netsparker delivers the detection, confirmation and exploitation of vulnerabilities in a single integrated environment.

When Netsparker identifies a vulnerability, it will let you exploit the vulnerability, if possible, so that you can see the real impact of an attack.

Currently Netsparker supports:

- Exploitation of SQL Injection vulnerabilities
- Getting a reverse shell from SQL Injection vulnerabilities

- Exploitation of LFI (*Local File Inclusion*) vulnerabilities
- Downloading the source code of all crawled pages via LFI (*Local File Inclusion*)
- Downloading known OS files via LFI (*Local File Inclusion*)

Post-Exploitation

Netsparker is the only web application security scanner with an integrated exploitation engine. This gives Netsparker an edge, and allows it to carry out post-exploitation security checks.

Initially, this is limited to checks carried out after SQL Injections, however the number and scope of checks will be increased in future releases of Netsparker.

When Netsparker identifies an SQL Injection, it will check to determine if the database user has admin privileges. If the user has administrator privileges, Netsparker will report a new issue called "**Database User Has Admin Privileges**"

Authentication

Netsparker supports several authentication methods:

- Basic Authentication
- Form Authentication
  The user can configure form authentication for different websites.
- NTLM Authentication
- Digest Authentication

This allows you to test an application which requires any one of the listed authentication methods.

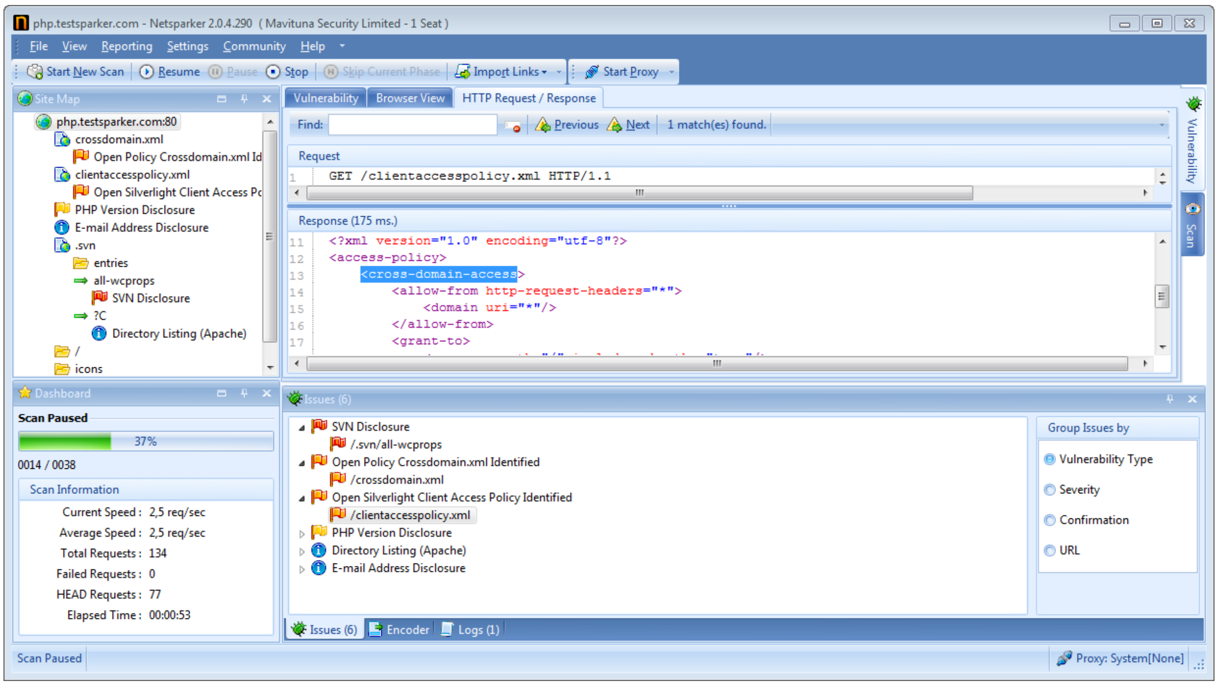
Knowledge Base

Netsparker reports informational items which can help the user to see overall design of the application such as:

- List of File Extensions
- List of E-mail Addresses
- List of Cookies
- List of Interesting Headers
- List of Pages With Inputs
- List of MIME Types
- List of JavaScript Files
- List of External Hosts
- List of External Scripts

Bug Tracking Integration

Netsparker can be integrated with external bug tracking systems and you can send the vulnerabilities to those systems using the **Send To** feature. Out of the box Netsparker has support for **FogBugz** and **JIRA** integration but it can be extended using the API.



1 match(es) found.

Request

```
GET /clientaccesspolicy.xml HTTP/1.1
```

Response (175 ms.)

```
<?xml version="1.0" encoding="utf-8"?>
<access-policy>
    <cross-domain-access>
        <allow-from http-request-headers="*">
            <domain uri="*"/>
        </allow-from>
        <grant-to>
```

```
php.testsparker.com:80
  crossdomain.xml
    Open Policy Crossdomain.xml Id
  clientaccesspolicy.xml
    Open Silverlight Client Access Pc
  PHP Version Disclosure
  E-mail Address Disclosure
  .svn
    entries
    all-wcprops
      SVN Disclosure
    ?C
      Directory Listing (Apache)
  /
  icons
```

Scan Paused

37%

0014 / 0038

Scan Information

```
   Current Speed :  2,5 req/sec
   Average Speed :  2,5 req/sec
  Total Requests :  134
 Failed Requests :  0
   HEAD Requests :  77
    Elapsed Time :  00:00:53
```

```
SVN Disclosure
  /.svn/all-wcprops
Open Policy Crossdomain.xml Identified
  /crossdomain.xml
Open Silverlight Client Access Policy Identified
  /clientaccesspolicy.xml
PHP Version Disclosure
Directory Listing (Apache)
E-mail Address Disclosure
```

Group Issues by

[✓ Vulnerability Type] Severity Confirmation URL

Scan Paused

Proxy: System[None]

Figure 1: A Sample View from the User Interface

Technical Details

When Netsparker identifies an SQL Injection, it can automatically determine how to exploit it and extract the version information from the application. When the version is successfully extracted, Netsparker will report the issue as confirmed, so that you can ensure that the issue is not a false-positive.

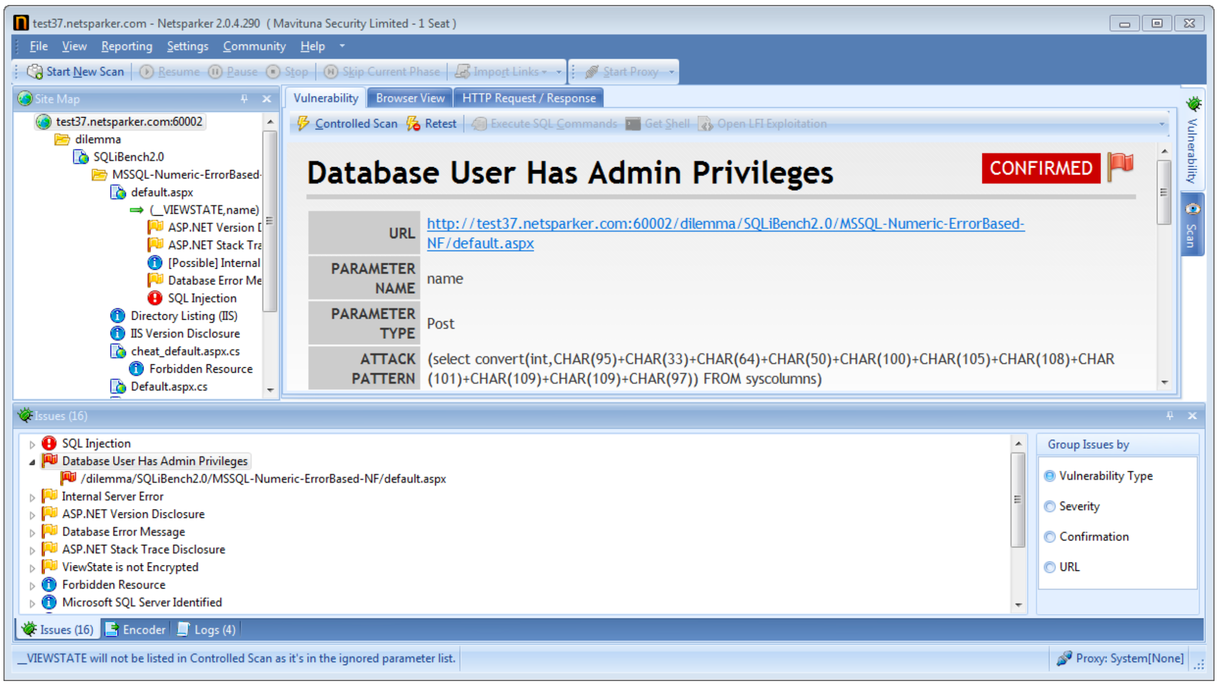
The same applies to other vulnerabilities such as XSS (*Cross-site Scripting*) where Netsparker loads the injection in an actual browser and observes the execution of JavaScript to confirm that the injection will actually get executed in the browser.

Below is a list of issues that Netsparker looks for.

SQL Injection

Netsparker can detect different SQL Injections including **Error Based**, **Blind** and **Time Based SQL** Injections. The SQL Injection engine is very comprehensive and can detect Blind SQL Injections even in complicated queries.

After identification of the vulnerability, Netsparker will carry out extra checks to determine if the database user used by the application has admin privileges. In this case Netsparker will report a separate issue called "Admin User DB Connection"


test37.netsparker.com - Netsparker 2.0.4.290 (Mavituna Security Limited - 1 Seat)
File View Reporting Settings Community Help
Start New Scan Resume Pause Stop Skip Current Phase Import Links Start Proxy
Site Map
Vulnerability Browser View HTTP Request / Response
Controlled Scan Retest Execute SQL Commands Get Shell Open LFI Exploitation
CONFIRMED
Vulnerability Scan
Group Issues by
Vulnerability Type
Severity
Confirmation
URL
Issues (16) Encoder Logs (4)

```
test37.netsparker.com:60002
  dilemma
    SQLiBench2.0
      MSSQL-Numeric-ErrorBased-
        default.aspx
          (__VIEWSTATE,name)
          ASP.NET Version D
          ASP.NET Stack Tra
          [Possible] Internal
          Database Error Me
          SQL Injection
        Directory Listing (IIS)
        IIS Version Disclosure
        cheat_default.aspx.cs
          Forbidden Resource
        Default.aspx.cs
```

Database User Has Admin Privileges

URL	http://test37.netsparker.com:60002/dilemma/SQLiBench2.0/MSSQL-Numeric-ErrorBased-NF/default.aspx
PARAMETER NAME	name
PARAMETER TYPE	Post
ATTACK PATTERN	(select convert(int,CHAR(95)+CHAR(33)+CHAR(64)+CHAR(50)+CHAR(100)+CHAR(105)+CHAR(108)+CHAR(101)+CHAR(109)+CHAR(109)+CHAR(97)) FROM syscolumns)

Issues (16)

```
SQL Injection
Database User Has Admin Privileges
  /dilemma/SQLiBench2.0/MSSQL-Numeric-ErrorBased-NF/default.aspx
Internal Server Error
ASP.NET Version Disclosure
Database Error Message
ASP.NET Stack Trace Disclosure
ViewState is not Encrypted
Forbidden Resource
Microsoft SQL Server Identified
```

__VIEWSTATE will not be listed in Controlled Scan as it's in the ignored parameter list.

Proxy: System[None]

Figure 2: Database User Has Administrator Rigths Example

XSS (Cross-site Scripting)

Netsparker identifies Permanent/Stored and Reflective Cross-site Scripting. Cross-site Scripting issues can be identified in parameters or in the URL.

Netsparker carries out several different attacks to bypass known and custom weak protection.

XSS (Cross-site Scripting) via Remote File Injection

Netsparker detects if it’s possible for an attacker to inject a remote file to the site in order to execute JavaScript in the current page.

This is a typical technique used by attackers to carry out Cross-site Scripting attacks.

XSS (Cross-site Scripting) in URLs

Netsparker also detects Cross-site Scripting issues in URLs. This type of attack is common in websites using URL Rewrite and PHP.

DOM based XSS (Cross-site Scripting)

Netsparker detects DOM based Cross-site Scripting issues. DOM based XSS is an XSS attack wherein the attack payload is executed as a result of modifying the DOM environment in the victim's browser used by the original client side script, so that the client side code runs in an unexpected manner. The response from the server does not change, but the client side code contained in the page executes differently due to the malicious modifications that have occurred in the DOM environment.

Command Injection

Netsparker detects pages that are susceptible to Command Injection, whereby input data is interpreted as an operating system command.

This type of vulnerability can allow an attacker to gain full access over the server and the web application.

Blind Command Injection

Netsparker detects pages that are susceptible to Blind Command Injection, whereby input data is interpreted as an operating system command but it can’t be directly identified from the output of the page. Netsparker will identify Blind Command Injection vulnerabilities by carrying out several requests and analyzing the time differences.

This type of vulnerability can allow an attacker to gain full access over the server and the web application.

E-mail Address Disclosure

Netsparker identifies email addresses exposed in the website. This can help users to identify email address-related information exposed on the internet to help you reduce the potential for spam.

Internal IP Disclosure

Netsparker identifies internal IP Disclosure issues where a system exposes its internal network IP address.

Cookies are not marked as Secure

Netsparker reports an issue if it finds cookies that are not marked as “Secure” in HTTPS websites.

Not marking cookies as “Secure” can allow attackers to steal the cookies over an HTTP connection and use those cookies to log in to the application.

Cookies are not marked as HTTPOnly

Netsparker reports an issue if it finds cookies that are not marked as HTTPOnly.

JavaScript can't read a cookie if the cookie is marked as "HTTPOnly", which means that a Cross-site Scripting attack is unable to steal the cookies via JavaScript. However that doesn't mean the application is secure. Cross-site Scripting vulnerabilities should be addressed even though cookies are marked as "HTTPOnly" since there are many other ways to use Cross-site Scripting attacks.

"HTTPOnly" should be considered as a secondary security measure (defense in depth) and should be used where possible.

Directory Listing

Netsparker detects if directory listing is enabled in the web server.

Directory listing can allow attackers to see all files on the system and can help them to download sensitive files or gain more information about the target system.

Stack Trace Disclosure

Netsparker determines if the target application is disclosing stack trace information.

A stack trace can leak information about the internals of the application that might include sensitive data or application logic-related clues.

Version Disclosure

Netsparker identifies version disclosures in HTTP headers and HTTP responses. It supports many frameworks, well known languages and web servers such as ORACLE, IIS, PHP, ASP.NET, Apache, Apache Modules, JSP, Mongrel, WebLogic etc.

Access Denied Resources

Netsparker reports an information issue when access is denied to the requested resources.

This can help the user to determine the design of the application and possible resources that exist in the web server but are not publicly available.

Internal Path Disclosure

Netsparker determines if an application discloses internal paths related to the application or the configuration.

This generally indicates a programming error in the application and can help an attacker to gain more information about the internals of the system. An attacker can use this information while crafting an exploit for another identified vulnerability.

Programming Error Messages

Netsparker provokes the target website to produce error messages and then reports on the outcome. These errors have no direct security impact; most of the time they indicate a programming error, quality issue, or a potential vulnerability in the application.

Many of these types of errors also leak information about the logic or the implementation of the application which can help an attacker to identify or exploit weaknesses in the application.

Database Identified

Netsparker detects the disclosure of the backend database type, from a list of supported database engines, including SQL Server, MySQL, PostgreSQL, DB2 and Oracle.

Database User Has Admin Privileges

Netsparker detects if the web application is accessing its backend database via a user account with administrative privileges. This vulnerability increases the potential exposure in the event of an SQL Injection attack and introduces a range of serious consequential issues, including:

- Allowing full and unrestricted access to the database server
- The ability to obtain a reverse shell and execute commands on the underlying OS
- The possibility of escalating privileges and gaining administrator access to the OS

Database Error Messages

Netsparker provokes and reports database error messages leaked by the website. If the problem is related to SQL Injection, a separate issue will be raised by Netsparker, otherwise the user is informed that the application is exposing database error messages which are potentially related to programming errors or possibly other problems regarding database connectivity.

Local File Inclusions & Arbitrary File Reading

Netsparker detects **Local File Inclusion** and **Arbitrary File Reading** issues. It detects if an attacker can access files and source code from the server on both Windows and *nix systems. It carries out advanced checks, uses process directories, Null byte injection attacks, dynamic file extension replacements and many other methods to bypass weak filters and blacklist entries.

In addition, it checks if the Local File Inclusion can be used for executing remote commands by injecting code into log files.

Netsparker has exploitation features for Local File Inclusion attacks.

Remote File Inclusions

Netsparker detects if the application is vulnerable to Remote File Inclusion, which would allow an attacker to inject a remote file and execute code on the server.

Netsparker carries out several dynamic requests, and tries to bypass many weaknesses and blacklist entries.

Remote Code Injection / Evaluation

Netsparker detects if the application evaluates/executes given code within itself by using dangerous calls such as eval().

This type of vulnerability can allow an attacker to execute code on the server.

CRLF / HTTP Header Injection / Response Splitting

Netsparker detects CRLF injection issues in web applications which can cause serious problems. The most common of these are Cross-site Scripting and session hijacking attacks, taking the form of session fixation attacks.

Find Backup Files

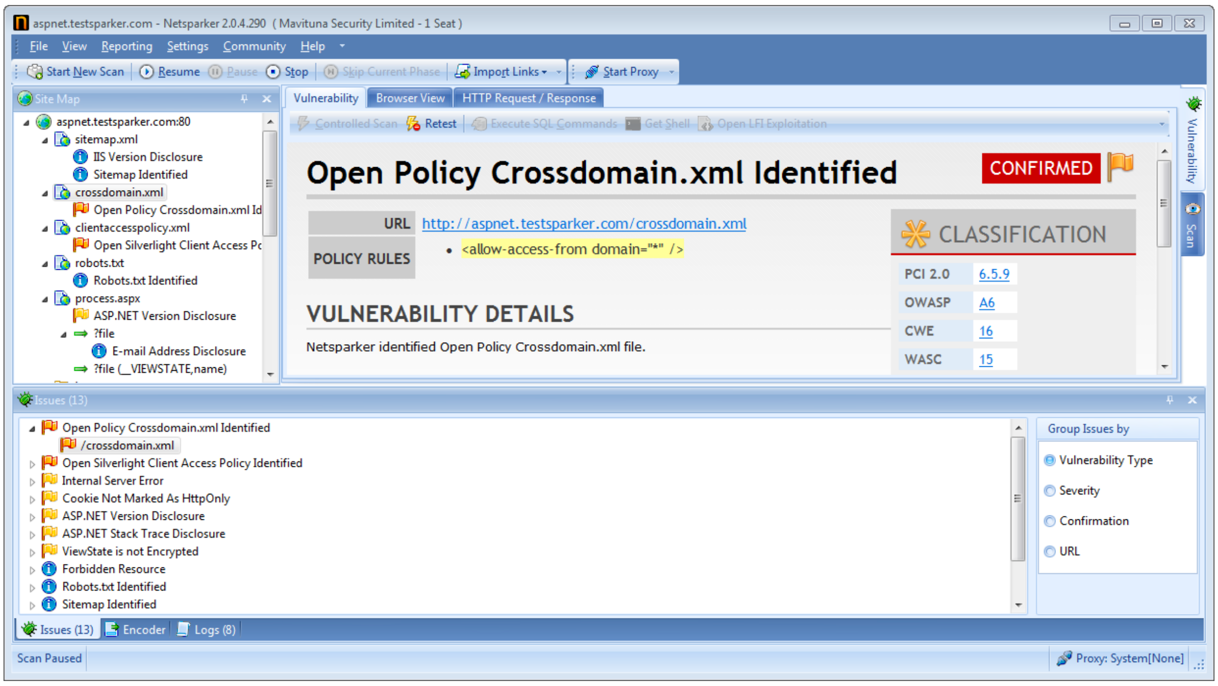
Netsparker tries to find backup and temporary files on the target website by using crawled file names and other well-known names.

Netsparker determines if this problem can lead to source code disclosure issues.

crossdomain.xml Analysis

Netsparker detects and analyses crossdomain.xml files for problems such as open access policy issues.

With this type of vulnerability, an attacker needs to attack an authenticated user of the website to exploit this problem successfully. The attacker is then able to read authenticated users' private messages or carry out actions as the attacked user. If the Crossdomain.xml file has an open policy, the attacker can bypass any CSRF protection (*nonce / CSRF tokens*).


aspnet.testsparker.com - Netsparker 2.0.4.290 (Mavituna Security Limited - 1 Seat)
File View Reporting Settings Community Help
Start New Scan Resume Pause Stop Skip Current Phase Import Links Start Proxy
Site Map
Vulnerability Browser View HTTP Request / Response
Controlled Scan Retest Execute SQL Commands Get Shell Open LFI Exploitation
Issues (13) Encoder Logs (8)
Group Issues by
Vulnerability Type
Severity
Confirmation
URL
Scan Paused
Proxy: System[None]

aspnet.testsparker.com:80
- sitemap.xml
  - IIS Version Disclosure
  - Sitemap Identified
- crossdomain.xml
  - Open Policy Crossdomain.xml Id
- clientaccesspolicy.xml
  - Open Silverlight Client Access Pc
- robots.txt
  - Robots.txt Identified
- process.aspx
  - ASP.NET Version Disclosure
  - ?file
    - E-mail Address Disclosure
  - ?file (__VIEWSTATE,name)

Open Policy Crossdomain.xml Identified CONFIRMED

URL	http://aspnet.testsparker.com/crossdomain.xml
POLICY RULES	• <allow-access-from domain="*" />

CLASSIFICATION

PCI 2.0	6.5.9
OWASP	A6
CWE	16
WASC	15

VULNERABILITY DETAILS

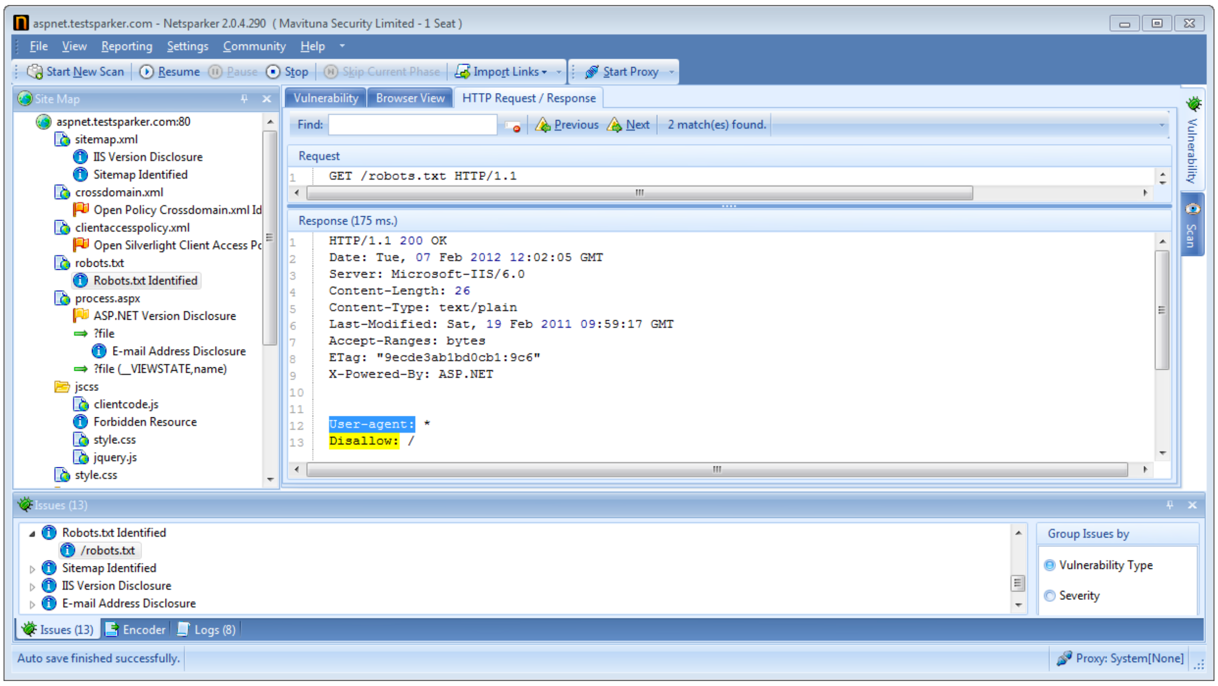
Netsparker identified Open Policy Crossdomain.xml file.

Issues (13)
- Open Policy Crossdomain.xml Identified
  - /crossdomain.xml
- Open Silverlight Client Access Policy Identified
- Internal Server Error
- Cookie Not Marked As HttpOnly
- ASP.NET Version Disclosure
- ASP.NET Stack Trace Disclosure
- ViewState is not Encrypted
- Forbidden Resource
- Robots.txt Identified
- Sitemap Identified

Figure 3: Open Policy Crossdomain.xml Identified

Finds and Analyses Potential Issues in Robots.txt

Netsparker detects and parses links in Robots.txt files. If it identifies a potentially critical URL listed in the Robots.txt it will report the problem, together with details.


aspnet.testsparker.com - Netsparker 2.0.4.290 (Mavituna Security Limited - 1 Seat)
File View Reporting Settings Community Help
Start New Scan Resume Pause Stop Skip Current Phase Import Links Start Proxy
Site Map
aspnet.testsparker.com:80
sitemap.xml
IIS Version Disclosure
Sitemap Identified
crossdomain.xml
Open Policy Crossdomain.xml Id
clientaccesspolicy.xml
Open Silverlight Client Access Po
robots.txt
Robots.txt Identified
process.aspx
ASP.NET Version Disclosure
?file
E-mail Address Disclosure
?file (__VIEWSTATE,name)
jscss
clientcode.js
Forbidden Resource
style.css
jquery.js
style.css
Vulnerability Browser View HTTP Request / Response
Find: Previous Next 2 match(es) found.
Request
Response (175 ms.)
Issues (13)
Robots.txt Identified
/robots.txt
Sitemap Identified
IIS Version Disclosure
E-mail Address Disclosure
Group Issues by
Vulnerability Type
Severity
Issues (13) Encoder Logs (8)
Auto save finished successfully.
Proxy: System[None]
Vulnerability
Scan

```
GET /robots.txt HTTP/1.1
```

```
HTTP/1.1 200 OK
Date: Tue, 07 Feb 2012 12:02:05 GMT
Server: Microsoft-IIS/6.0
Content-Length: 26
Content-Type: text/plain
Last-Modified: Sat, 19 Feb 2011 09:59:17 GMT
Accept-Ranges: bytes
ETag: "9ecde3ab1bd0cb1:9c6"
X-Powered-By: ASP.NET


User-agent: *
Disallow: /
```

Figure 4: Robots File Identified

Password Transmitted over HTTP

Netsparker identifies if the website sends passwords over HTTP.

An attacker sitting between the user and the website might carry out a MITM (*Man in the middle*) or sniffing attack to capture the user's password.

Password Transmitted over Query String

Netsparker identifies if the website accepts passwords transmitted in a request query string.

An attacker sitting between the user and the website might carry out a MITM (*Man in the middle*) or sniffing attack to capture the user's password.

Password Form Served over HTTP

Netsparker determines if a login form is served over HTTP when the target of the form is HTTPS.

Many developers might not be aware that this is a security issue; therefore Netsparker provides a detailed report for this problem to ensure that the issue is correctly addressed by developers.

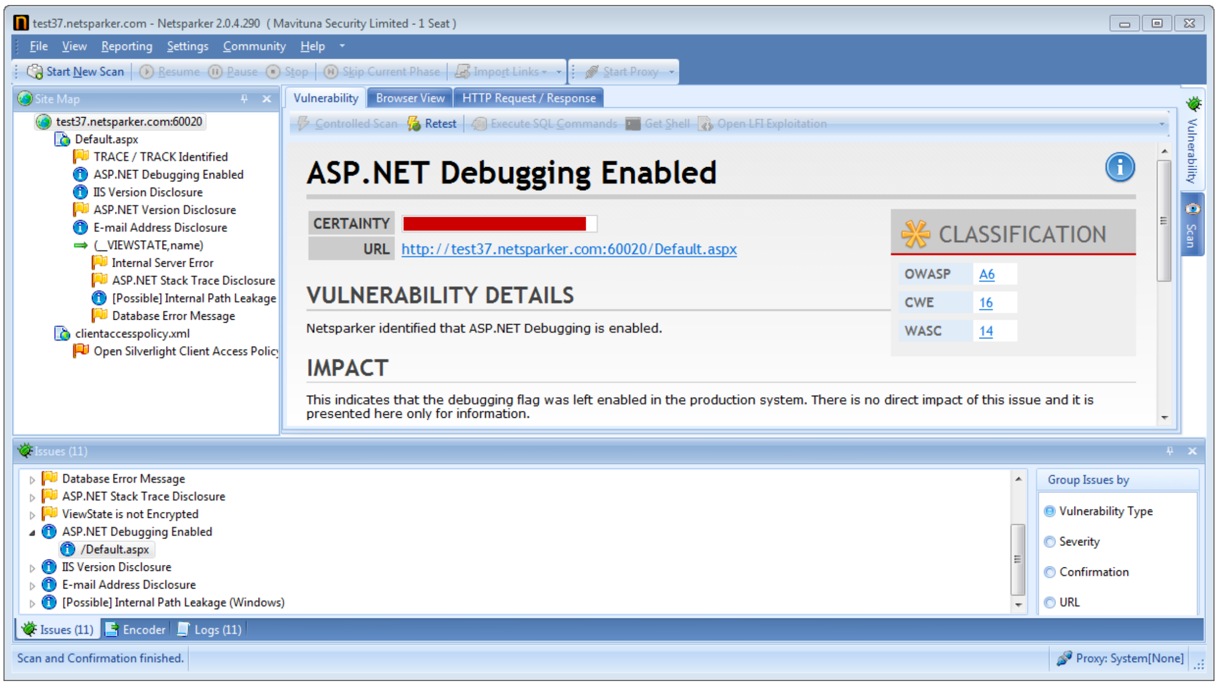
An attacker sitting between the user and the website might carry out a MITM (*Man in the middle*) and inject a piece of JavaScript code to steal the password before it reaches HTTPS, or he/she can easily change the target of the form to HTTP as well to steal the user's password.

Detect TRACE / TRACK Method Support

Netsparker checks and determines if the TRACE / TRACK HTTP Methods are supported and enabled by the web server.

Detect ASP.NET Debugging

Netsparker detects if ASP.NET Debugging is enabled.



test37.netsparker.com:60020
- Default.aspx
  - TRACE / TRACK Identified
  - ASP.NET Debugging Enabled
  - IIS Version Disclosure
  - ASP.NET Version Disclosure
  - E-mail Address Disclosure
  - (__VIEWSTATE,name)
    - Internal Server Error
    - ASP.NET Stack Trace Disclosure
    - [Possible] Internal Path Leakage
    - Database Error Message
- clientaccesspolicy.xml
  - Open Silverlight Client Access Polic

ASP.NET Debugging Enabled

CERTAINTY	
URL	http://test37.netsparker.com:60020/Default.aspx

CLASSIFICATION

OWASP	A6
CWE	16
WASC	14

VULNERABILITY DETAILS

Netsparker identified that ASP.NET Debugging is enabled.

IMPACT

This indicates that the debugging flag was left enabled in the production system. There is no direct impact of this issue and it is presented here only for information.

Issues (11)
- Database Error Message
- ASP.NET Stack Trace Disclosure
- ViewState is not Encrypted
- ASP.NET Debugging Enabled
  - /Default.aspx
- IIS Version Disclosure
- E-mail Address Disclosure
- [Possible] Internal Path Leakage (Windows)

Group Issues by
- [✓ Vulnerability Type]
- Severity
- Confirmation
- URL

Scan and Confirmation finished.

Proxy: System[None]

Figure 5: ASP.NET Debugging Enabled

Detect ASP.NET Trace

Netsparker detects if ASP.NET Tracing is enabled and accessible.

An attacker can use ASP.NET Tracing output to access active users' sessions, and gather information about the application and its structure.

Checks for CVS, GIT and SVN Information and Source Code Disclosure Issues

Netsparker detects files disclosed by source code versioning systems such as CVS, GIT and SVN.

An attacker might exploit this problem to gain access to the source code of the application, or might retrieve configuration and/or other important files.

Finds PHPInfo() pages and PHPInfo() disclosure in other pages

Netsparker attempts to find forgotten phpinfo files in the system. It also reports the PHPinfo() output in all crawled pages.

Information disclosed from PHPInfo() might help attackers gain more information about the target system.

MS Office Information Disclosure

Netsparker detects if pages served by the web application were generated using Microsoft Office. Such pages contain embedded user information and can be used for social engineering attacks.

Finds Apache Server-Status and Apache Server-Info pages

Netsparker detects if the Apache **Server-Status** or **Server-Info** pages are publicly accessible.

Apache Server-Status and Server-Info can be used by attackers to gain more information about the target system and will help them to find hidden URLs and currently visited URLs.

Find Hidden Resources

Netsparker looks for hidden files and directories in the target website.

These include:

- Test files
- Management files and directories
- Known vulnerable files / scripts

For example, even if it's not linked anywhere in the website, Netsparker will identify the "admin" directory.

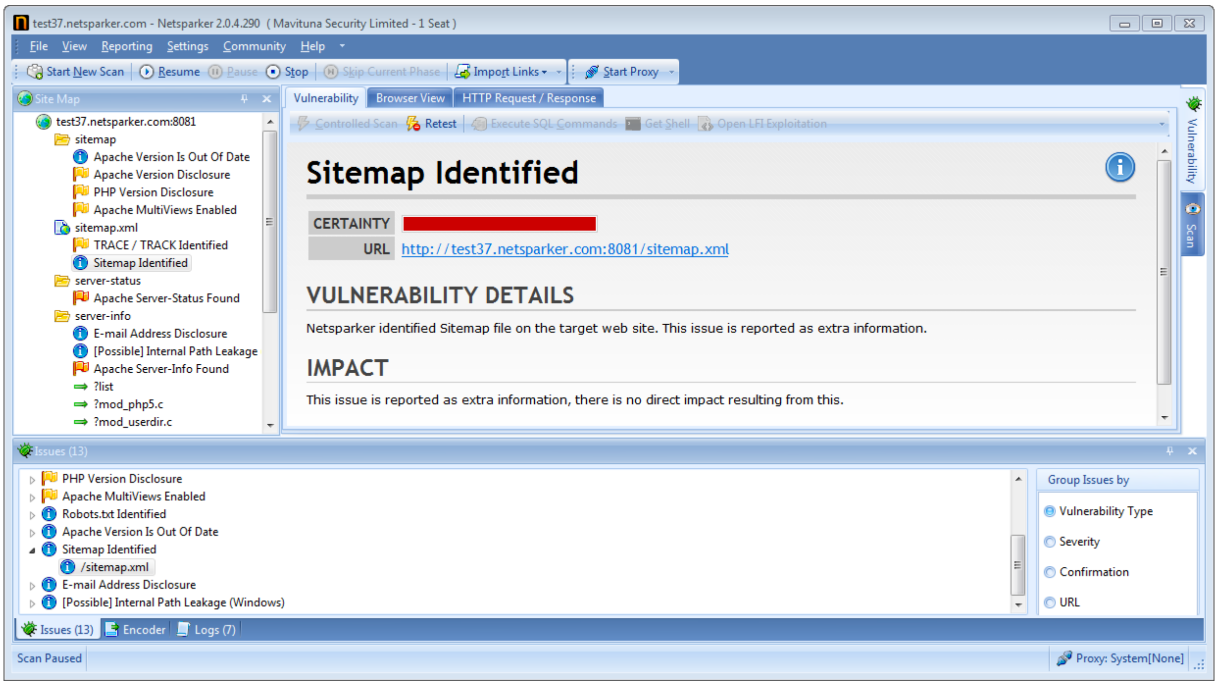
Insecure Authentication Scheme Used Over HTTP

Netsparker detects if the application is configured to use Basic, Digest or NTLM authentication over HTTP. These authentication schemes are considered to be sufficiently secure if they are used over HTTPS. Using them over HTTP can result in a variety of consequential issues, including:

- Information leakage
- The possibility to lock or brute force user accounts
- Transmission of user credentials on a clear-text form

Finds and Analyses Google Sitemap Files

Netsparker detects and parses Google Sitemap files to increase coverage and inform the user that the sitemap file is accessible (in order to confirm that this is the intended configuration).



test37.netsparker.com - Netsparker 2.0.4.290 (Mavituna Security Limited - 1 Seat)
File View Reporting Settings Community Help
Start New Scan Resume Pause Stop Skip Current Phase Import Links Start Proxy
Site Map
Vulnerability Browser View HTTP Request / Response
Controlled Scan Retest Execute SQL Commands Get Shell Open LFI Exploitation
test37.netsparker.com:8081
sitemap
Apache Version Is Out Of Date
Apache Version Disclosure
PHP Version Disclosure
Apache MultiViews Enabled
sitemap.xml
TRACE / TRACK Identified
Sitemap Identified
server-status
Apache Server-Status Found
server-info
E-mail Address Disclosure
[Possible] Internal Path Leakage
Apache Server-Info Found
?list
?mod_php5.c
?mod_userdir.c
Issues (13)
PHP Version Disclosure
Apache MultiViews Enabled
Robots.txt Identified
Apache Version Is Out Of Date
Sitemap Identified
/sitemap.xml
E-mail Address Disclosure
[Possible] Internal Path Leakage (Windows)
Group Issues by
Vulnerability Type
Severity
Confirmation
URL
Issues (13) Encoder Logs (7)
Scan Paused
Proxy: System[None]
Vulnerability
Scan

Sitemap Identified

CERTAINTY

URL http://test37.netsparker.com:8081/sitemap.xml

VULNERABILITY DETAILS

Netsparker identified Sitemap file on the target web site. This issue is reported as extra information.

IMPACT

This issue is reported as extra information, there is no direct impact resulting from this.

Figure 6: Sitemap File Identified

Source Code Disclosure

Netsparker provokes the web server to disclose source code where possible, and detects whether the source code disclosure is due to a configuration problem, a security issue, or due to a programming error where revealing comments are left in the code.

An attacker can access hard coded passwords and might gain information about the logic of the application (and the system) by reading the disclosed source code.

Autocomplete Enabled

Netsparker determines if Autocomplete is left Enabled in sensitive form fields such as **Credit Card** number fields.

An attacker who can access the user’s computer can access these cached autocomplete datasets via the browser. This is especially critical if the website is accessed from public computers.

ASP.NET ViewState Analysis

Netsparker analyses ViewState related issues in ASP.NET pages.

ViewState is not Signed

Netsparker reports a new issue if the ViewState in the page is not signed. In this case an attacker might modify the content of the ViewState and subvert the logic of the application, or carry out other attacks by changing the ViewState.

Strict Scope Options

Netsparker supports several scope restrictions before the scan is started. You can also exclude pages / parameters and paths from the scan before Netsparker starts to attack.

Heuristic URL Rewrite Detection

Netsparker can heuristically detect common URL Rewrite patterns to avoid repeatedly scanning the same resources. The Netsparker development team is constantly improving this engine to understand more and more URL Rewrite patterns.

File Upload Functionality Identified

Netsparker detects pages that allow users to upload files to the web server, which are potentially dangerous unless they are coded with a great deal of care.

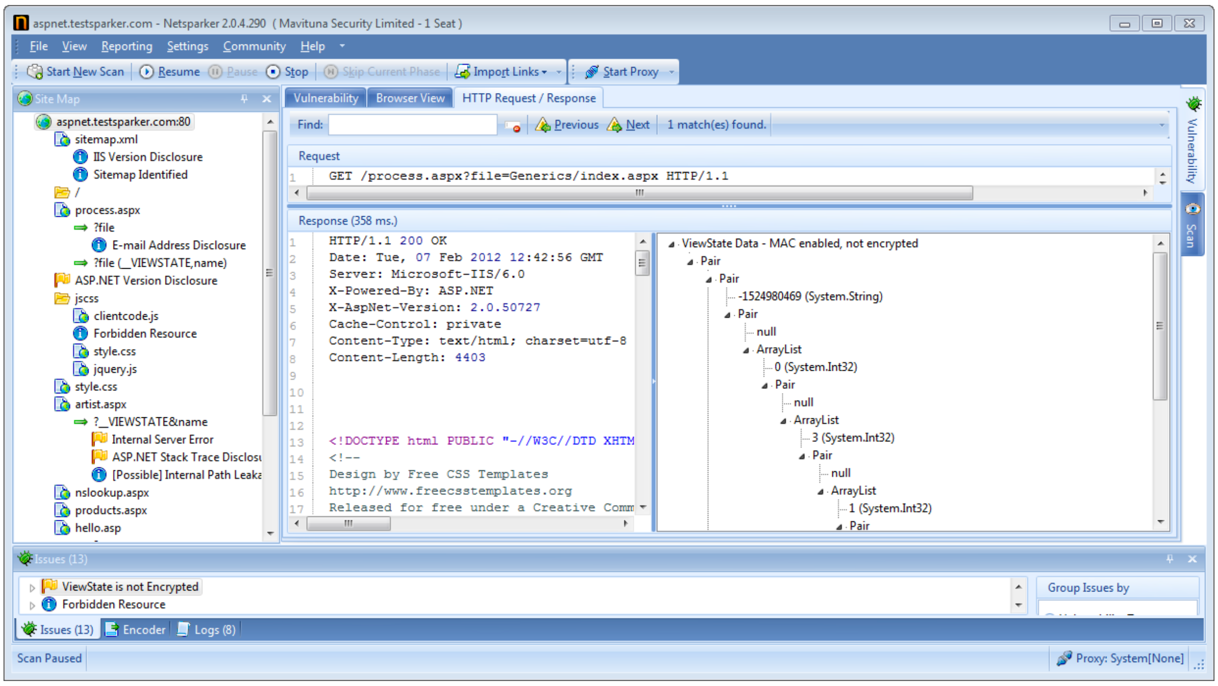
Manual Proxy Mode

You can use Netsparker's internal proxy to browse your target web application to allow Netsparker to crawl and test hidden sections of the application or parts accessed by third party technologies such as ActiveX, Java and Flash.

This feature also allows you to test forms which require special input such as credit card or social security numbers.

ViewState is not Encrypted

Netsparker reports a vulnerability if the ViewState in the page is not encrypted. In this case an attacker can read the data within the ViewState by simply decoding it. This might leak sensitive information to the attacker.


aspnet.testsparker.com - Netsparker 2.0.4.290 (Mavituna Security Limited - 1 Seat)
File View Reporting Settings Community Help
Start New Scan Resume Pause Stop Skip Current Phase Import Links Start Proxy
Site Map
aspnet.testsparker.com:80
sitemap.xml
IIS Version Disclosure
Sitemap Identified
/
process.aspx
?file
E-mail Address Disclosure
?file (__VIEWSTATE,name)
ASP.NET Version Disclosure
jscss
clientcode.js
Forbidden Resource
style.css
jquery.js
style.css
artist.aspx
?__VIEWSTATE&name
Internal Server Error
ASP.NET Stack Trace Disclosu
[Possible] Internal Path Leaka
nslookup.aspx
products.aspx
hello.asp
Vulnerability Browser View HTTP Request / Response
Find: Previous Next 1 match(es) found.
Request
Response (358 ms.)
Issues (13)
ViewState is not Encrypted
Forbidden Resource
Group Issues by
Issues (13) Encoder Logs (8)
Scan Paused
Proxy: System[None]
Vulnerability
Scan

```
GET /process.aspx?file=Generics/index.aspx HTTP/1.1
```

```
HTTP/1.1 200 OK
Date: Tue, 07 Feb 2012 12:42:56 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
Cache-Control: private
Content-Type: text/html; charset=utf-8
Content-Length: 4403




<!DOCTYPE html PUBLIC "-//W3C//DTD XHTM
<!--
Design by Free CSS Templates
http://www.freecsstemplates.org
Released for free under a Creative Comm
```

```
ViewState Data - MAC enabled, not encrypted
  Pair
    Pair
      -1524980469 (System.String)
      Pair
        null
        ArrayList
          0 (System.Int32)
          Pair
            null
            ArrayList
              3 (System.Int32)
              Pair
                null
                ArrayList
                  1 (System.Int32)
                  Pair
```

Figure 7: ASP.NET ViewState Analysis

Custom 404 Detection

Netsparker automatically detects custom 404 and Error pages. You don't have to configure this manually.

Authentication Brute Forcing

Netsparker maintains a user-configurable word list and uses it to attempt brute force bypassing of Basic, NTLM and Digest Authentication schemes, reporting a corresponding vulnerability when a brute force attempt succeeds.

Basic Authentication over Clear Text

Netsparker detects if the application is using Basic Authentication over HTTP, which sends user credentials in plain text and exposes the risk that an attacker can intercept network traffic and steal them.

Anti-CSRF Token Support

Many web applications incorporate protection mechanisms to guard against CSRF (Cross-site Request Forgery). However, most other web application security scanners are unable to successfully scan pages that use these mechanisms, rendering them ineffective at security auditing such sites.

Netsparker addresses this challenge by requesting a new token from the application when required and using that token in the following request(s), enabling it to offer the only complete security scanning solution for this scenario.

Sensitive Files

Netsparker maintains a target list of known file names that are used by a variety of web applications and are susceptible to attack. It systematically probes for these files and, if found, attempts to attack their known vulnerabilities.

Silverlight Client Access Policy Identified

Netsparker detects the presence of the Silverlight Open Policy file (clientaccesspolicy.xml), which allows other Silverlight client services to make HTTP requests to the target server. It could potentially be used for accessing one time tokens and CSRF nonces to bypass CSRF restrictions.

Web Backdoor

Netsparker detects the presence of web backdoors on the target web server. This may indicate that the web server has been successfully attacked on a previous occasion.

Redirect Response BODY Is Too Large

Netsparker detects HTTP redirect responses that also contain body content. This generally indicates that, after redirection, the server did not abort generation of the redirecting page in the intended manner. This can lead to authentication bypass if the redirection mechanism is being used to restrict access to a private page that requires authentication.

Redirect Response BODY Has Two Responses

Netsparker detects HTTP redirects that include two responses. This generally indicates that, after redirection, the server did not abort generation of the redirecting page in the intended manner. This can lead to authentication bypass if the redirection mechanism is being used to restrict access to a private page that requires authentication.

Vulnerability Database

When Netsparker detects version disclosure in a web application or infrastructure component, it uses the disclosed version information to reference an embedded database of known vulnerabilities, to identify all security vulnerabilities that were known to be present in the identified version. This provides a rich insight into the vulnerabilities present in supported web applications, including known vulnerabilities that cannot ordinarily be detected or confirmed by automated scanning.

Netsparker's vulnerability database is maintained and developed on an ongoing basis. The applications currently represented in the database include Apache, Tomcat, SQL Server and MySQL.

Weak Credentials

Netsparker detects the use of weak usernames and passwords to access web resources. Depending on the nature of the password-protected resource, an attacker might exploit this to access the contents of the resource or to access password protected administrative mechanisms, potentially allowing full control of the application.

Frame Injection
Netsparker detects the use of Frame Injection on a web page. Frame Injection occurs when a frame on a vulnerable web page displays another web page via a user controllable input. An attacker might exploit this vulnerability to redirect users to other malicious web sites which are used for phishing and similar attacks.

Default Page Identified
Netsparker detects the presence of the default installation page on Apache, IIS 6, IIS 7 and several other systems. This issue is reported for information only.

XSS Protection Disabled
Netsparker detects if a web application issues an HTTP request that disables Internet Explorer's built-in XSS Protection feature.

Open Redirection
Netsparker detects if a web page is being redirected to another web page via a user controllable input. An attacker might exploit this vulnerability to redirect users to other malicious web sites which are used for phishing and similar attacks.

MySQL Username Disclosure
Netsparker detects if an application error message contains a MySQL user name. This information might be used as part of an attack on the database server.

Expression Language Injection
Netsparker detects if input data is evaluated by an expression language interpreter. This might be exploited by an attacker to gain the control of the underlying system.

Ruby on Rails Remote Code Execution
Netsparker checks if remote code execution occurs in the XML or JSON request processor of the Ruby on Rails application framework.

WebDAV Security Checks
Netsparker checks if it is possible to add / change content of the website via WebDAV extension and it leads to a code execution for specific file extensions.

Web App Fingerprinting
Netsparker identifies if the application is a well-known of-the-shelf product that includes known vulnerabilities. Netsparker can detect web applications such as WordPress, Joomla!, Drupal, MediaWiki, phpBB.

GWT Scanning Support

Netsparker can crawl and scan web applications built on the GWT (Google Web Toolkit) platform. Netsparker detects GWT RPC requests and attacks for all types of vulnerabilities, including Cross-Site Scripting and SQL Injection. You can view GWT RPC requests' details in knowledge base section.

Installing Netsparker

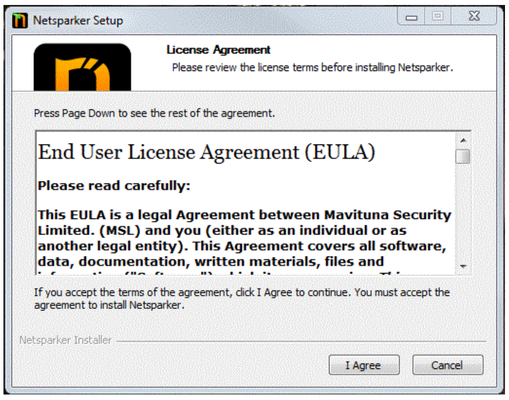
System Requirements

- Microsoft Windows XP Professional Edition Service Pack 3 or Windows Server 2003 Service Pack 2 or higher (Microsoft Windows 7 x64 recommended)
- Microsoft Internet Explorer 7 or higher (Internet Explorer 9 recommended)
- Microsoft .NET Framework 4.0 runtime
- 1Ghz Pentium processor or higher (Intel i7 1.6GHz or higher recommended)
- 1GB of available RAM (minimum) (4GB recommended)
- 100 MB of HDD space for installation and an additional 100 MB for scanning, with up to 4.2 GB required per scan, depending on target application size
- CD or DVD drive is **not** required

Installation Instructions

Please ensure that you have the latest service packs and Windows updates on your computer.

Download the latest version of Netsparker from the download location provided, and save the setup file to your computer. When the download is complete, run **NetsparkerSetup.exe** to start the installation wizard.



Netsparker Setup

License Agreement
Please review the license terms before installing Netsparker.

Press Page Down to see the rest of the agreement.

End User License Agreement (EULA)

Please read carefully:

This EULA is a legal Agreement between Mavituna Security Limited. (MSL) and you (either as an individual or as another legal entity). This Agreement covers all software, data, documentation, written materials, files and

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Netsparker.

Figure 8: Installer Window

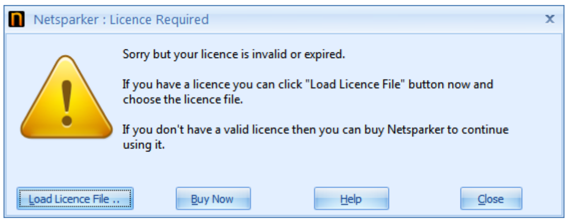
The first page of the wizard is the License Agreement. Review the License Agreement and click I Agree to continue.

Netsparker Setup will then provide a default installation folder; if you need to change the destination folder, click **Browse...** to change it. When you are happy with the installation location, press **Continue** to proceed.

You will then be prompted to choose the Start Menu folder. You can change this if you prefer, then press **Install** to start installation.

Wait a few minutes for the installation to complete.

When the installation is complete, Netsparker will start and ask for the license file. Click **Load License File...** and locate the license file. This will copy your license file into the program folder and load the main user interface.



Sorry but your licence is invalid or expired.

If you have a licence you can click "Load Licence File" button now and choose the licence file.

If you don't have a valid licence then you can buy Netsparker to continue using it.

Figure 9: License Question Window

Updating and Automatic Updates

Netsparker can automatically update itself over the Internet when a newer version is available. If you decide not to enable this feature, you can update manually at any time by selecting **Check for Updates** on the **Help** menu, or by using the **Ctrl+U** keyboard shortcut.

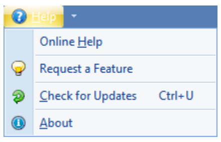


Figure 10: Check for Updates Menu

When automatic updates are enabled, Netsparker connects to the update server to check for available updates once a day. When an update is available, Netsparker will prompt you to confirm the download and installation of the update. After clicking **Download & Install** the update will be downloaded and applied by restarting the application. You may postpone the update by clicking **Remind Me Later** and Netsparker will remind you to update one week later.

You can enable or disable Automatic Updates at any time by using **Tools > Options > Auto Updates** menu.

Getting Started

Starting a new scan

To start a new scan with Netsparker, click Start New Scan from the **File** menu or from the **toolbar**. Type the address (URL) of the web application to be scanned in the Target URL box in the opened dialog window, and click the Start button.

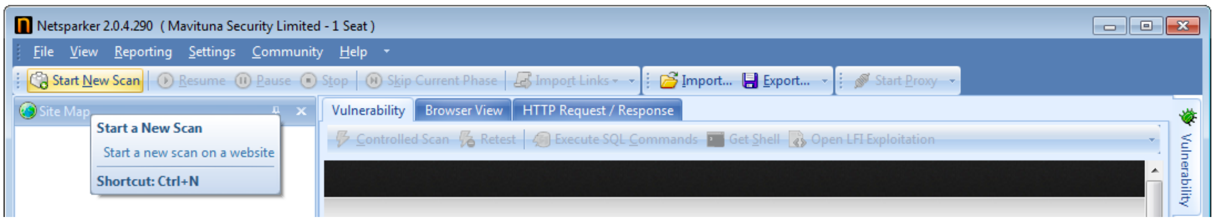


Figure 11: Start a New Scan Button

After typing the URL of the web application, you can customize the scan by clicking the Profiles button and selecting one of the preset scan profiles. When you have finished configuring these options, start the scan by clicking the Start Scan button. You can find detailed information on customization options in the Start New Scan dialog window in the "Customizing Scan Profile" section

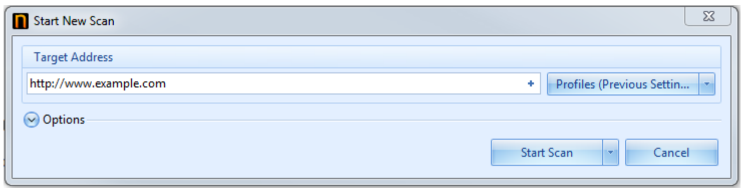


Figure 12: Start New Scan

The scan time will vary depending on the size of the scanned web application, the performance of the server on which it is running, and also the selected scan profile. During the scan, you can find detailed information about the scan progress on the Dashboard, which allows you to track issues identified during the scan in the Issues and Sitemap panels.

Scan Modes

Netsparker 1.5.0.0 introduces new scan modes. These new scan modes will help you to control the scan in a more granular way:

In total there are 4 scan modes:

- **Normal Scan**
  This is the classic scan mode. Netsparker will start to crawl the target web application, will find new links, and attack all identified pages without waiting.
  - Automatically find and follow new links

  - **Do not** wait after crawling phase
- **Crawl & Wait**
  This is very similar to the **Normal Scan** mode however, after the crawling phase, Netsparker will pause. This is useful if you want to **exclude** any identified pages from the scan **before** attacking.
  - Automatically **find** and follow new links
  - **Wait** after crawling phase

- **Manual Crawl (Proxy Mode)**
  This mode is useful if you need to test only certain files or if Netsparker can't be configured to access certain pages. Refer to the Manual Crawl (Proxy Mode) chapter for more information about this mode.
  - **Do not** Automatically find and follow new links
  - **Wait** after crawling phase
  - Start internal proxy automatically

- **Scan Imported Links Only**
  If you only want to test certain parts of the application you can either manually enter these URLs and HTTP Requests to test, or import them from a proxy log file. When you choose the **Scan Imported Links Only** scan mode, Netsparker will not find new links and will not try to access the rest of the application. It will only attack the requests / links you have imported.
  - **Do not** Automatically find and follow new links
  - **Do not** Wait after crawling phase

Import / Enter Links and HTTP Requests

If you need to feed Netsparker with custom Links or HTTP Requests you can do this by importing Proxy logs or manually entering URLs and HTTP Requests from the user interface.

This is especially useful if you want to:

- Add a directory to the scan which is not possible to find unless the user gives it to Netsparker (*i.e. http://example.com/hidden-admin-directory-3344/*)
- Add a new parameter in a URL to test, where the parameter is not linked from anywhere
- If the application is using a third party component such as Flash, Java Applet, or ActiveX then you can capture their request via a Proxy and import and feed Netsparker's crawler with them

Also you can use Netsparker's Proxy feature and Manual Crawl features to accomplish the same task in a different way, please refer to Manual Crawl (Proxy Mode).

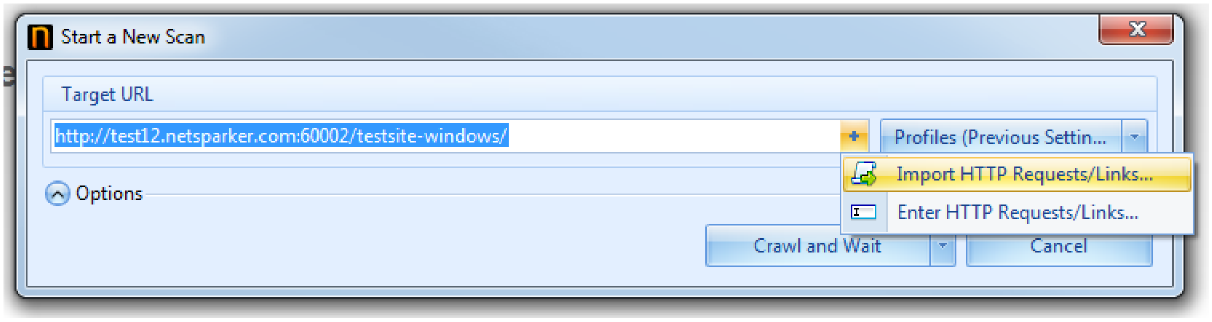

Start a New Scan
Target URL
http://test12.netsparker.com:60002/testsite-windows/
Profiles (Previous Settin...
Import HTTP Requests/Links...
Enter HTTP Requests/Links...
Options
Crawl and Wait
Cancel

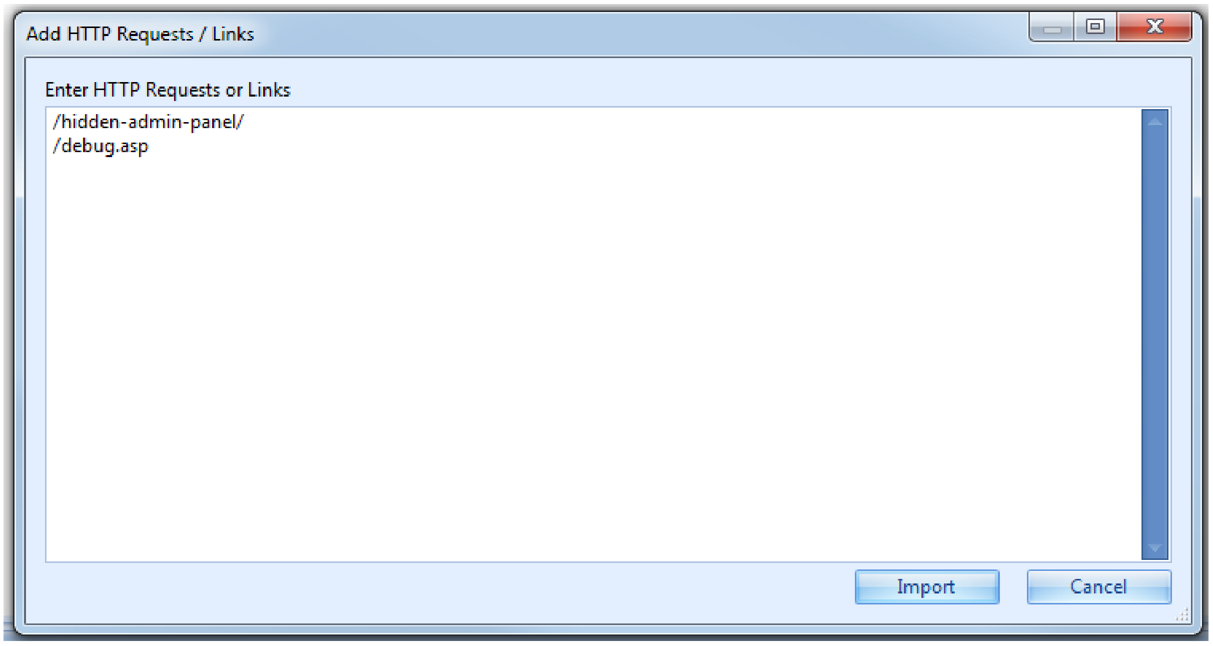
Figure 13: Importing HTTP Requests or Links

You can choose multiple files to import. Don't worry about the file type or extension as Netsparker will automatically identify the file type if it's a known file format.

You can import:

- Complete URLs starting with http(s)
  - A txt file with list of URLs that you want to test, each separated by a newline
- Proxy Logs
  - Fiddler SAZ files
  - Burp XML Logs
  - Paros Logs
  - WebScarab Logs
- List of HTTP Requests
  - Any file that includes a list of HTTP Requests

If you only want to add specific pages to the scan, you can enter links from the **Enter HTTP Requests / Links** interface.



```
/hidden-admin-panel/
/debug.asp
```

Figure 14: Adding HTTP Requests or Links

You can enter:

- Complete URLs starting with http(s)
    - A txt file with list of URLs, each separated with newline that you want to test
- Relative URLs starting with "/"
    - A txt file with list of URLs that you want to test, each separated by a newline

Import Scope

During the import, Netsparker will follow the scan scope rules including Inclusion and Exclusion and Regular Expressions. For example if you import a Proxy log which includes Google and example.com traffic, and if you are scanning example.com then Netsparker will only import requests going to example.com and will not use requests from other domains.

Importing is possible during the crawl phase by right-clicking the Site Map or while starting a new scan from the **Start New Scan** window.

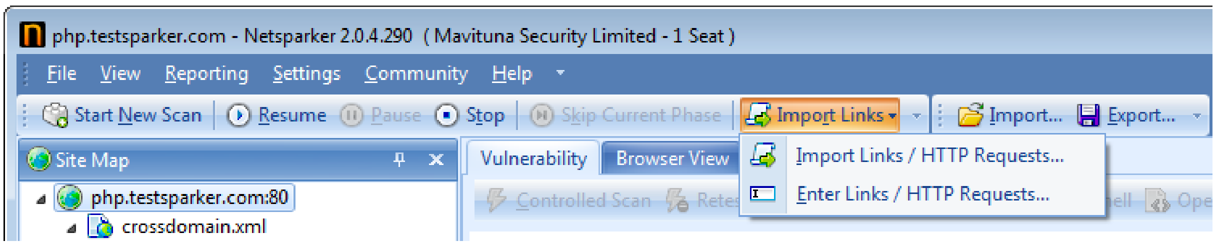


Figure 15: Importing Links into the Site Map

Customizing Scan Settings

Scan Settings Tab

In this tab, you can choose the issues the scanned web application will be tested against by selecting a scan policy. You can also select authentication types, enter custom cookies and crawling options.

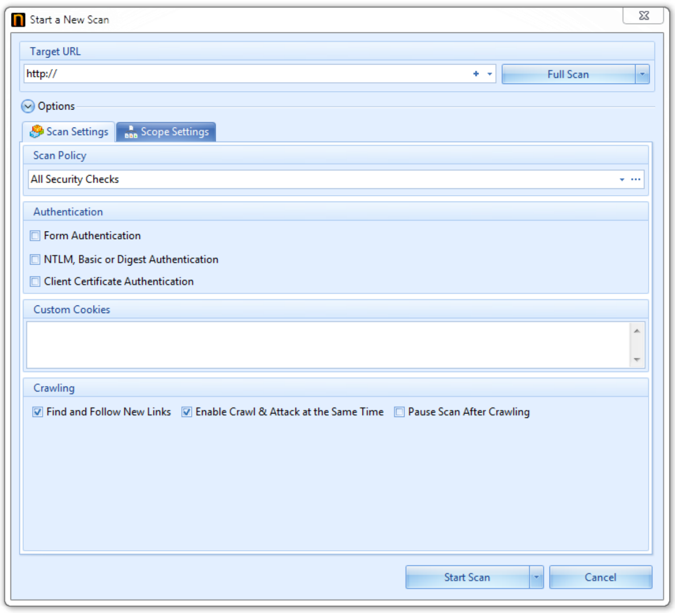
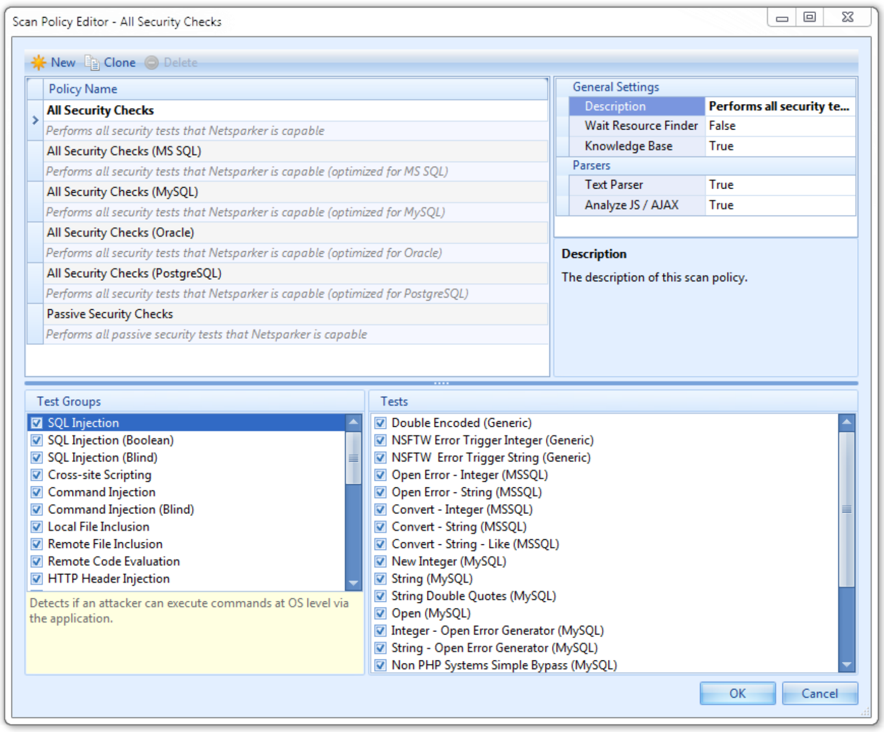


Figure 16: Scan Settings Tab

Scan Policy

A scan policy in Netsparker defines which security tests will be performed on the target web site. Netsparker ships with some built-in policies but it is possible to create new scan policies by using Scan Policy Editor. You can access Scan Policy Editor by clicking the ellipsis button next to scan policy selection combo box or you can use Tools > Scan Policy Editor menu item.

The first list on Scan Policy Editor dialog lists all the scan policies. The rows with grey background denotes built-in scan policies which cannot be modified or deleted. You can create new policies either by clicking the New button to create an empty policy or Clone button to copy an existing one. The policies you have created can be deleted by using the Delete button. The property editor next to the policy list specifies the general settings of the selected scan policy. You can check or uncheck Test Groups below to include or exclude security tests that selected policy will perform. Most of the Test Groups also contain more fine grained Tests that can be checked or unchecked. Some Test Groups can also have property editors which you can fine tune their behaviour.



Scan Policy Editor - All Security Checks

New | Clone | Delete

Policy Name
All Security Checks
Performs all security tests that Netsparker is capable
All Security Checks (MS SQL)
Performs all security tests that Netsparker is capable (optimized for MS SQL)
All Security Checks (MySQL)
Performs all security tests that Netsparker is capable (optimized for MySQL)
All Security Checks (Oracle)
Performs all security tests that Netsparker is capable (optimized for Oracle)
All Security Checks (PostgreSQL)
Performs all security tests that Netsparker is capable (optimized for PostgreSQL)
Passive Security Checks
Performs all passive security tests that Netsparker is capable

General Settings	
Description	Performs all security te...
Wait Resource Finder	False
Knowledge Base	True
Parsers	
Text Parser	True
Analyze JS / AJAX	True

Description

The description of this scan policy.

Test Groups
- [x] SQL Injection
- [x] SQL Injection (Boolean)
- [x] SQL Injection (Blind)
- [x] Cross-site Scripting
- [x] Command Injection
- [x] Command Injection (Blind)
- [x] Local File Inclusion
- [x] Remote File Inclusion
- [x] Remote Code Evaluation
- [x] HTTP Header Injection

Detects if an attacker can execute commands at OS level via the application.

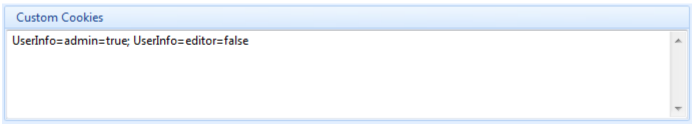
Tests
- [x] Double Encoded (Generic)
- [x] NSFTW Error Trigger Integer (Generic)
- [x] NSFTW Error Trigger String (Generic)
- [x] Open Error - Integer (MSSQL)
- [x] Open Error - String (MSSQL)
- [x] Convert - Integer (MSSQL)
- [x] Convert - String (MSSQL)
- [x] Convert - String - Like (MSSQL)
- [x] New Integer (MySQL)
- [x] String (MySQL)
- [x] String Double Quotes (MySQL)
- [x] Open (MySQL)
- [x] Integer - Open Error Generator (MySQL)
- [x] String - Open Error Generator (MySQL)
- [x] Non PHP Systems Simple Bypass (MySQL)

OK | Cancel

Figure 17: Scan Policy Editor

Custom Cookies

You can set custom cookies. These cookies will be sent with every request and cannot be expired by the server responses.



Custom Cookies

UserInfo=admin=true; UserInfo=editor=false

Figure 18: Custom Cookies

You can easily add a cookie in this format: `CookieName=Value` or add multiple cookies in the form `CookieName1=Value1; CookieName2=Value2`

Crawling

To adjust the crawling process to suit your scenario, Netsparker offers two configuration options:

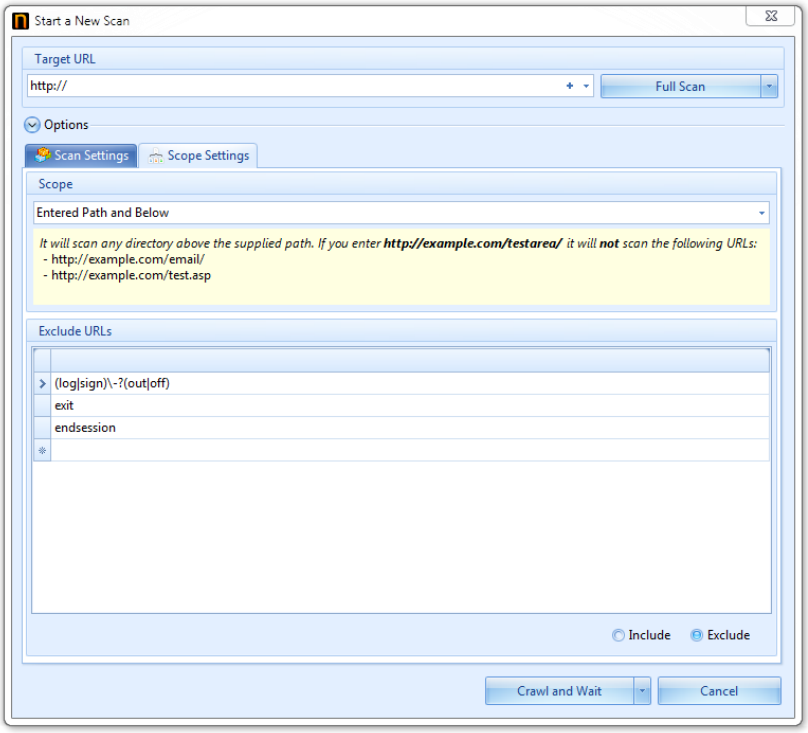
- **Find and Follow New Links:** Enabled by default, this option instructs Netsparker to parse every retrieved page and extract all contained links, subsequently visiting the target pages if they fall within defined scan scope. This discovery process may be disabled when, for example, using proxy mode, if the intention is to test only a specific part of a web application.

- **Enable Crawl and Attack at the Same Time:** Starting with v2.0, Netsparker added the capability to start attacking its target address list even while crawling is still in progress. Disabling this option causes Netsparker to execute crawling and attacking sequentially, as in previous versions.

- **Pause Scan After Crawling:** This option is used to crawl the application without attacking. Crawl Only mode will not identify issues requiring active attacking such as SQL Injection and Cross-site Scripting. This mode is useful if you simply want to assess security best practices passively. Crawl Only mode reports issues such as “Autocomplete is enabled”, “Password Transmitted over HTTP” and “Cookies are not Marked as Secure”.

Scope Settings Tab

Scan scope defines which part of the application Netsparker is allowed to crawl and attack.



Entered Path and Below

It will scan any directory above the supplied path. If you enter ***http://example.com/testarea/*** *it will* ***not*** *scan the following URLs:*
- http://example.com/email/
- http://example.com/test.asp

(log|sign)\-?(out|off)
exit
endsession

Include [✓ Exclude]

Figure 19: Scope Settings Tab

The scan scope can be chosen from three different options:

Entered Path and Below

Scan requests and attacks are only made to the target path, and URLs under that path.

For example if the entered URL was `https://example.com/testfolder/`

The following will be tested:

- https://example.com/testfolder/test.php
- https://example.com/testfolder/test/modify.php
- https://example.com/testfolder/test/
- https://example.com/testfolder/test/
- http://example.com/testfolder/test/

The following will not be tested:

- https://example.com/test.php
  URL is not under the given target

- https://test.example.com
  Different domain or subdomain

Only Entered URL

Scan requests and attacks are only made to the target link and no external links are followed. This function is quite useful if you want to test only one page and all parameters in that page without testing the whole web application.

If you enter http://example.com/test , http://example.com/testx will be tested as well. This scope includes all URLs that start with the given target URL.

For example, if the entered URL was `https://example.com/testfolder/test.php`

The following will be tested:

- https://example.com/testfolder/test.php
- https://example.com/testfolder/test/test.php?id=1

The following will not be tested:

- https://example.com/testfolder/register.php
  URL does not start with the given target

- http://example.com/testfolder/test.php
  Protocol is different (target URL was https)

- http://example.com/test.php
  URL does not start with the given target

- http://test.example.com
  Different domain or subdomain

Whole Domain

The target URL is taken as the start point and all the URLs beginning with the same hostname are scanned.

For example if the entered URL was `https://example.com/testfolder/test.php`

The following will be tested:

- https://example.com/index.php
- http://example.com/register/
- https://example.com/testfolder/test.php

- http://example.com/testfolder/test/test.php?id=1

The following will not be tested:

- http://test.example.com
  Different domain or subdomain

Exclude or Include Links

Netsparker can limit requests according to criteria determined by you. This will allow you to test **only** some parts of the target application, or **exclude** some parts of the application from the test.

To see how this mechanism works, consider the following example:

When you choose the Exclude option and type "`logout`" in the relevant field, Netsparker will not make requests to links including the text "`logout`" while scanning your application.

Netsparker will only look for the name in the URL not in the Link's name or the title. If the Logout page's name is sessionend.php you need to use "sessionend" instead of the text in the link such as "Logout".

In the same way, when you choose the Include option and type "`test`" in the relevant field, Netsparker will only make requests to links including the text "`test`".

This field allows you to enter RegEx rules. For example if you want to exclude `logout.php` and `/nottotest` folder you can use the following RegEx and choose the Exclude option.

```
(logout\.php|/nottotest)
```

Creating and Using Custom Scan Profiles

The combination of configuration settings that governs the execution of a scan is known as a scan profile. Netsparker's default installation comes with a number of pre-defined scan profiles, which are accessible from the Profiles drop-down list on the Start a New Scan dialog.

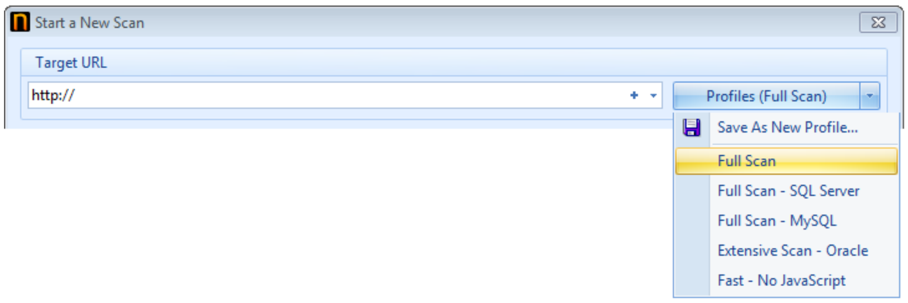
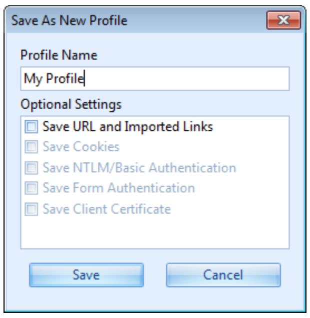


Figure 20: Select a Scan Profile

To populate your scan settings from one of the pre-defined profiles, just select it from the list.

To create a custom profile, first configure your desired scan settings and then select the Save As New Profile … option from the same drop-down list.

When you save a custom profile, you will prompted to choose a name and to optionally set a number of options that determine which aspects of the loaded scan settings will be saved.


Save As New Profile

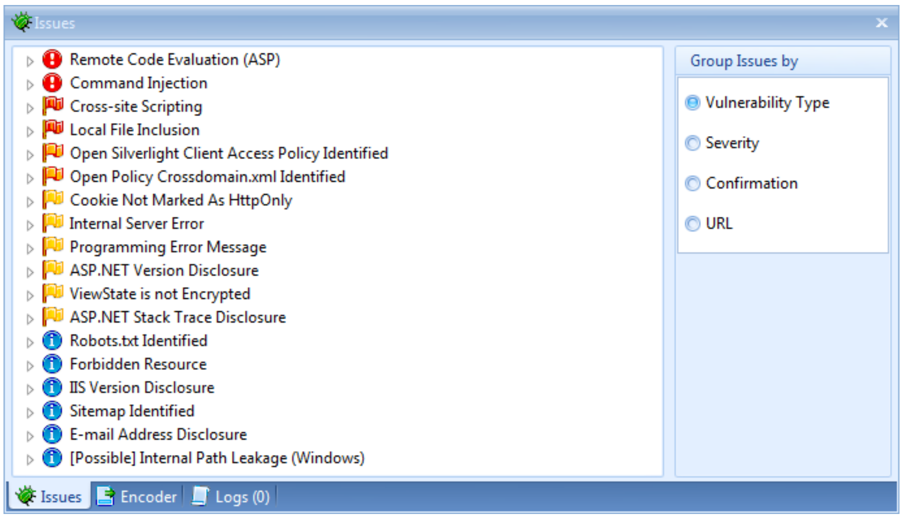
Profile Name: My Profile

Optional Settings
- [ ] Save URL and Imported Links
- [ ] Save Cookies
- [ ] Save NTLM/Basic Authentication
- [ ] Save Form Authentication
- [ ] Save Client Certificate

Figure 21: Save a Custom Scan Profile

Analyzing Vulnerabilities

When Netsparker detects a new issue as a result of a scan, you can access the details of this issue from the Issues and Sitemap panels. There is no need to wait for completion of the scan to examine the issues.

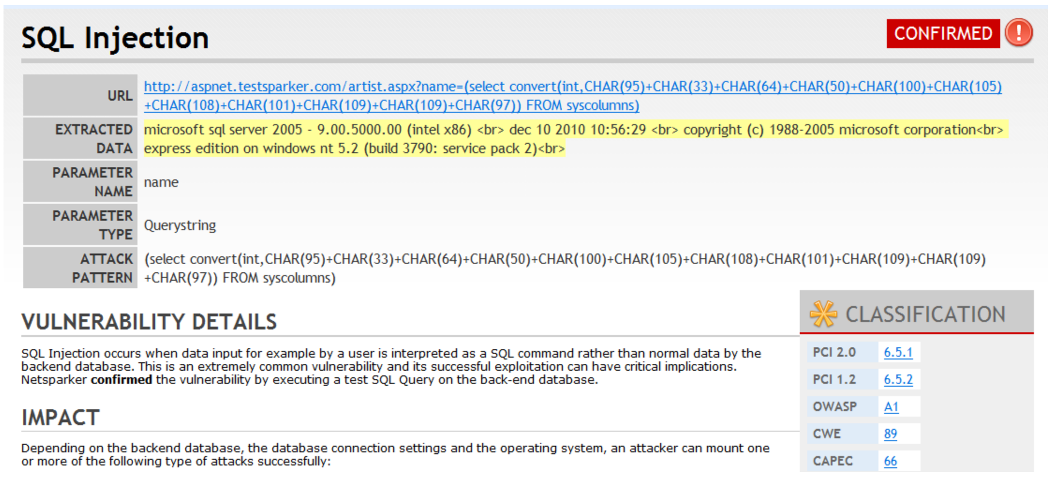

Issues

- Remote Code Evaluation (ASP)
- Command Injection
- Cross-site Scripting
- Local File Inclusion
- Open Silverlight Client Access Policy Identified
- Open Policy Crossdomain.xml Identified
- Cookie Not Marked As HttpOnly
- Internal Server Error
- Programming Error Message
- ASP.NET Version Disclosure
- ViewState is not Encrypted
- ASP.NET Stack Trace Disclosure
- Robots.txt Identified
- Forbidden Resource
- IIS Version Disclosure
- Sitemap Identified
- E-mail Address Disclosure
- [Possible] Internal Path Leakage (Windows)

Group Issues by
- [✓ Vulnerability Type]
- Severity
- Confirmation
- URL

Figure 22: Issues Panel

In the Issues panel, issues are listed in groups according to 4 different properties.



SQL Injection

CONFIRMED

URL	http://aspnet.testsparker.com/artist.aspx?name=(select convert(int,CHAR(95)+CHAR(33)+CHAR(64)+CHAR(50)+CHAR(100)+CHAR(105)+CHAR(108)+CHAR(101)+CHAR(109)+CHAR(109)+CHAR(97)) FROM syscolumns)
EXTRACTED DATA	microsoft sql server 2005 - 9.00.5000.00 (intel x86) dec 10 2010 10:56:29 copyright (c) 1988-2005 microsoft corporation express edition on windows nt 5.2 (build 3790: service pack 2)
PARAMETER NAME	name
PARAMETER TYPE	Querystring
ATTACK PATTERN	(select convert(int,CHAR(95)+CHAR(33)+CHAR(64)+CHAR(50)+CHAR(100)+CHAR(105)+CHAR(108)+CHAR(101)+CHAR(109)+CHAR(109)+CHAR(97)) FROM syscolumns)

VULNERABILITY DETAILS

SQL Injection occurs when data input for example by a user is interpreted as a SQL command rather than normal data by the backend database. This is an extremely common vulnerability and its successful exploitation can have critical implications. Netsparker **confirmed** the vulnerability by executing a test SQL Query on the back-end database.

IMPACT

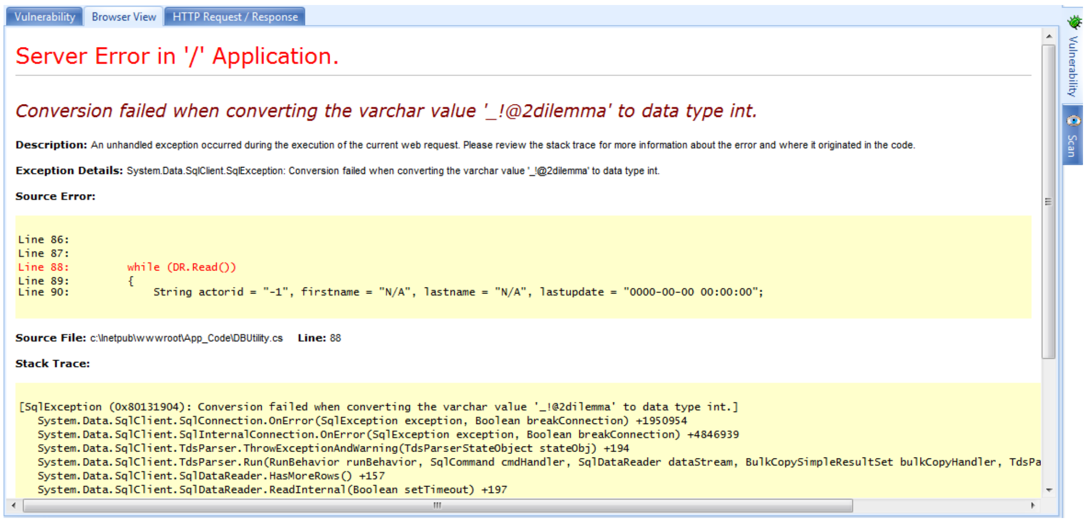
Depending on the backend database, the database connection settings and the operating system, an attacker can mount one or more of the following type of attacks successfully:

CLASSIFICATION

PCI 2.0	6.5.1
PCI 1.2	6.5.2
OWASP	A1
CWE	89
CAPEC	66

Figure 23: Sample Issue Details

After you choose the Issue Details that you want to see from the **Sitemap** or **Issues** list, you can see the page including the issue from the **Browser View** tab, and at the same time you can see HTTP requests and responses from the **HTTP Request / Response** tab.



Server Error in '/' Application.

Conversion failed when converting the varchar value '_!@2dilemma' to data type int.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: Conversion failed when converting the varchar value '_!@2dilemma' to data type int.

Source Error:

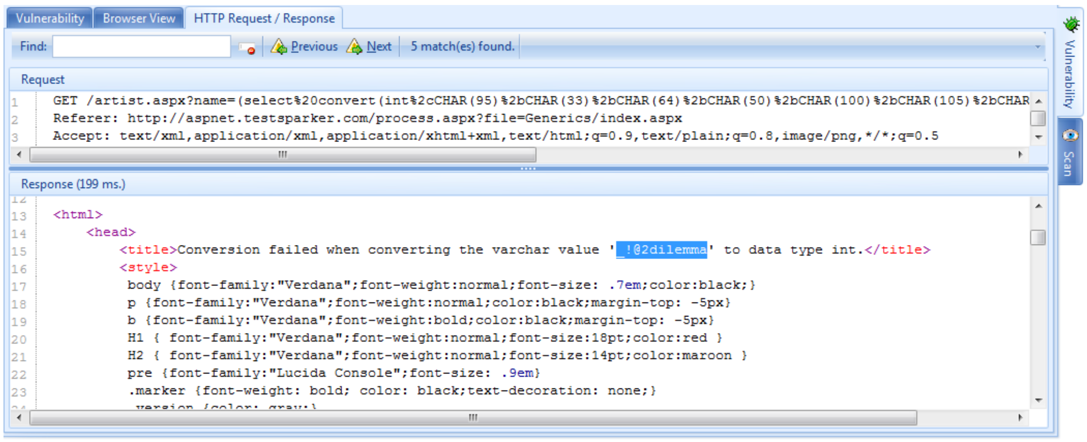
```
Line 86:
Line 87:
Line 88:        while (DR.Read())
Line 89:        {
Line 90:            String actorid = "-1", firstname = "N/A", lastname = "N/A", lastupdate = "0000-00-00 00:00:00";
```

Source File: c:\Inetpub\wwwroot\App_Code\DBUtility.cs **Line:** 88

Stack Trace:

```
[SqlException (0x80131904): Conversion failed when converting the varchar value '_!@2dilemma' to data type int.]
   System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection) +1950954
   System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean breakConnection) +4846939
   System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj) +194
   System.Data.SqlClient.TdsParser.Run(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsPa
   System.Data.SqlClient.SqlDataReader.HasMoreRows() +157
   System.Data.SqlClient.SqlDataReader.ReadInternal(Boolean setTimeout) +197
```

Figure 24: Browser View

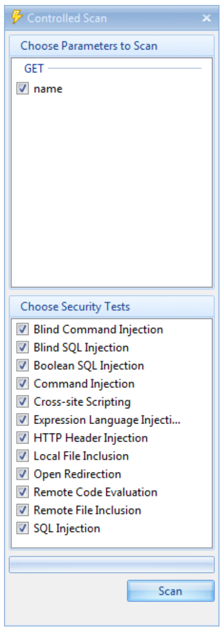


```
GET /artist.aspx?name=(select%20convert(int%2cCHAR(95)%2bCHAR(33)%2bCHAR(64)%2bCHAR(50)%2bCHAR(100)%2bCHAR(105)%2bCHAR
Referer: http://aspnet.testsparker.com/process.aspx?file=Generics/index.aspx
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
```

```
<html>
    <head>
        <title>Conversion failed when converting the varchar value '_!@2dilemma' to data type int.</title>
        <style>
         body {font-family:"Verdana";font-weight:normal;font-size: .7em;color:black;}
         p {font-family:"Verdana";font-weight:normal;color:black;margin-top: -5px}
         b {font-family:"Verdana";font-weight:bold;color:black;margin-top: -5px}
         H1 { font-family:"Verdana";font-weight:normal;font-size:18pt;color:red }
         H2 { font-family:"Verdana";font-weight:normal;font-size:14pt;color:maroon }
         pre {font-family:"Lucida Console";font-size: .9em}
         .marker {font-weight: bold; color: black;text-decoration: none;}
```

Figure 25: HTTP Request / Response View

Controlled Scan and Retest



Controlled Scan and **Retest** allows you to test a vulnerability during or after the scan. This flexibility gives you additional scope for the scanning and reporting of vulnerabilities in an application.

Controlled Scan is a simple attack method that can be used to scan a link with selected parameters and engines. It is useful when crawling a web application or imported links. It can also be used as an alternative to type-and-go scanning, since it allows Netsparker to work as a proxy and attack links in a controlled manner.

To use this option, click the Controlled Scan button on a links' information page, and select the parameters and security tests to use while scanning. Pressing the Scan button will carry out the required web requests and complete that controlled scan.

A **Retest** is an easy way of scanning a vulnerability to check if it is fixed. When Netsparker identifies a vulnerability, the Retest button is activated, allowing you to repeat the scan on the vulnerability with the same parameters and engines. Retesting a vulnerability informs you of the test results, so you can be sure that the vulnerability is fixed.

Note: A full scan is always recommended even if you use the Retest feature to confirm that vulnerabilities are fixed. This is because you may have implemented new vulnerabilities within the application.

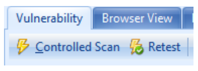
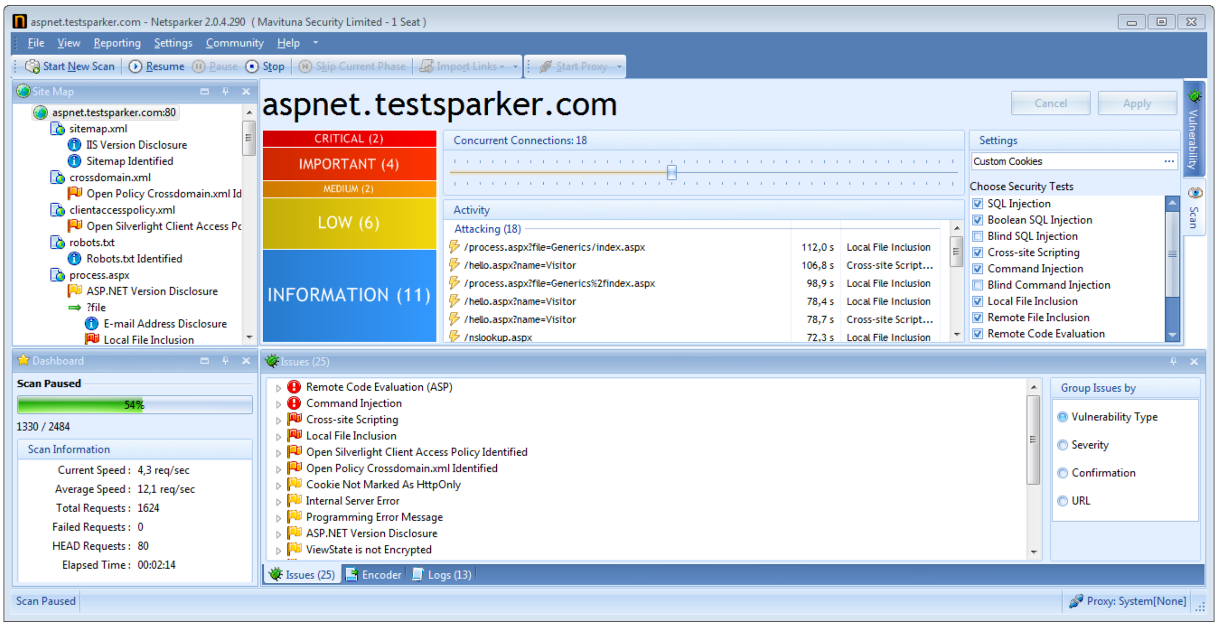


Figure 26: Launching a Controlled Scan

Using Netsparker

Scan Summary Dashboard

During scanning Netsparker presents a real-time view of its activity and status in the Scan Summary Dashboard. From this interface, you may also modify certain key scan settings, with your changes having an immediate effect on the scan session in progress.



Site Map:

- aspnet.testsparker.com:80
  - sitemap.xml
    - IIS Version Disclosure
    - Sitemap Identified
  - crossdomain.xml
    - Open Policy Crossdomain.xml Id
  - clientaccesspolicy.xml
    - Open Silverlight Client Access Pc
  - robots.txt
    - Robots.txt Identified
  - process.aspx
    - ASP.NET Version Disclosure
    - ?file
      - E-mail Address Disclosure
      - Local File Inclusion

Concurrent Connections: 18

Activity

Attacking (18)		
/process.aspx?file=Generics/index.aspx	112,0 s	Local File Inclusion
/hello.aspx?name=Visitor	106,8 s	Cross-site Script...
/process.aspx?file=Generics%2findex.aspx	98,9 s	Local File Inclusion
/hello.aspx?name=Visitor	78,4 s	Local File Inclusion
/hello.aspx?name=Visitor	78,7 s	Cross-site Script...
/nslookup.aspx	72,3 s	Local File Inclusion

Settings

Custom Cookies

Choose Security Tests
- [✓] SQL Injection
- [✓] Boolean SQL Injection
- [ ] Blind SQL Injection
- [✓] Cross-site Scripting
- [✓] Command Injection
- [ ] Blind Command Injection
- [✓] Local File Inclusion
- [✓] Remote File Inclusion
- [✓] Remote Code Evaluation

Scan Paused

54%

1330 / 2484

Scan Information

Current Speed :	4,3 req/sec
Average Speed :	12,1 req/sec
Total Requests :	1624
Failed Requests :	0
HEAD Requests :	80
Elapsed Time :	00:02:14

Issues (25)
- Remote Code Evaluation (ASP)
- Command Injection
- Cross-site Scripting
- Local File Inclusion
- Open Silverlight Client Access Policy Identified
- Open Policy Crossdomain.xml Identified
- Cookie Not Marked As HttpOnly
- Internal Server Error
- Programming Error Message
- ASP.NET Version Disclosure
- ViewState is not Encrypted

Group Issues by
- [✓ Vulnerability Type]
- Severity
- Confirmation
- URL

Figure 27: Netsparker's Scan Dashboard

The Scan Summary Dashboard provides at-a-glance feedback about the active scanning session. The bar at the left provides a graphical summary of the issues detected thus far, classified according to their severity. The activity pane in the center reports the last completed activity on each of Netsparker's open HTTP connections. Connections are organized into two groups (crawling and attacking) and, for each connection, Netsparker reports the target URL and the time since the last request was issued. For attacking connections, Netsparker also identifies the security check type used to perform the attack.

During a scan, you may also use the Scan Summary Dashboard to modify some of the key settings that you specified when launching the scan session:

- Concurrent Connections: The number of concurrent connections that Netsparker uses for HTTP requests. The selected value is used to define the number of concurrent connections kept by both crawling and attacking.
- Custom Cookies: Name value pairs that represent user-defined cookies that will be added to every HTTP request. You might use this feature, for example, to update an expired session cookie.

- **Security Tests:** The list of all security tests in Netsparker. Security tests which are currently active for the scan are checked.

After changing the value of one of these settings, click the **Apply** button to accept and activate your changes, or **Cancel** to discard them.

Include & Exclude

You can exclude certain parts of the target application from the scan during the crawl or attack. When Netsparker identifies a new resource to attack, it will add it to the Site Map. From here, all you need to do is right-click the node that you want to exclude from the attack and choose **Exclude**.

You can also choose to **Exclude** a branch; this means all new resources identified under this branch will be excluded from the scan automatically as well. You can choose **Include** to revert these changes. All URLs are included by default.

By default, after the crawl, Netsparker will not pause and will start attacking immediately. If you know that you need to exclude some parts of the application before attack phase, start the scan in **Crawl & Wait** mode. In this case, Netsparker will pause the scan after the crawl, so you can analyze the site map and exclude certain URLs. You can then click **Resume** (F5) to start attacking phase.

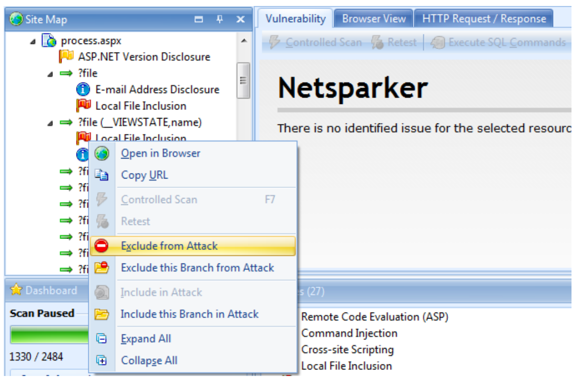
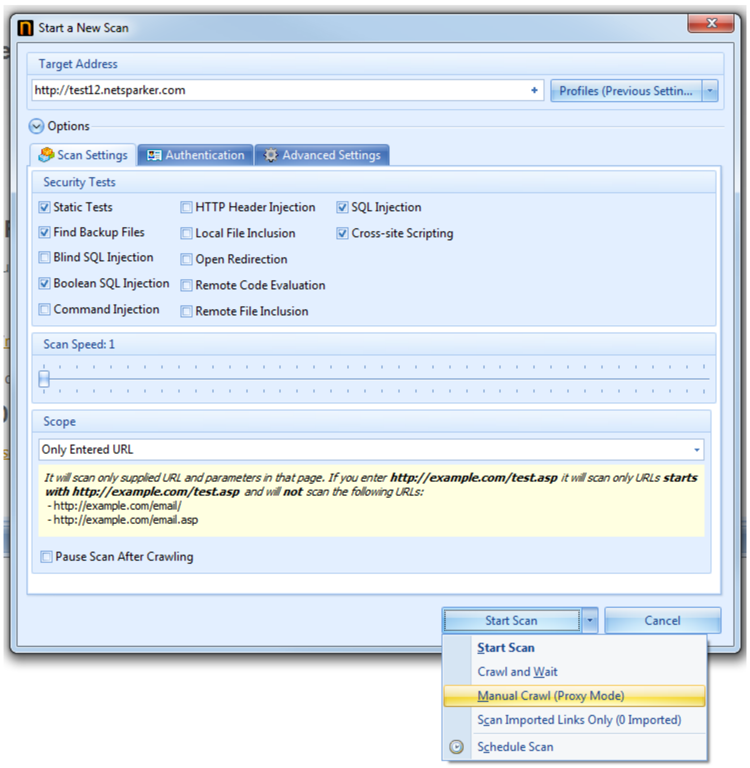


Figure 28: Setting Individual Site Map URL Actions

Manual Crawl (Proxy Mode)

Manual crawl is generally useful if you need to scan only a part of the website.

- Start a new scan with the settings you want to use
- Choose **Manual Crawl (Proxy Mode)** from the **Start Scan** button


Start a New Scan

Target Address
http://test12.netsparker.com
Profiles (Previous Settin...

Options
Scan Settings | Authentication | Advanced Settings

Security Tests
☑ Static Tests ☐ HTTP Header Injection ☑ SQL Injection
☑ Find Backup Files ☐ Local File Inclusion ☑ Cross-site Scripting
☐ Blind SQL Injection ☐ Open Redirection
☑ Boolean SQL Injection ☐ Remote Code Evaluation
☐ Command Injection ☐ Remote File Inclusion

Scan Speed: 1

Scope
Only Entered URL

It will scan only supplied URL and parameters in that page. If you enter ***http://example.com/test.asp*** *it will scan only URLs* ***starts with http://example.com/test.asp*** *and will* ***not*** *scan the following URLs:*
- http://example.com/email/
- http://example.com/email.asp

☐ Pause Scan After Crawling

Start Scan | Cancel

Start Scan
Crawl and Wait
Manual Crawl (Proxy Mode)
Scan Imported Links Only (0 Imported)
Schedule Scan

Figure 29: Launching Proxy Mode

The scan will start but Netsparker will pause the scan after requesting the starting URL and will not crawl other links. You'll notice that in the toolbar, the Proxy is already started.

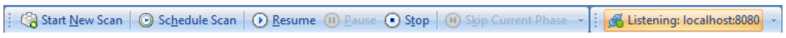
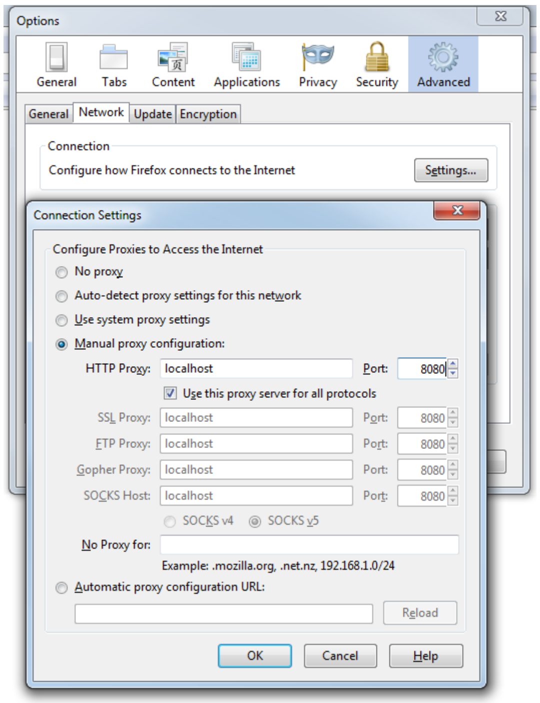


Figure 30: Confirmation of Proxy Mode Operation

If your web application requires **NTLM**, **Basic, Kerberos** or **Digest Authentication**, it must be configured (from the **Authentication** tab) prior to launching proxy mode.

Now, open your browser and configure your browser's proxy to Netsparker's proxy.

Sample Proxy Settings in Firefox


Options

General | Tabs | Content | Applications | Privacy | Security | Advanced

General | Network | Update | Encryption

Connection

Configure how Firefox connects to the Internet [Settings...]

Connection Settings

Configure Proxies to Access the Internet

- ○ No proxy
- ○ Auto-detect proxy settings for this network
- ○ Use system proxy settings
- ● Manual proxy configuration:

HTTP Proxy: localhost Port: 8080

☑ Use this proxy server for all protocols

SSL Proxy: localhost Port: 8080

FTP Proxy: localhost Port: 8080

Gopher Proxy: localhost Port: 8080

SOCKS Host: localhost Port: 8080

○ SOCKS v4 ● SOCKS v5

No Proxy for:

Example: .mozilla.org, .net.nz, 192.168.1.0/24

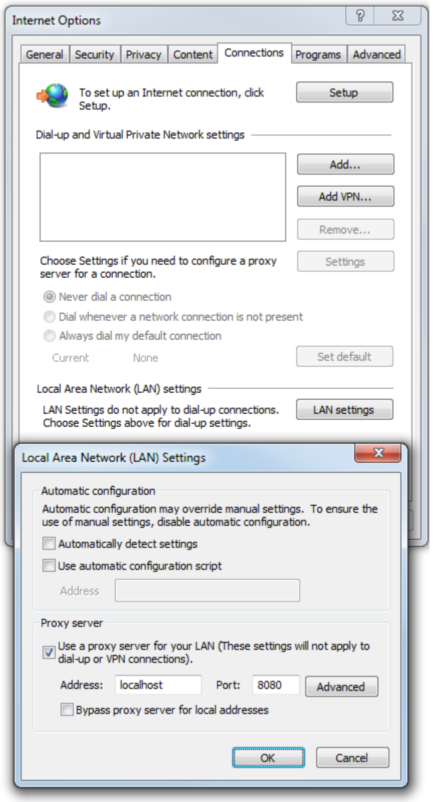
○ Automatic proxy configuration URL:

[Reload]

[OK] [Cancel] [Help]

Figure 31: Configuring Firefox to Use a Proxy

Sample Proxy Settings in Internet Explorer


Internet Options

General | Security | Privacy | Content | Connections | Programs | Advanced

To set up an Internet connection, click Setup. [Setup]

Dial-up and Virtual Private Network settings

[Add...] [Add VPN...] [Remove...]

Choose Settings if you need to configure a proxy server for a connection. [Settings]

- ● Never dial a connection
- ○ Dial whenever a network connection is not present
- ○ Always dial my default connection

Current None [Set default]

Local Area Network (LAN) settings

LAN Settings do not apply to dial-up connections. Choose Settings above for dial-up settings. [LAN settings]

Local Area Network (LAN) Settings

Automatic configuration

Automatic configuration may override manual settings. To ensure the use of manual settings, disable automatic configuration.

☐ Automatically detect settings

☐ Use automatic configuration script

Address

Proxy server

☑ Use a proxy server for your LAN (These settings will not apply to dial-up or VPN connections).

Address: localhost Port: 8080 [Advanced]

☐ Bypass proxy server for local addresses

[OK] [Cancel]

Figure 32: Configuring Internet Explorer to Use a Proxy

Now all you need to do is browse the target application from your web browser, as soon as you visit new URLs you'll see that the Sitemap in Netsparker will start to show those new URLs.

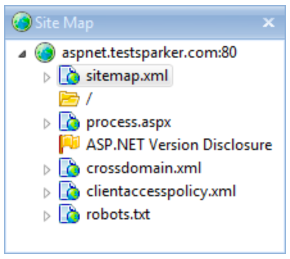


Figure 33: Original Site Map View

After visiting the following URL from the browser: http://test12.netsparker.com:60001/XSS-Basic/ Netsparker will get that URL and show it in the sitemap.

This is the new sitemap view:

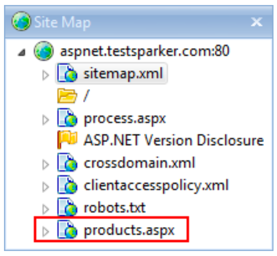


Figure 34: SIte Map View After Browsing in Proxy Mode

In **Manual Crawl (Proxy Mode)** Netsparker will **not** find and crawl new links. This includes parameters pointing to the same page. For example if you visit test.php, Netsparker won't attack "test.php?id=1" unless you visit "test.php?id=1" as well.

Netsparker will only get links and parameters from the proxy. If you want Netsparker to crawl further, use **Crawl & Wait** mode and start the proxy from the toolbar.

When you’ve crawled all the links and parameters you want to test, you can click the **Resume** button from the toolbar and Netsparker will start the attacking phase and report vulnerabilities.

If you want to exclude some of the crawled links from the scan, you can right click them on the sitemap and choose **Exclude From Attack**.

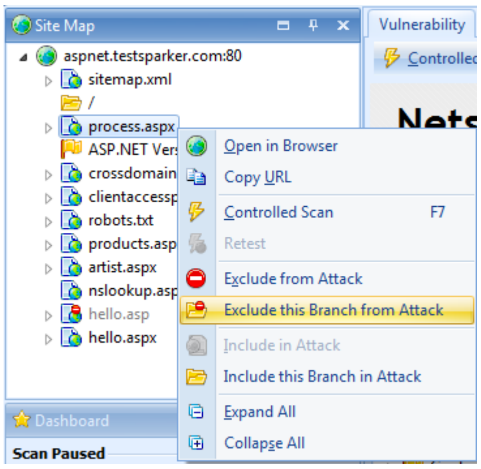


Figure 35: Excluding a URL From Attack

Final Notes About Manual Crawl

- Cookies originating from the proxy request will override cookies originating from Netsparker. For example, if you login to the application from your web browser, Netsparker will use that session and will see pages as you see them.
- The Manual crawl will not crawl new links and parameters automatically (*use Crawl & Wait for this*)
- You can't use a Proxy during the Attack phase, a proxy can only be used during the Crawl phase
- If you are testing "localhost" ensure that localhost is not in the "bypass proxy list" of your browser
- While you browse pages from your web browser, Crawler will do the very same request and will report passively identifiable vulnerabilities such as "Programming Error Messages" and "Version Disclosure"

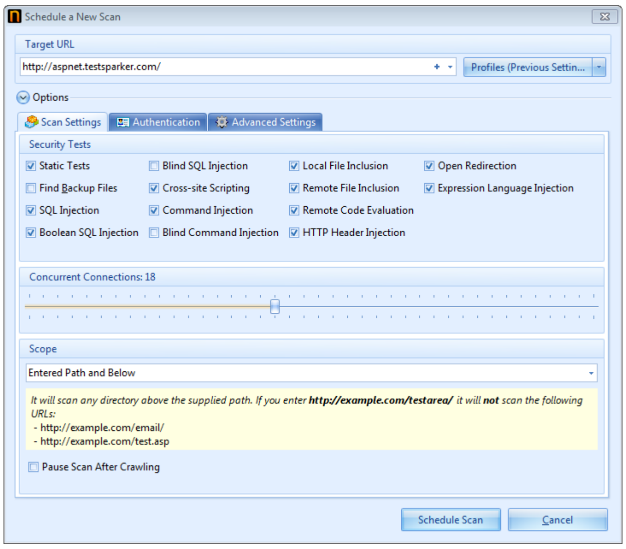
Scheduling

Thanks to scheduling support, you can make your Netsparker scans run with daily, weekly or monthly intervals, and have the results saved to the given folder after the scan is complete.

To schedule a new scan click Start a New Scan or Schedule a New Scan then customize the scan and click the Schedule Scan button.

Scan Profile and Schedule Settings

First thing to do is create a profile to specify the settings scans will use while running.



Schedule a New Scan

Target URL

http://aspnet.testsparker.com/

Security Tests

✓ Static Tests	☐ Blind SQL Injection	✓ Local File Inclusion	✓ Open Redirection
☐ Find Backup Files	✓ Cross-site Scripting	✓ Remote File Inclusion	✓ Expression Language Injection
✓ SQL Injection	✓ Command Injection	✓ Remote Code Evaluation	
✓ Boolean SQL Injection	☐ Blind Command Injection	✓ HTTP Header Injection	

Concurrent Connections: 18

Scope

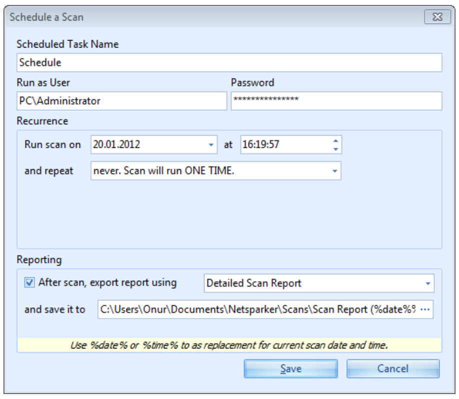
Entered Path and Below

It will scan any directory above the supplied path. If you enter ***http://example.com/testarea/*** *it will* ***not*** *scan the following URLs:*
- http://example.com/email/
- http://example.com/test.asp

☐ Pause Scan After Crawling

Figure 36: Schedule a New Scan Window

After the profile is set, you should specify the Scheduling settings by clicking the **Schedule Scan** button. After this button is clicked, the **Schedule Scan** window will open.



Schedule a Scan

Scheduled Task Name: Schedule

Run as User: PC\Administrator

Password: ***************

Recurrence

Run scan on 20.01.2012 at 16:19:57

and repeat never. Scan will run ONE TIME.

Reporting

✓ After scan, export report using Detailed Scan Report

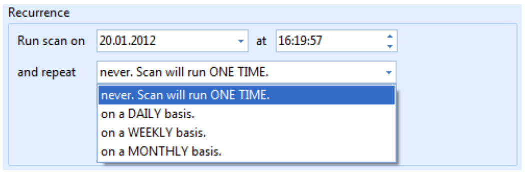
and save it to C:\Users\Onur\Documents\Netsparker\Scans\Scan Report (%date%?

Use %date% or %time% to as replacement for current scan date and time.

Figure 37: Schedule Settings

In the above window, the following settings should be modified:

- **Scheduled Task Name:** Name of task to be scheduled
- **Run as User / Password:** User name and password for running the relevant scan task
- **Recurrence:** The time and intervals the scan will be repeated
    - **Run scan on / at:** Start date and time of the scan.
    - **Repeat:** Repeat setting of the scan. This can be set as daily, weekly or monthly.



Recurrence

Run scan on 20.01.2012 at 16:19:57

and repeat never. Scan will run ONE TIME.

Figure 38: Scan Recurrence Settings

- Reporting: Reporting settings

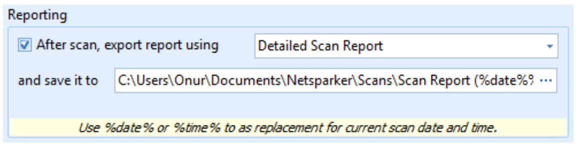

Reporting
After scan, export report using: Detailed Scan Report
and save it to: C:\Users\Onur\Documents\Netsparker\Scans\Scan Report (%date%
Use %date% or %time% to as replacement for current scan date and time.

Figure 39: Reporting Settings

  - Path: File path the report will be saved.

    Report path can include %date% or %time% variables in the file name. This will allow you to run the same scan multiple times and have a separate report file for each. For example you can use the following path:
    `C:\My Reports\Testsite-%date%-%time%.pdf`

    The filenames will be generated in the following format:
    `c:\My Reports\Testsite-05032009-125012.pdf`

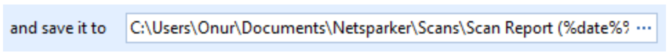

and save it to: C:\Users\Onur\Documents\Netsparker\Scans\Scan Report (%date%

Figure 40: Report Path

After the relevant settings are made and saved, at the specified time, the "Windows Task Scheduler" will start the relevant scan task and save the result report into the specified folder when the scan is complete. After the schedule has been saved, you can use "Windows Task Scheduler" to track the scan task.

Reporting

Netsparker's reporting features allow you to generate a range of standard reports as well as defining report templates that support custom reporting.

Netsparker ships with four predefined report formats. Custom reports may be easily created by defining new templates, based on the shipped report templates.

The shipped report templates currently include a Detailed Scan Report, vulnerability lists in CSV and XML format and a scan Comparison Report.

Generating a Report

With a scan completed or in progress, a scan report may be generated by selecting the relevant report type from the Reporting menu:

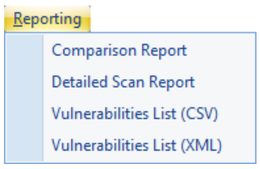
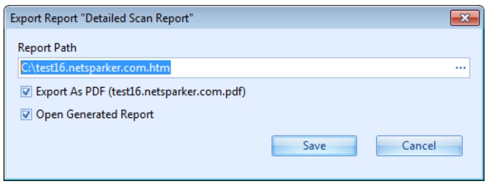


Figure 41: The Reporting Menu

The standard report templates that ship with Netsparker are as follows:

- Detailed Scan Report: A comprehensive statistical summary of the detected vulnerabilities, together with detailed diagnostic information for each vulnerability. This report format is human readable and is generated as HTML and, optionally, PDF output.
- Vulnerabilities List (CSV): A detailed tabulated presentation of the detected vulnerabilities that may be analysed in Excel or another tool that supports CSV format.
- Vulnerabilities List (XML): A raw XML representation of the detected vulnerabilities that is intended for conversion into other file formats or to be imported into other security tools.
- Comparison Report: A comparison between the current scan session and one or more previously saved sessions. This report (which is also available in HTML and PDF formats) shows the evolution of your application's security status over a period of time, enabling you to track progress in resolving issues and to detect new issues that have arisen (or returned) since you started tracking your security status.

After selecting the required template for your report, you will be prompted to specify the location and name of the generated output file and to select your preferred output options:


Export Report "Detailed Scan Report"

Report Path

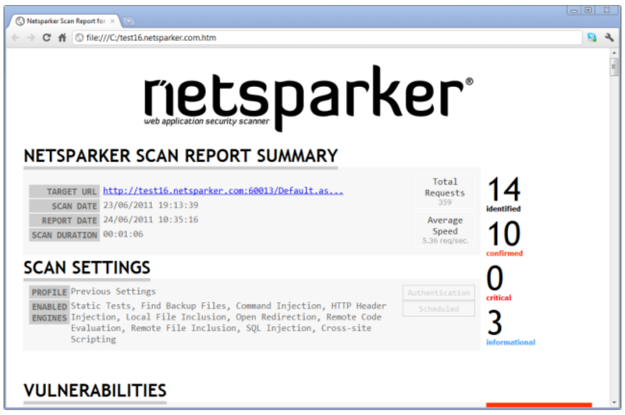
C:\test16.netsparker.com.htm

☑ Export As PDF (test16.netsparker.com.pdf)

☑ Open Generated Report

Figure 42: Report Output Options

Generation of the report takes a few seconds and, when complete, if you chose one of the HTML format reports and selected the **Open Generated Report** option, it will be displayed in your Browser.


Netsparker Scan Report for
file:///C:/test16.netsparker.com.htm
netsparker®
web application security scanner

NETSPARKER SCAN REPORT SUMMARY

TARGET URL	http://test16.netsparker.com:60013/Default.as...
SCAN DATE	23/06/2011 19:13:39
REPORT DATE	24/06/2011 10:35:16
SCAN DURATION	00:01:06

Total Requests 359

Average Speed 5.36 req/sec.

14 identified

10 confirmed

0 critical

3 informational

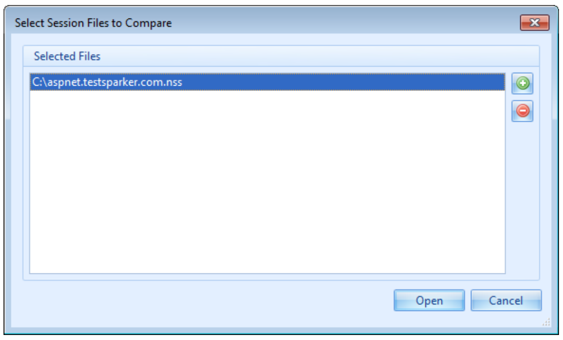
SCAN SETTINGS

PROFILE	Previous Settings
ENABLED ENGINES	Static Tests, Find Backup Files, Command Injection, HTTP Header Injection, Local File Inclusion, Open Redirection, Remote Code Evaluation, Remote File Inclusion, SQL Injection, Cross-site Scripting

VULNERABILITIES

Figure 43: Detailed Scan Report

The Comparison Report introduces one additional step in which you select one or more historic Netsparker scan session files to compare with your current scan:



Select Session Files to Compare

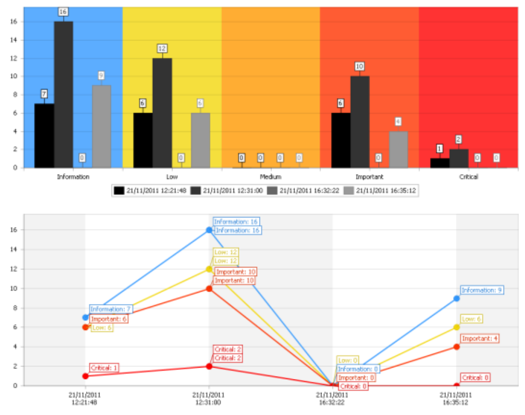
Selected Files

C:\aspnet.testsparker.com.nss

Figure 44: Select Session Files to Compare

Once generated, the report provides a graphical summary of the evolution of your security status, as well as a detailed list that represents the union of all vulnerabilities detected in any of the compared scan sessions.

netsparker® web application security scanner



VULNERABILITY COMPARISON

aspnet.testsparker.com	Parameter	Method		Vulnerability	21/11/2011 12:21:48	21/11/2011 12:31:00	21/11/2011 12:31:00	21/11/2011 16:32:22	21/11/2011 16:35:12
/			🟨	ASP.NET Version Disclosure	New	Not Fixed	Not Fixed	Fixed	Revived
			🟦	[Possible] Internal Path Leakage (Windows)	New	Not Fixed	Not Fixed	Fixed	Revived
/artist.aspx	name	GET	🟥	SQL Injection	Not Exist	New	Not Fixed	Fixed	Fixed
			🟦	Microsoft SQL Server Identified	Not Exist	New	Not Fixed	Fixed	Fixed
			🟦	[Possible] Internal Path Leakage (Windows)	New	Not Fixed	Not Fixed	Fixed	Revived
/hello.asp	name	GET	🟥	Cross-site Scripting	New	Fixed	Fixed	Fixed	Fixed
	name	GET	🟥	Cross-site Scripting	New	Fixed	Fixed	Fixed	Fixed
	name	GET	🟥	Cross-site Scripting	New	Not Fixed	Not Fixed	Fixed	Revived
	name	GET	🟥	Cross-site Scripting	New	Fixed	Fixed	Fixed	Fixed
			🟨	Cookie Not Marked As HttpOnly	New	Not Fixed	Not Fixed	Fixed	Revived
	name	GET	🟨	Programming Error Message	New	Not Fixed	Not Fixed	Fixed	Revived
/hello.aspx	name	GET	🟥	Cross-site Scripting	New	Not Fixed	Not Fixed	Fixed	Revived
	name	GET	🟥	Cross-site Scripting	New	Not Fixed	Not Fixed	Fixed	Revived
	name	GET	🟥	Cross-site Scripting	Not Exist	New	Not Fixed	Fixed	Fixed
	name	GET	🟥	Cross-site Scripting	Not Exist	New	Not Fixed	Fixed	Fixed
			🟥	[Possible] Permanent Cross-site Scripting	Not Exist	Fixed	Fixed	Fixed	New
/jscss/			🟦	Forbidden Resource	New	Not Fixed	Not Fixed	Fixed	Revived
/nslookup.aspx	param	POST	🟥	Command Injection	New	Fixed	Fixed	Fixed	Fixed
			🟦	[Possible] Internal Path Leakage (Windows)	New	Not Fixed	Not Fixed	Fixed	Revived
/process.aspx			🟨	ViewState is not Encrypted	New	Not Fixed	Not Fixed	Fixed	Revived
			🟦	E-mail Address Disclosure	New	Not Fixed	Not Fixed	Fixed	Revived
/process.aspx%3ffile%3dGenerics/			🟦	IIS Version Disclosure	New	Not Fixed	Not Fixed	Fixed	Revived
/products.aspx	pro	GET	🟨	Internal Server Error	New	Not Fixed	Not Fixed	Fixed	Revived
	pro	GET	🟨	ASP.NET Stack Trace Disclosure	New	Not Fixed	Not Fixed	Fixed	Revived
			🟦	[Possible] Internal Path Leakage (Windows)	New	Not Fixed	Not Fixed	Fixed	Revived
/products.aspx/			🟦	[Possible] Internal Path Leakage (Windows)	Not Exist	Fixed	Fixed	Fixed	New
/products.aspx/%22ns=%22netsparker(0x00008A)			🟦	[Possible] Internal Path Leakage (Windows)	Not Exist	Fixed	Fixed	Fixed	New

Figure 45: Detailed Comparison of Vulnerability Evolution

Custom Reporting

Netsparker allows you to define your own report templates. These may be used in generating custom reports that suit your business needs, and for integration with other software applications. The custom reporting tool employs Razor templating engine to run your C# code to generate reports.

How does it work?

At start-up, Netsparker scans for C# template files (*.cshtml) in the "Report Templates" directory, which is located within the “Resources” sub-directory of the “Netsparker” data directory (by default resides in current Windows user’s Documents/My Documents directory). Every identified file will be visible in the "Reporting" menu as a custom report.

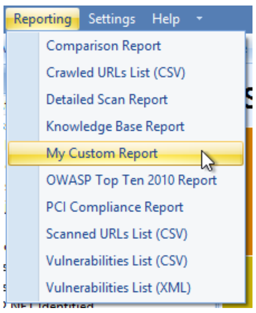


Figure 46: The Customized Reporting Menu

Scripting Language

Netsparker’s scripting language is C#, making it easy to make simple changes.

Here is a sample custom report code:

```
@using System
@using System.Linq
@using MSL.Common.Text;
@using MSL.Core.Configuration
@using MSL.Core.Data.Resources
@using MSL.Core.Entities.Vulnerability
@using MSL.Core.Process.Reporting;
@inherits HelperBaseTemplate<ReportTemplateData>
<?xml version="1.0" encoding="utf-8" ?>
<netsparker generated="@DateTime.Now.ToString()">
 <target>

  <url>@ReportingUtility.XmlShortEscape(Model.ScanProfile.Uri.AbsoluteUri)</url
 >

  <scantime>@Convert.ToInt32(ScanSettings.Instance.ElapsedTime.TotalSeconds)</s
 cantime>
 </target>
@{
 // Sort vulnerabilities based on their severity, Type, confirmation and
 certainty
 var sortedVulns = from IVulnerabilityView v in Model.Vulnerabilities
                   orderby v.Severity descending, v.Order ascending, v.Type
 ascending, v.IsConfirmed descending, v.Certainty descending
                   where v.Visibility != VulnerabilityVisibility.Hidden
                   select v;
```

```
foreach (var vuln in sortedVulns)
{
 if (vuln.Visibility != VulnerabilityVisibility.Hidden)
 {

   <vulnerability confirmed="@vuln.IsConfirmed.ToString()">
     <url>@ReportingUtility.XmlShortEscape(vuln.AbsoluteUri)</url>
     <type>@vuln.Type</type>
     <severity>@vuln.Severity.ToString()</severity>
     <certainty>@vuln.Certainty</certainty>

     @if (!string.IsNullOrEmpty(vuln.AttackParameterName))
     {

 <vulnerableparametertype>@ReportingUtility.XmlShortEscape(vuln.AttackParamete
rTypeName)</vulnerableparametertype>

 <vulnerableparameter>@ReportingUtility.XmlShortEscape(vuln.AttackParameterNam
e)</vulnerableparameter>

 <vulnerableparametervalue>@ReportingUtility.XmlShortEscape(vuln.AttackParamet
erValue)</vulnerableparametervalue>
     }

 <rawrequest>@ReportingUtility.XmlEscapeCharacterData(vuln.GetRawRequest())</r
awrequest>

 <rawresponse>@ReportingUtility.XmlEscapeCharacterData(vuln.GetFullResponse())
</rawresponse>
     <extrainformation>
       @foreach (var field in vuln.CustomFields)
       {
         <info
name="@field.Key">@ReportingUtility.XmlEscapeCharacterData(field.Value.HasMult
ipleValues ? string.Join(", ", field.Value.Values) : field.Value.Value)</info>
       }
     </extrainformation>

     @if (vuln.Classification != null)
     {

       <classification>
         <OWASP>@vuln.Classification.Owasp</OWASP>
         <WASC>@vuln.Classification.Wasc</WASC>
         <CWE>@vuln.Classification.Cwe</CWE>
         <CAPEC>@vuln.Classification.Capec</CAPEC>
         <PCI>@vuln.Classification.Pci</PCI>
         <PCI2>@vuln.Classification.Pci2</PCI2>
       </classification>
     }

     @if (vuln.VersionVulnerabilities.Any())
     {
       <knownvulnerabilities>
         @foreach (var implied in vuln.VersionVulnerabilities)
         {
           <knownvulnerability>
             <title>@implied.Title</title>
```

```
                    <severity>@implied.Severity</severity>
                 </knownvulnerability>
              }
           </knownvulnerabilities>
        }
     </vulnerability>
   }
  }
}
</netsparker>
```

This will generate an XML file which includes:

- All vulnerabilities
- Vulnerable Parameter and type (GET/POST)
- Vulnerability Details
- Confirmation Status
- Extra exploitation data
- Scan time
- Vulnerability severity etc...

You can add more details into the reports, customize them, or filter your reports with custom criteria.

Documentation

You can access an MSDN-style API documentation under the "Help" menu of Netsparker.

Defining the extension of the report

The name of the C# code file will be visible under the **Reporting** menu. When selected, the generated report will use the extension from the custom report file name.

For example:

- "Vulnerabilities List (XML).xml.cshtml" - File extension will be "xml"
- "Vulnerabilities List as Web Page.html.cshtml" - File extension will be "html"

Testing Reports

You don't need to restart Netsparker every time you change the source code of your report. After Netsparker adds it to the report menu once, all you need to do is run it again. If it fails to compile it'll let you know with an error message.

Security

The Reporting engine runs with current user's privileges. So don't run the report unless you trust the author of the report.

Command Line Arguments

Netsparker may be launched from the command line and its runtime behaviour may be influenced by passing command line parameter values that represent its configuration. This mode of operation can be used for automating scan operations, including complex scan sequences that target many domains.

Only a limited range of command line configuration options are supported, as detailed in the following table. However, more complex configurations may be established using Netsparker's scan profiles, which may then be referenced using the **/profile** command line configuration parameter.

/a, /auto	**Autopilot Mode** When this option is active (and when all other parameters are specified correctly), Netsparker will attempt to execute the designated scan, generate a report at the specified location, using the specified report template and then exit. The autopilot option is most commonly used when Netsparker is invoked as part of a batch process.
/p, /profile	**Profile Name** This option specifies the name of a pre-defined scan profile that will be applied during the scan. If no profile is specified, the default profile will be used. Profiles provide a powerful and convenient mechanism to pass complex sets of scan configuration data during command line execution. For example, a profile allows the following scan settings to be defined within Netsparker, and accessed as a single named configuration: • The target URL and the scope of the scan, relative to that URL • The range of selected security tests • The number of threads used for crawling and attacking • The active parsers and how their extracted links will be interpreted • The target back-end database Profiles also support advanced scanning concepts, such as the ability to manually import links or add HTTP requests.
/u, /url	**Target URL** This option specifies the address of the website to be scanned. It is mandatory unless a profile has been specified. If a profile has been specified and the URL configuration parameter is not specified, Netsparker will derive its target URL from the profile. If a profile has been specified and the URL configuration parameter is also specified, Netsparker will ignore the profile and derive its target URL from the target URL parameter.

/r, /report	**Report Path** This option specifies the file name or the file path to which the report output file will be saved. The full physical file path may be specified or, alternatively, if only the file name is specified, the report will be created in the folder from which Netsparker is launched. If the target path contains space characters, the path must be specified in double quotes. You may use this option multiple times to export multiple types of reports. When this option is used, the **/auto** parameter should also be used.
/rt, /reporttemplate	**Report Template Name** This option specifies the name of the template file used to generate report output. If not specified, reporting will default to using the first report template file discovered by Netsparker. You may use this option multiple times to export multiple types of reports.
/h, /help	**Help** This option shows Netsparker's command line help dialog.
/silent	**Silent Mode** This option specifies that Netsparker will not display any error messages. It should be used with scheduling and automated jobs.
/auth	**Authentication Credentials** This option specifies the combination of username, password and domain information that Netsparker will use with Basic, Digest and NTLM authentication. Valid examples are as follows: /auth username password /auth username password "host or domain" /auth username@domain password /auth host\username password If either the username, password or domain entries contain a space character, it must be specified in double quotes, as shown in the above samples. As an alternative to using the **/auth** configuration parameter, credentials may be configured as part of a scan profile and specified using the **/profile** option.

/lr, /logrequests	**Log HTTP Requests** When this option is active, all HTTP requests issued by Netsparker will be logged to a file named "HttpRequests.saz" which is located within the current scan directory. You can use Fiddler to view the contents of the log file.

Sample Usages:

Scan `http://test23.example.com` and save the report to `C:\reports\scan report.html`.

```
Netsparker /a /url http://test23.example.com /r "C:\reports\scan report.html"
```

Scan `http://test23.example.com` and save multiple types of reports. Generate "`scan report-1`" by using report template "`Detailed Scan Report`" and generate "`scan report-2`" by using report template "`OWASP Top Ten 2013 Report`".

```
Netsparker /a /url http://test23.example.com /r "C:\reports\scan report-1.html"
 /rt "Detailed Scan Report" /r "C:\reports\scan report-2.html" /rt "OWASP Top
 Ten 2013 Report"
```

Scan `http://test23.example.com` by using NTLM authentication.

```
Netsparker /a /url http://test23.example.com /auth john.doe "secret password"
 example.com
```

Launch a new scan parameter with a custom profile and URL:

```
Netsparker /url http://test23.example.com /p LFI
```

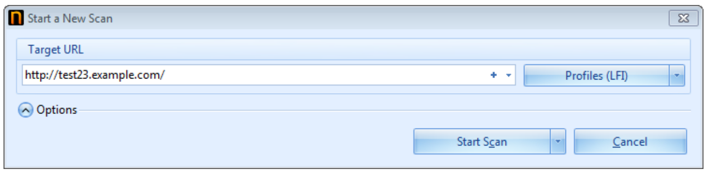


Figure 47: Netsparker Launched from Command Line with Custom Profile and URL

Encoder

Netsparker incorporates a built-in encoder tool, which supports URL, HTML, Base64, UTF7, MD5, SHA1, SHA256, SHA512 , SQL Server Character and ROT13 encoding and decoding.

It is typically used when reviewing the content of HTTP responses and is accessible via the Encoder tool window:

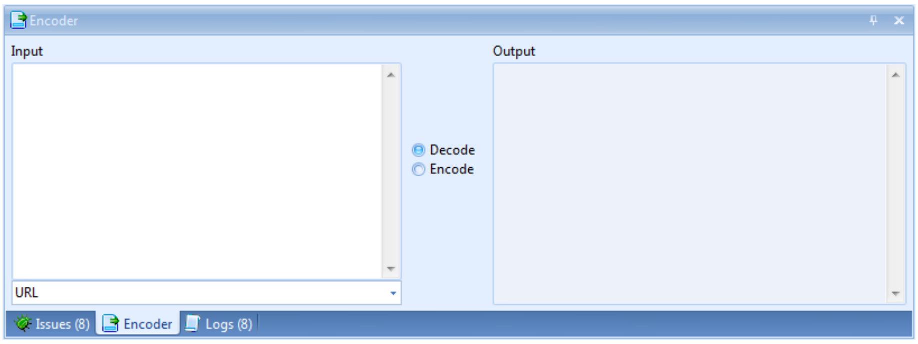


Figure 48: Encoder Tool Window

Exploitation

Fundamentals

Netsparker's integrated exploitation engine allows you to exploit certain vulnerabilities directly from the user interface. This does not require exploitation expertise and can be used easily. Currently Netsparker supports exploitation for the following vulnerabilities:

- SQL Injection
  - Error Based SQL Injection
  - Boolean SQL Injection
- LFI (Local File Inclusions)

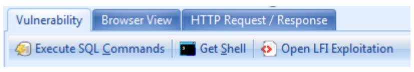
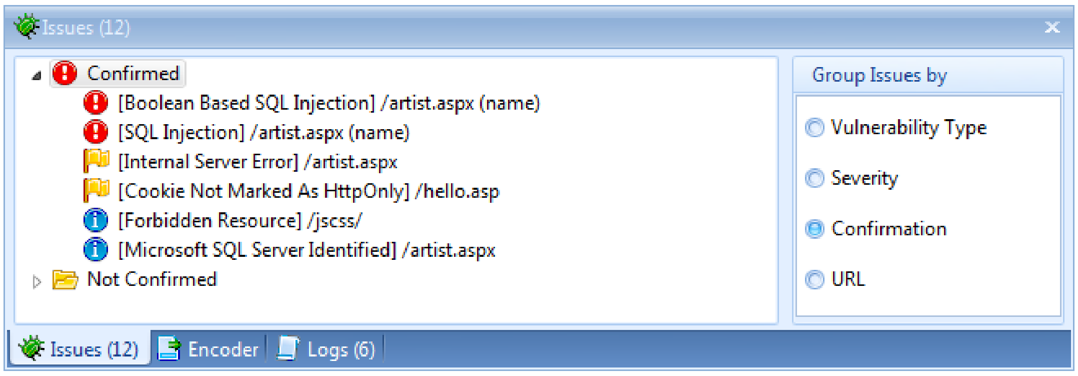


Figure 49: Exploitation Toolbar

Netsparker will highlight exploitation buttons based on the selected vulnerability from the "Issues Panel" or "Sitemap". Netsparker will not highlight buttons if the exploitation is not possible. For example if the database user in an SQL Injection vulnerability does not have priviliges to execute commands, the **Get Shell** button will not get highlighted.

Exploiting SQL Injections

You can exploit the Error Based or Boolean Based SQL Injection vulnerabilities identified in the web application, and run custom SQL queries in the application's database via Netsparker's SQL Injection panel.



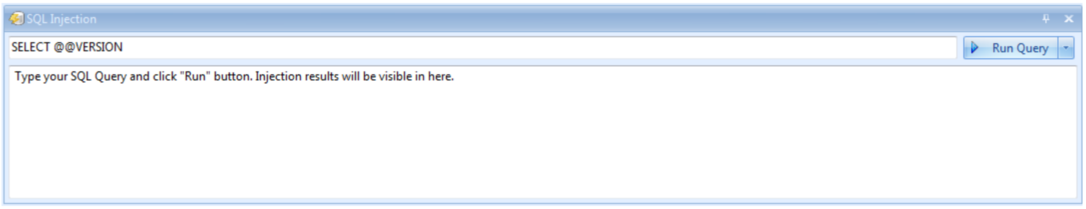
Confirmed
- [Boolean Based SQL Injection] /artist.aspx (name)
- [SQL Injection] /artist.aspx (name)
- [Internal Server Error] /artist.aspx
- [Cookie Not Marked As HttpOnly] /hello.asp
- [Forbidden Resource] /jscss/
- [Microsoft SQL Server Identified] /artist.aspx

Not Confirmed

Vulnerability Type
Severity
[✓ Confirmation]
URL

Figure 50: SQL Injection issues in the Issues Panel

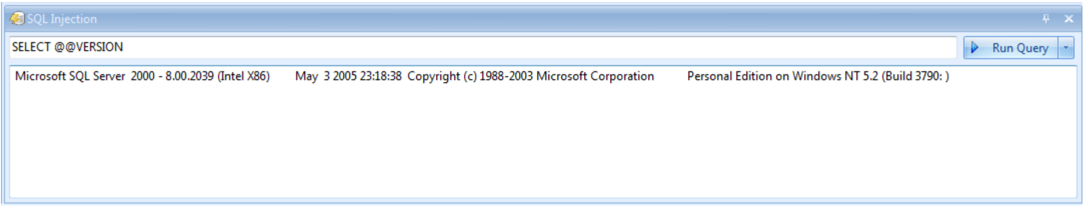
From the issues in the **Issues** panel, choose a confirmed "SQL Injection" or "Boolean Based SQL Injection" issue, click the **Execute SQL Commands** button and the **SQL Injection** panel will appear where you can run custom SQL queries.



```
SELECT @@VERSION
Type your SQL Query and click "Run" button. Injection results will be visible in here.
```

Figure 51: Running Custom SQL Queries in SQL Injection Panel

Afterwards, you can type an SQL query and run the query with the **Run Query** button. The response of the query will be displayed in the panel.



```
SELECT @@VERSION
Microsoft SQL Server  2000 - 8.00.2039 (Intel X86)      May  3 2005 23:18:38  Copyright (c) 1988-2003 Microsoft Corporation       Personal Edition on Windows NT 5.2 (Build 3790: )
```

Figure 52: Sample Custom SQL Query Output

Getting a Reverse Shell

When Netsparker identifies and confirms an SQL Injection vulnerability, Netsparker checks the database user automatically. If the user has admin rights and can access the file system, they can access the command prompt on the target system. The panel reverse shell dialog will pop up when you click the **Get Shell** button.

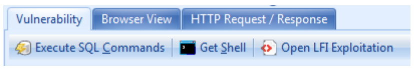
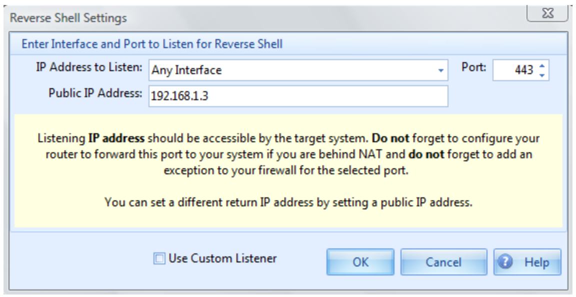


Figure 53: Get Shell Button



Reverse Shell Settings

Enter Interface and Port to Listen for Reverse Shell

IP Address to Listen: Any Interface

Port: 443

Public IP Address: 192.168.1.3

Listening **IP address** should be accessible by the target system. **Do not** forget to configure your router to forward this port to your system if you are behind NAT and **do not** forget to add an exception to your firewall for the selected port.

You can set a different return IP address by setting a public IP address.

☐ Use Custom Listener

OK Cancel Help

Figure 54: Reverse Shell Dialog

Reverse shell will send a small executable to the target system, and when this executable gets executed by Netsparker it will connect back to your system. When it connects, you can run commands in the target system. To enable this, you should configure reverse shell settings in this dialog. You can either use your computer to listen for a shell or you can use a "Custom Listener".

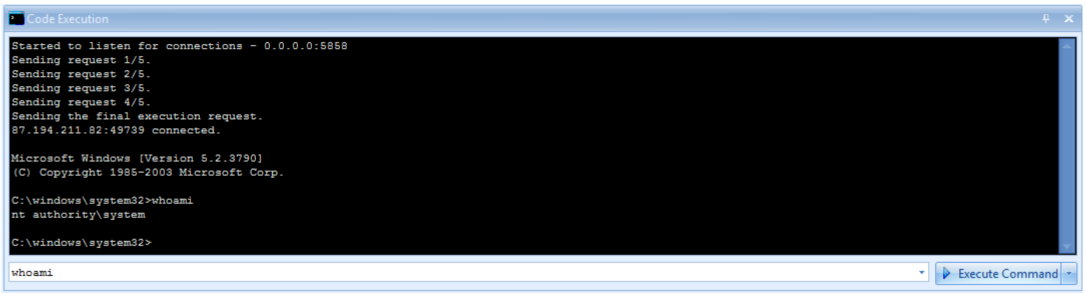
You should choose **IP Address to Listen** so you can receive the shell. If you are not sure about addresses or if you are connected to a single network, we recommend using the "**Any Interface"** option.

In the **Public IP Address** field, type your public IP address. This is the IP Address where the transferred executable will connect to. Therefore, ensure that this IP address is reachable by the target. If you access the target system over the internet this IP address should be your public IP address. You can find your public IP address via websites such as www.whatismyipaddress.com. If you need to enter your public address, do not forget to configure port forwarding in your router. The www.portforward.com website can guide you on how to configure port forwarding in your router.

If the target is in the local network you can use your local IP Address.

In the **Port** field, you should specify the port number on your computer over which the reverse shell connection will be made; since this port will be used for listening, the required firewall configuration should be completed.

After filling in the relevant fields, click the **OK** button to start sending the requests required to get reverse shell.



```
Started to listen for connections - 0.0.0.0:5858
Sending request 1/5.
Sending request 2/5.
Sending request 3/5.
Sending request 4/5.
Sending the final execution request.
87.194.211.82:49739 connected.

Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\windows\system32>whoami
nt authority\system

C:\windows\system32>
```

```
whoami
```

Figure 55: Sample Reverse Shell Session

After a connection is made successfully, you can run commands on the opposite system from the `Code Execution` panel.

Using a Custom Listener

You can activate the `Use Custom Listener` option to use a listener of your own choice instead of Netsparker's connection listener. This is quite useful if you already have a public internet facing system in place. Just configure a tool such as Netcat to listen to the given port and set `Public IP Address` to that system's IP Address. In this scenario you won't see the reverse shell in Netsparker's screen but you will be able to access the shell from the Netcat. Use the following command to listen port 443 with Netcat:

```
nc -vv -l -p 443
```

Netsparker will use a standard reverse shell, therefore you can use a tool such Netcat or any other similar tool.

Exploiting LFI

It is possible to exploit the "Local File Inclusion" vulnerabilities identified in the target application; Netsparker can download files on the system and can access the source code of the application.

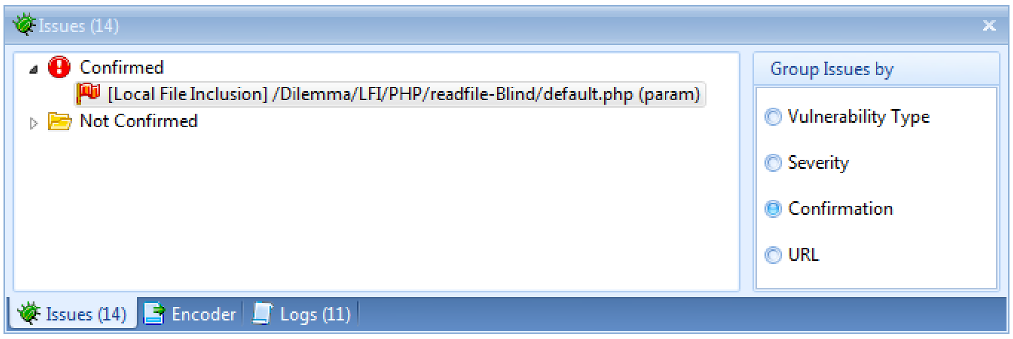
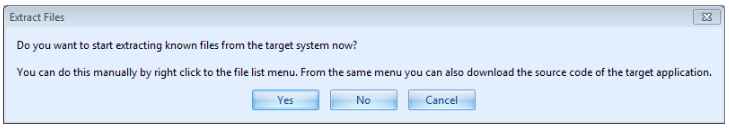


Figure 56: Local File Inclusion Vulnerability in Issues Panel

After an issue of **Local File Inclusion** type is selected from the Issue list, you can click the **Open LFI Exploitation** button to access the panel that will enable you to read files on the system and access the source code of the application.



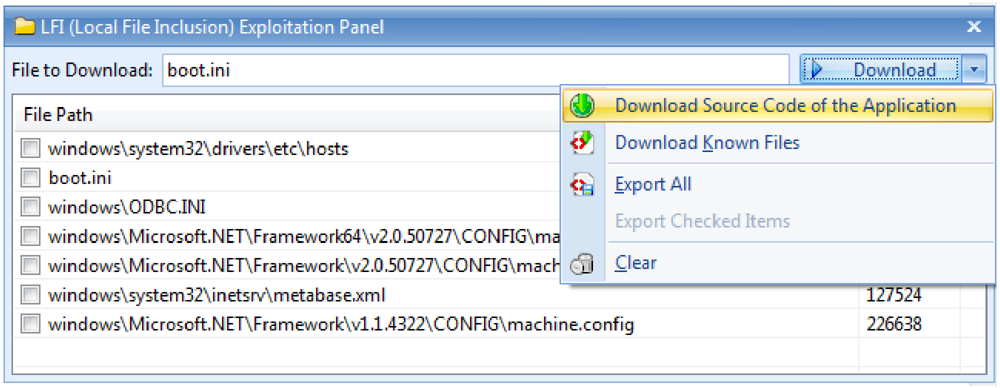
Extract Files

Do you want to start extracting known files from the target system now?

You can do this manually by right click to the file list menu. From the same menu you can also download the source code of the target application.

Figure 57: LFI Exploitation Confirmation

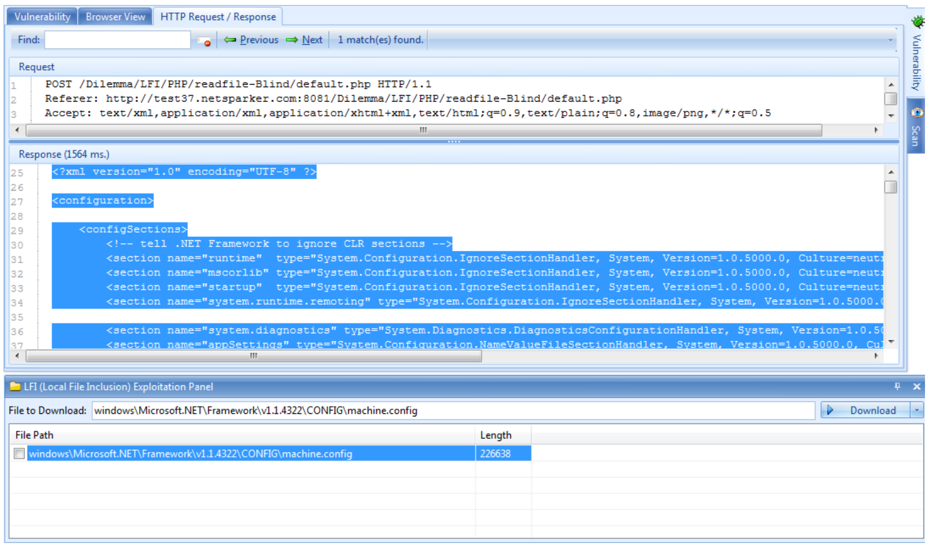
It is possible to download some known files automatically from the target system.


LFI (Local File Inclusion) Exploitation Panel
File to Download: boot.ini
Download
Download Source Code of the Application
Download Known Files
Export All
Export Checked Items
Clear

File Path	
windows\system32\drivers\etc\hosts	
boot.ini	
windows\ODBC.INI	
windows\Microsoft.NET\Framework64\v2.0.50727\CONFIG\ma	
windows\Microsoft.NET\Framework\v2.0.50727\CONFIG\mach	
windows\system32\inetsrv\metabase.xml	127524
windows\Microsoft.NET\Framework\v1.1.4322\CONFIG\machine.config	226638

Figure 58: Download Files

Even if you did not select automatic download, you can use the **Download** button to re-download known files on the system and source code of the application.



Request

```
POST /Dilemma/LFI/PHP/readfile-Blind/default.php HTTP/1.1
Referer: http://test37.netsparker.com:8081/Dilemma/LFI/PHP/readfile-Blind/default.php
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
```

Response (1564 ms.)

```
<?xml version="1.0" encoding="UTF-8" ?>

<configuration>

    <configSections>
        <!-- tell .NET Framework to ignore CLR sections -->
        <section name="runtime"  type="System.Configuration.IgnoreSectionHandler, System, Version=1.0.5000.0, Culture=neut
        <section name="mscorlib" type="System.Configuration.IgnoreSectionHandler, System, Version=1.0.5000.0, Culture=neut
        <section name="startup"  type="System.Configuration.IgnoreSectionHandler, System, Version=1.0.5000.0, Culture=neut
        <section name="system.runtime.remoting" type="System.Configuration.IgnoreSectionHandler, System, Version=1.0.5000.

        <section name="system.diagnostics" type="System.Diagnostics.DiagnosticsConfigurationHandler, System, Version=1.0.5
        <section name="appSettings" type="System.Configuration.NameValueFileSectionHandler, System, Version=1.0.5000.0, Cu
```

LFI (Local File Inclusion) Exploitation Panel

File to Download: windows\Microsoft.NET\Framework\v1.1.4322\CONFIG\machine.config

File Path	Length
windows\Microsoft.NET\Framework\v1.1.4322\CONFIG\machine.config	226638

Figure 59: LFI Exploation Downloaded File View

You can also type the file name manually and download a file that you know exists on the system in the LFI Exploitation panel. After you click the **Download** button, you can see the contents of the file in question when the download is complete.

Application Settings

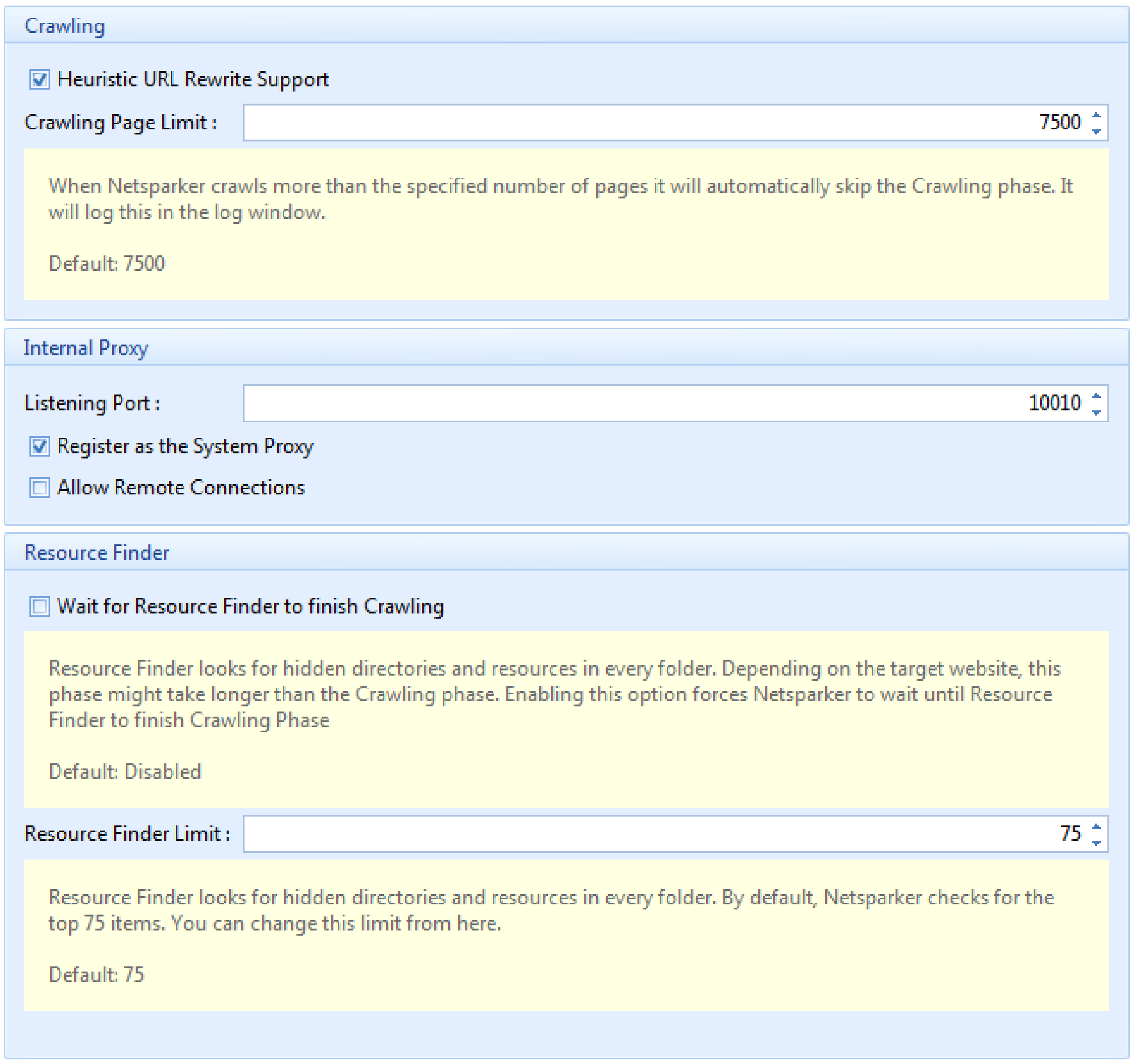
Although Netsparker is designed to be easy to use and ready for work from the moment you install it, it is also very flexible and able to adapt to a broad range of more complex runtime scenarios.

By overriding one or more the default settings, Netsparker's configuration may be tuned to accommodate specific operational needs. This section documents Netsparker's application settings and describes how to configure these settings to achieve specific goals.

Scan Settings

Crawling

Crawling settings influence how Netsparker acts during the crawling phase. These settings define how it will crawl the target domain and discover pages to attack.



Crawling

☑ Heuristic URL Rewrite Support

Crawling Page Limit : 7500

When Netsparker crawls more than the specified number of pages it will automatically skip the Crawling phase. It will log this in the log window.

Default: 7500

Internal Proxy

Listening Port : 10010

☑ Register as the System Proxy

☐ Allow Remote Connections

Resource Finder

☐ Wait for Resource Finder to finish Crawling

Resource Finder looks for hidden directories and resources in every folder. Depending on the target website, this phase might take longer than the Crawling phase. Enabling this option forces Netsparker to wait until Resource Finder to finish Crawling Phase

Default: Disabled

Resource Finder Limit : 75

Resource Finder looks for hidden directories and resources in every folder. By default, Netsparker checks for the top 75 items. You can change this limit from here.

Default: 75

Heuristic URL Rewrite Support enables Netsparker to detect the use of URL rewriting on the target web application. When this option is enabled, crawling efficiency is improved because Netsparker aggregates multiple URLs that represent the same underlying URL. When this option is disabled, Netsparker will unconditionally crawl all detected URLs.

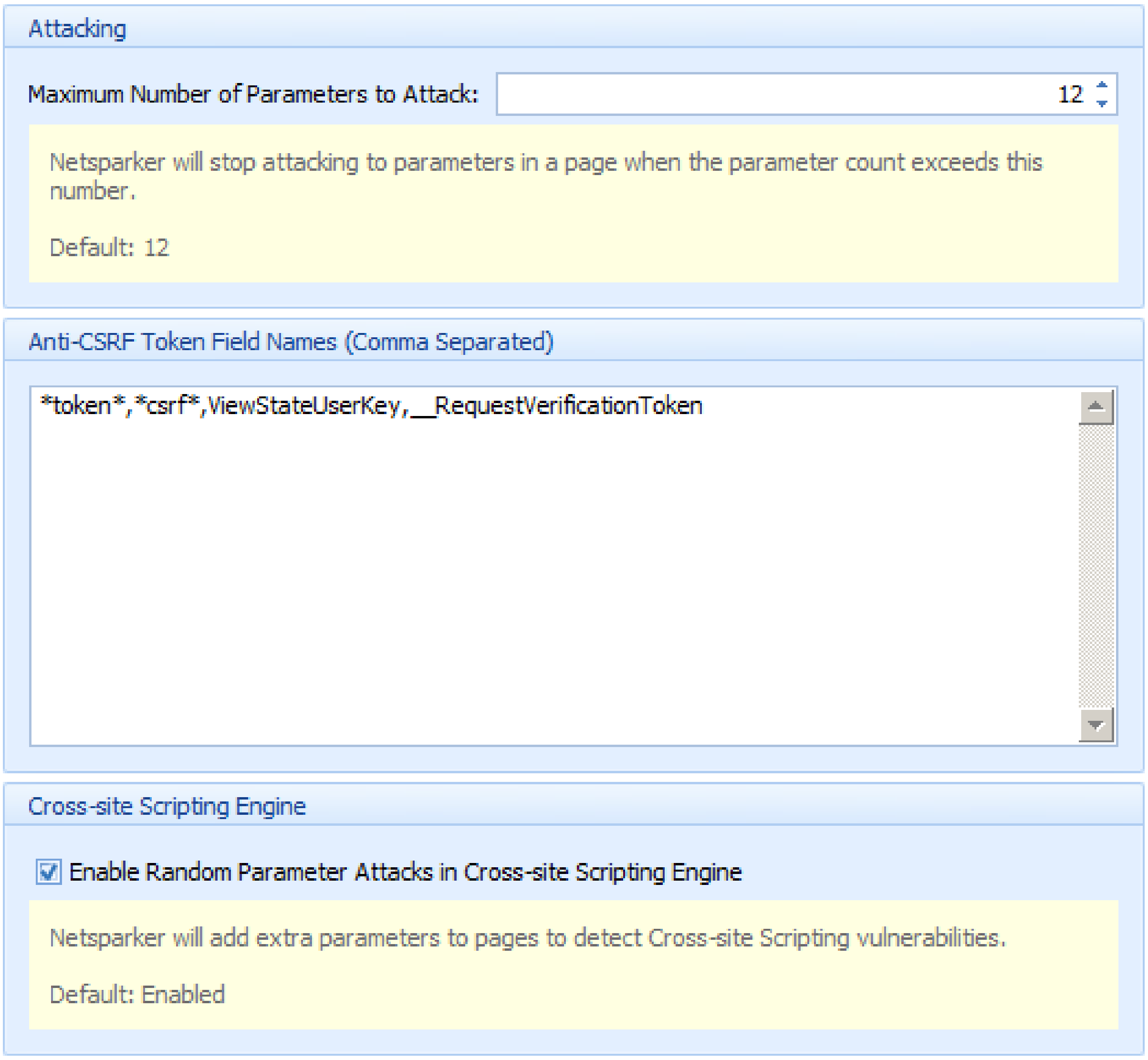
Crawling Page Limit specifies the maximum number of URLs that Netsparker will crawl.

The internal proxy settings govern how Netsparker's built-in proxy operates (when Netsparker is scanning using manual crawl mode). The **Listening Port** setting specifies the port number on which the internal proxy listens. When configured to **Register as the System Proxy**, Netsparker's proxy inserts itself within the local proxy chain, which means that it will capture requests from all applications that use the system proxy. The **Allow Remote Connections** setting determines whether Netsparker's proxy captures requests from other computers.

Netsparker uses a pre-defined name list to attempt to discover web application resources in each of the directories that crawls. This is a time-consuming process that can potentially exceed the duration of crawling. When the **Wait for Resource Finder to finish Crawling** setting is enabled, Netsparker will delay completion of the crawling phase until resource detection is complete. To reduce detection time, testing may be limited to a specified number of entries from the name list, as configured in the **Resource Finder Limit** setting.

Attacking

Attacking settings determine how Netsparker acts during the attacking phase. These settings define the behaviour of its attacking operations.



Attacking

Maximum Number of Parameters to Attack: 12

Netsparker will stop attacking to parameters in a page when the parameter count exceeds this number.

Default: 12

Anti-CSRF Token Field Names (Comma Separated)

token,*csrf*,ViewStateUserKey,__RequestVerificationToken

Cross-site Scripting Engine

[✓] Enable Random Parameter Attacks in Cross-site Scripting Engine

Netsparker will add extra parameters to pages to detect Cross-site Scripting vulnerabilities.

Default: Enabled

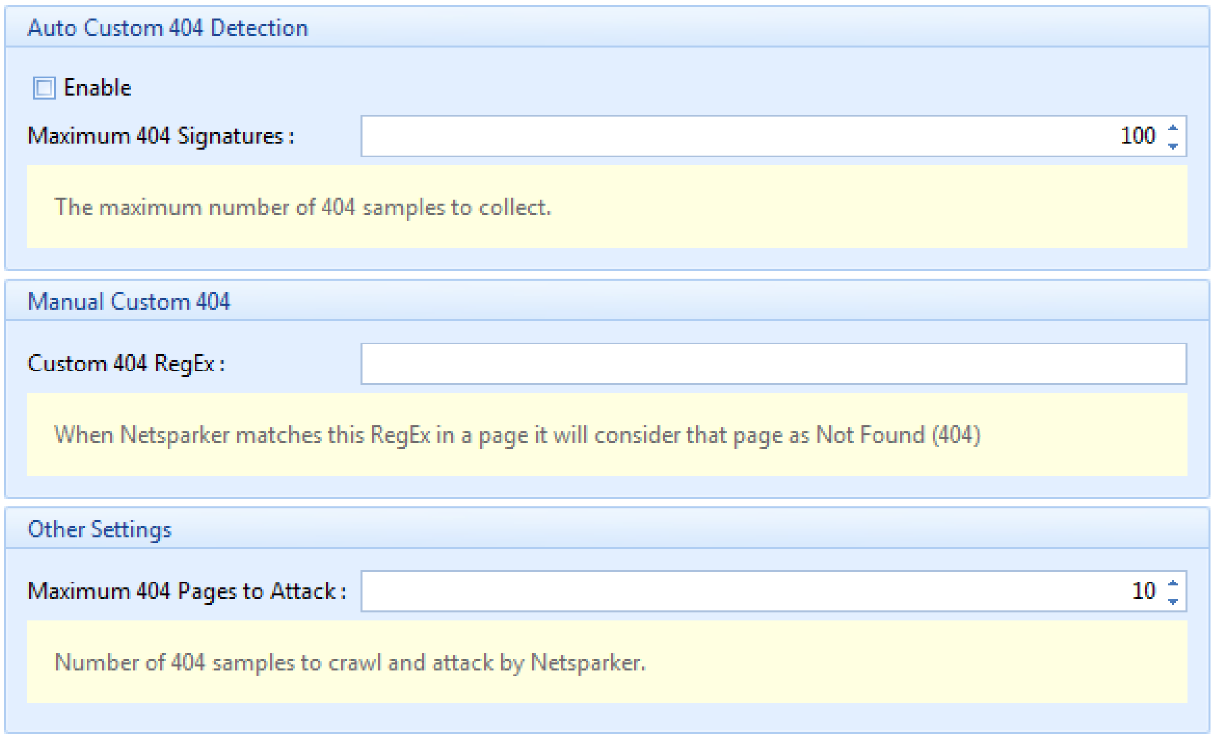
The **Maximum Number of Parameters to Attack** setting limits the number of parameters that will be attacked on any given page. This may be used to improve attacking performance for web applications with pages that contain an unusually large number of parameters.

Netsparker's anti-CSRF support relies on a pre-defined list of token patterns, defined in **Anti-CSRF Token Field Names**, that are used to identify token parameters. The standard configuration includes patterns that will match many of the common token formats. New patterns may be added by extending the existing comma-separated list. Patterns may use the * character to represent wildcard matches.

The cross-site scripting configuration includes a single setting that influences XSS attacking. When **Enable Random Parameter Attacks in Cross-site Scripting Engine** is checked, Netsparker will add extra random parameters to pages in order to attempt detection of XSS vulnerabilities.

Custom 404

Custom 404 settings determine how Netsparker acts when it detects a custom 404 error page.


Auto Custom 404 Detection

☐ Enable

Maximum 404 Signatures : 100

The maximum number of 404 samples to collect.

Manual Custom 404

Custom 404 RegEx :

When Netsparker matches this RegEx in a page it will consider that page as Not Found (404)

Other Settings

Maximum 404 Pages to Attack : 10

Number of 404 samples to crawl and attack by Netsparker.

By default, Netsparker uses its automatic detection of custom 404 pages, which entails requesting non-existent pages and collecting their responses as samples of custom 404 page content. Subsequently, a similarity algorithm is used to compare these samples against other received HTTP responses to determine whether any given response represents a custom 404 page.

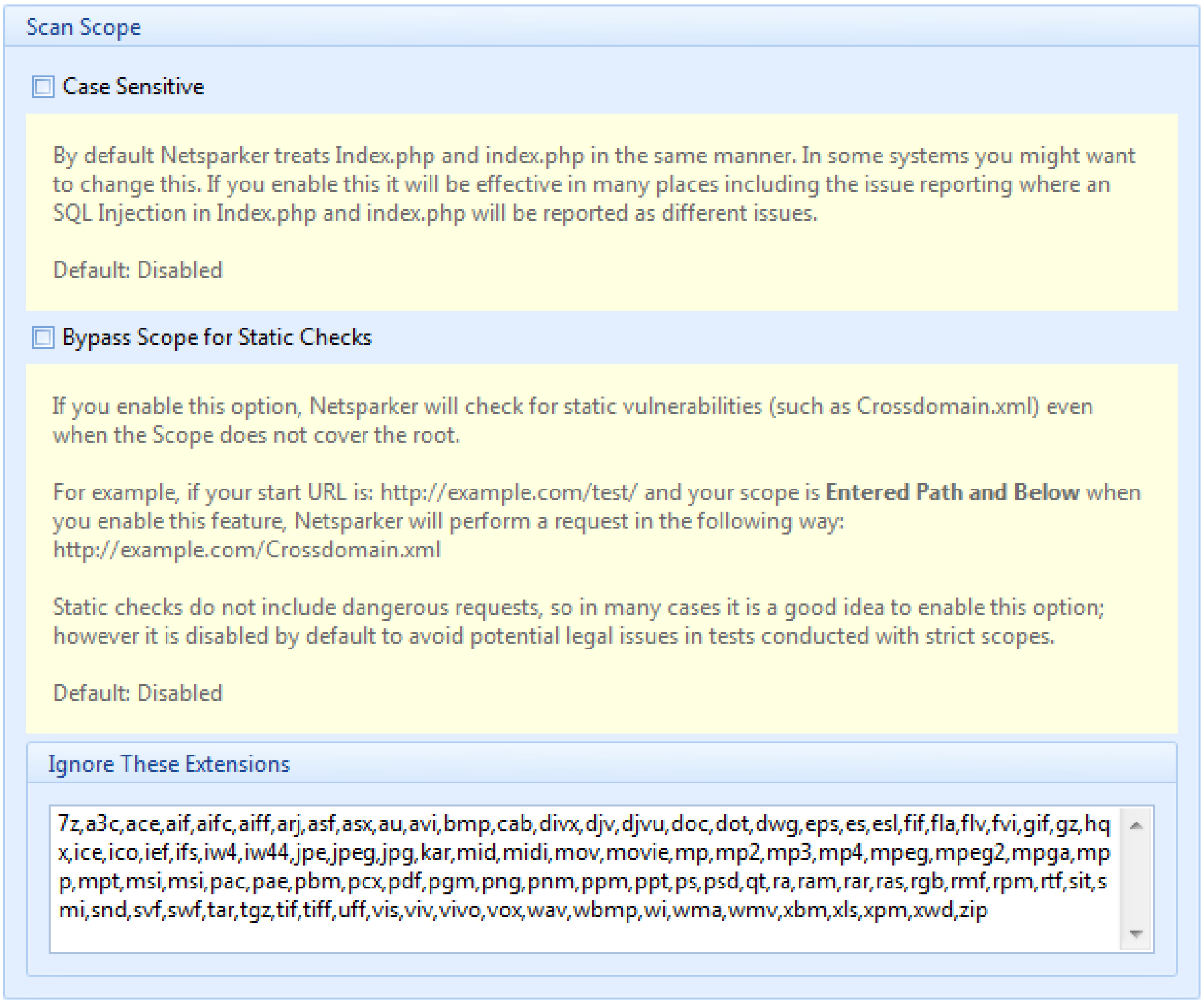
When the **Enabled** option is checked, Netsparker will use its built-in mechanism to collect custom 404 page samples. The number of collected samples is configurable in the **Maximum 404 Signatures** setting.

Alternatively (or in addition to Netsparker's automatic custom 404 page detection) it is also possible to specify the **Custom 404 Regex** setting, which represents a regular expression matching pattern for custom 404 pages.

To optimize Netsparker's attack phase, the **Maximum 404 Pages to Attack** setting limits the number of identified custom 404 pages that will be attacked.

Scope

Scan Scope settings influence how specific aspects of the scanning process are executed, within the context of the defined scan scope.


Scan Scope

☐ Case Sensitive

By default Netsparker treats Index.php and index.php in the same manner. In some systems you might want to change this. If you enable this it will be effective in many places including the issue reporting where an SQL Injection in Index.php and index.php will be reported as different issues.

Default: Disabled

☐ Bypass Scope for Static Checks

If you enable this option, Netsparker will check for static vulnerabilities (such as Crossdomain.xml) even when the Scope does not cover the root.

For example, if your start URL is: http://example.com/test/ and your scope is **Entered Path and Below** when you enable this feature, Netsparker will perform a request in the following way:
http://example.com/Crossdomain.xml

Static checks do not include dangerous requests, so in many cases it is a good idea to enable this option; however it is disabled by default to avoid potential legal issues in tests conducted with strict scopes.

Default: Disabled

Ignore These Extensions

```
7z,a3c,ace,aif,aifc,aiff,arj,asf,asx,au,avi,bmp,cab,divx,djv,djvu,doc,dot,dwg,eps,es,esl,fif,fla,flv,fvi,gif,gz,hq
x,ice,ico,ief,ifs,iw4,iw44,jpe,jpeg,jpg,kar,mid,midi,mov,movie,mp,mp2,mp3,mp4,mpeg,mpeg2,mpga,mp
p,mpt,msi,msi,pac,pae,pbm,pcx,pdf,pgm,png,pnm,ppm,ppt,ps,psd,qt,ra,ram,rar,ras,rgb,rmf,rpm,rtf,sit,s
mi,snd,svf,swf,tar,tgz,tif,tiff,uff,vis,viv,vivo,vox,wav,wbmp,wi,wma,wmv,xbm,xls,xpm,xwd,zip
```

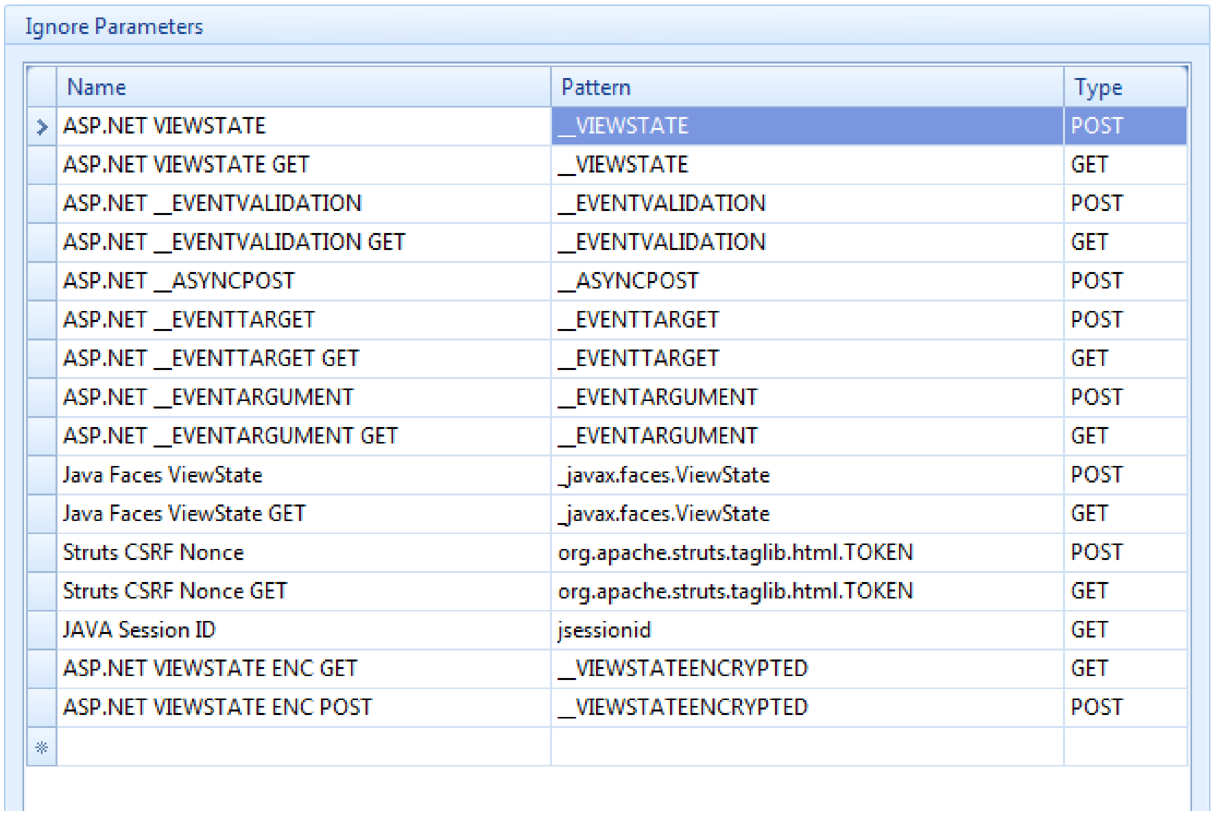
The **Case Sensitive** setting causes Netsparker to treat URLs as being case-sensitive, which may result in many additional requests being issued, and any issues arising from these requests will be reported separately for each of their case variants.

The **Bypass Scope for Static Checks** setting determines whether Netsparker is allowed to perform static checks (which are non-destructive) outside the scope defined within the scan configuration.

The **Ignore These Extensions** setting contains a comma-separated list of file extension names that are ignored by Netsparker.

Ignored Parameters

The Ignored Parameters table defines a list of well-known HTTP parameter names that will not be attacked by Netsparker.



Ignore Parameters

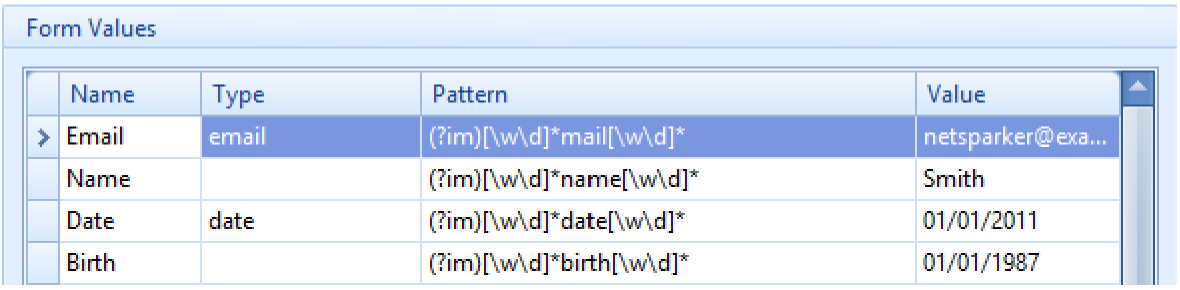
Name	Pattern	Type
ASP.NET VIEWSTATE	__VIEWSTATE	POST
ASP.NET VIEWSTATE GET	__VIEWSTATE	GET
ASP.NET __EVENTVALIDATION	__EVENTVALIDATION	POST
ASP.NET __EVENTVALIDATION GET	__EVENTVALIDATION	GET
ASP.NET __ASYNCPOST	__ASYNCPOST	POST
ASP.NET __EVENTTARGET	__EVENTTARGET	POST
ASP.NET __EVENTTARGET GET	__EVENTTARGET	GET
ASP.NET __EVENTARGUMENT	__EVENTARGUMENT	POST
ASP.NET __EVENTARGUMENT GET	__EVENTARGUMENT	GET
Java Faces ViewState	_javax.faces.ViewState	POST
Java Faces ViewState GET	_javax.faces.ViewState	GET
Struts CSRF Nonce	org.apache.struts.taglib.html.TOKEN	POST
Struts CSRF Nonce GET	org.apache.struts.taglib.html.TOKEN	GET
JAVA Session ID	jsessionid	GET
ASP.NET VIEWSTATE ENC GET	__VIEWSTATEENCRYPTED	GET
ASP.NET VIEWSTATE ENC POST	__VIEWSTATEENCRYPTED	POST

The **Name** column in this table provides a user-friendly identification for each defined parameter, whilst the **Pattern** column specifies the actual parameter name affected by the line entry.

The **Type** column identifies the HTTP method type for which a given named parameter will be ignored. Note that, in order to ignore a named parameter for both HTTP GET and POST requests, separate GET and POST entries are required.

Form Values

The Form Values table defines a list of HTML form field names for which specified values will be used during attacking.



Form Values

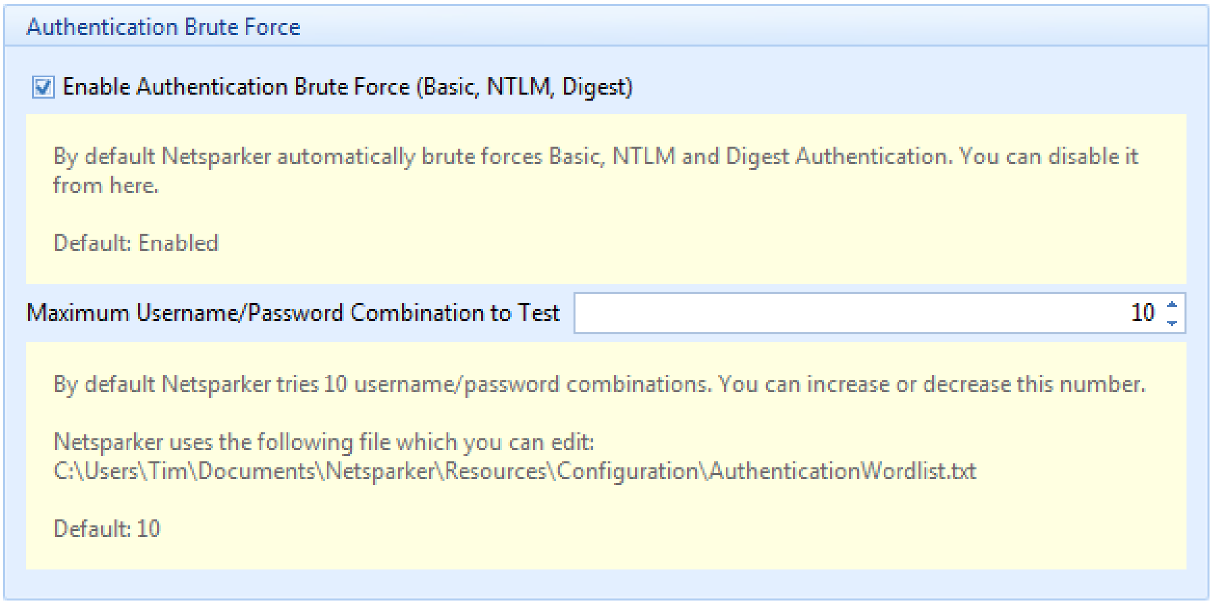
Name	Type	Pattern	Value
Email	email	(?im)[\w\d]*mail[\w\d]*	netsparker@exa...
Name		(?im)[\w\d]*name[\w\d]*	Smith
Date	date	(?im)[\w\d]*date[\w\d]*	01/01/2011
Birth		(?im)[\w\d]*birth[\w\d]*	01/01/1987

The Name column in this table provides a user-friendly identification for each defined form value. The Type column defines the input type of a form field, whilst the Pattern column specifies a regular expression that matches the name of a form field. Netsparker first checks for the input type if it fails then it checks for the pattern. The Value column defines the value that will be applied to form fields that match the specified item.

#DEFAULT# is a special name, whose value will be used when Netsparker needs to supply a value for an empty form field. It is recommended to use numeric values for this definition, as that's the most generic form of input that will be most likely to bypass checks.

Brute Force

Brute Force settings influence how Netsparker will attempt to perform brute-force authentication.



Authentication Brute Force

☑ Enable Authentication Brute Force (Basic, NTLM, Digest)

By default Netsparker automatically brute forces Basic, NTLM and Digest Authentication. You can disable it from here.

Default: Enabled

Maximum Username/Password Combination to Test: 10

By default Netsparker tries 10 username/password combinations. You can increase or decrease this number.

Netsparker uses the following file which you can edit:
C:\Users\Tim\Documents\Netsparker\Resources\Configuration\AuthenticationWordlist.txt

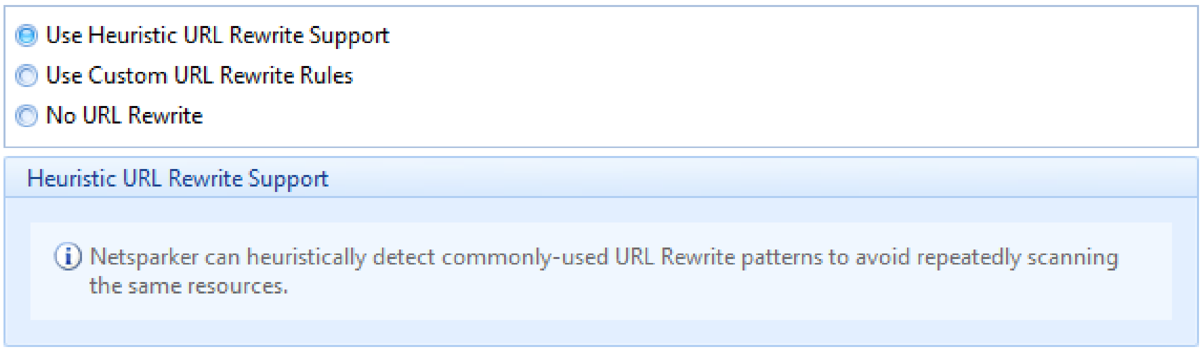
Default: 10

The Enable Authentication Brute Force setting determines whether Netsparker will attempt to authenticate to your web application by brute force.

The Maximum Username/Password Combination to Test settings determines how many entries from its pre-defined authentication name list to use.

URL Rewrite

URL Rewrite settings determine how Netsparker acts on websites that use URL Rewrite/RESTful techniques. (i.e. http://example.com/books/mobydick instead of http://example.com/?cat=books&title=mobydick)



● Use Heuristic URL Rewrite Support
○ Use Custom URL Rewrite Rules
○ No URL Rewrite

Heuristic URL Rewrite Support

Netsparker can heuristically detect commonly-used URL Rewrite patterns to avoid repeatedly scanning the same resources.

Select **Use Heuristic URL Rewrite Support** to heuristically detect common URL Rewrite patterns to avoid repeatedly scanning the same resources.

If you know that the target website is using a URL Rewrite you should choose **Custom URL Rewrite** as it'll always perform better than Heuristic URL Rewrite.

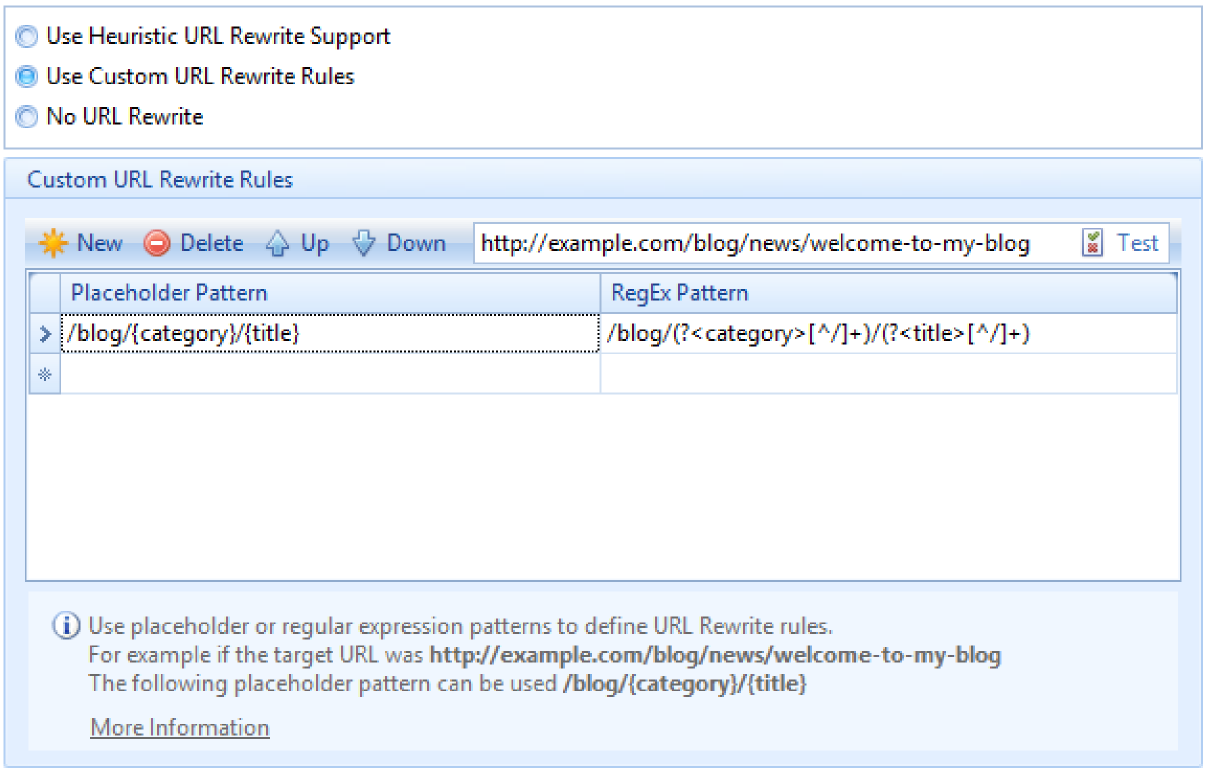
Select **Use Custom URL Rewrite Rules** to define your own custom rules.

Select **No URL Rewrite** to disable URL Rewrite completely. When this option is selected, Netsparker will unconditionally crawl all detected URLs.

Custom URL Rewrite Rules

Netsparker can recognize query parameters automatically, but if parameters are located in the path fragments you need to write custom URL Rewrite rules for these parameters.

You can define a rule by entering either a placeholder or a regular expression pattern.



Use Heuristic URL Rewrite Support
[✓ Use Custom URL Rewrite Rules]
No URL Rewrite

Custom URL Rewrite Rules

http://example.com/blog/news/welcome-to-my-blog

Placeholder Pattern	RegEx Pattern
/blog/{category}/{title}	/blog/(?<category>[^/]+)/(?<title>[^/]+)

Use placeholder or regular expression patterns to define URL Rewrite rules.
For example if the target URL was **http://example.com/blog/news/welcome-to-my-blog**
The following placeholder pattern can be used **/blog/{category}/{title}**

More Information

A placeholder pattern can be defined with a parameter name wrapped between curly braces. You can define multiple parameters, but each parameter must have a unique name. Parameter names can only contain letters.

Parameters names are only used for your reference, they won't effect anything during the scan.

For example if the target URL was **http://example.com/blog/news/welcome-to-my-blog**

The following placeholder pattern can be used **/blog/{category}/{title}**

"news" path segment will be identified as category parameter and "welcome-to-my-blog" path segment will be identified as title parameter.

If a regular expression pattern is entered, each captured regular expression group will be identified as a parameter.

For example if the target URL was **http://example.com/books/hobbit**

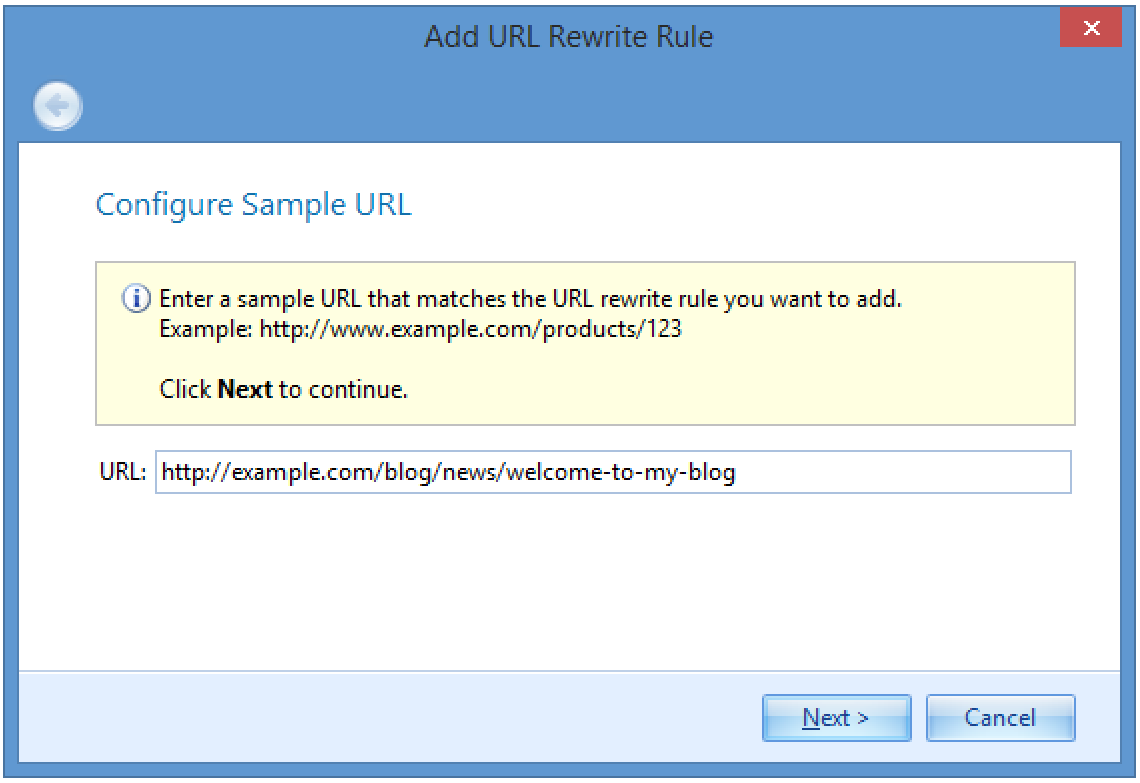
The following regular expression pattern can be used **/books/(?<title>[^/]+)**

Create A Rule by Using The Wizard

You can create a rule by clicking the **New** button as well. This will open a wizard that will guide you to create a placeholder pattern.

Step 1: Configure Sample URL

Enter a sample URL that matches the URL rewrite rule you want to add. Click the **Next** button to proceed to the next step of the wizard.



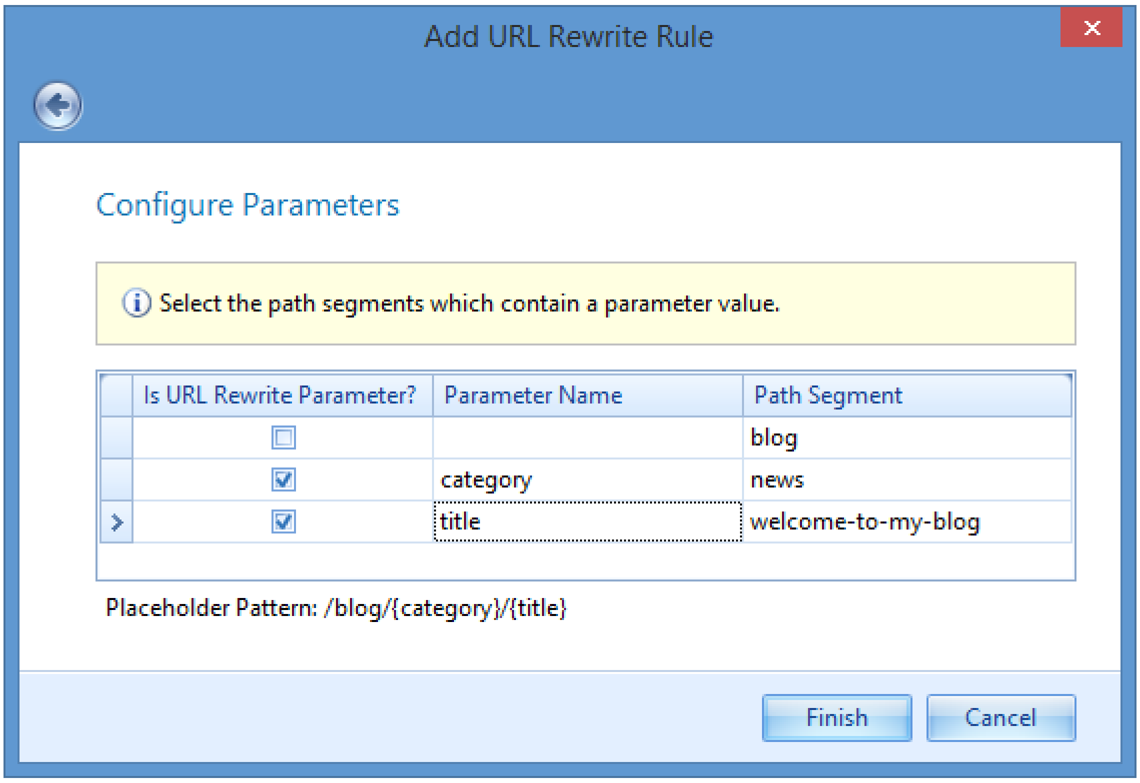
Enter a sample URL that matches the URL rewrite rule you want to add.
Example: http://www.example.com/products/123

Click **Next** to continue.

URL: http://example.com/blog/news/welcome-to-my-blog

Step 2: Configure Parameters

Select the path segments which contain a parameter value. Click the **Finish** button to complete the wizard.



Configure Parameters

Select the path segments which contain a parameter value.

Is URL Rewrite Parameter?	Parameter Name	Path Segment
☐		blog
☑	category	news
☑	title	welcome-to-my-blog

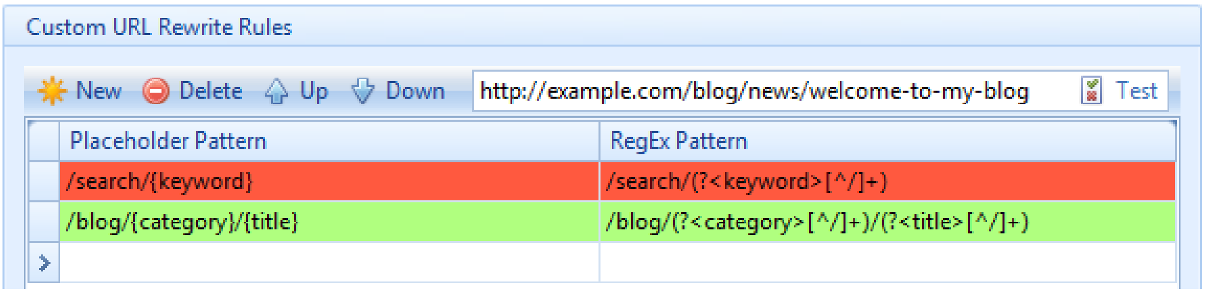
Placeholder Pattern: /blog/{category}/{title}

Rule Evaluation

Rules are evaluated in sequence from the first rule to the last rule. When a match occurs, no more rules are evaluated. Make sure to enter your rules in the correct order.

Testing Your Rules

To test your rules, enter a test URL and click the **Test** button. Matched rule will be highlighted in green and unmatched rules are highlighted in red.



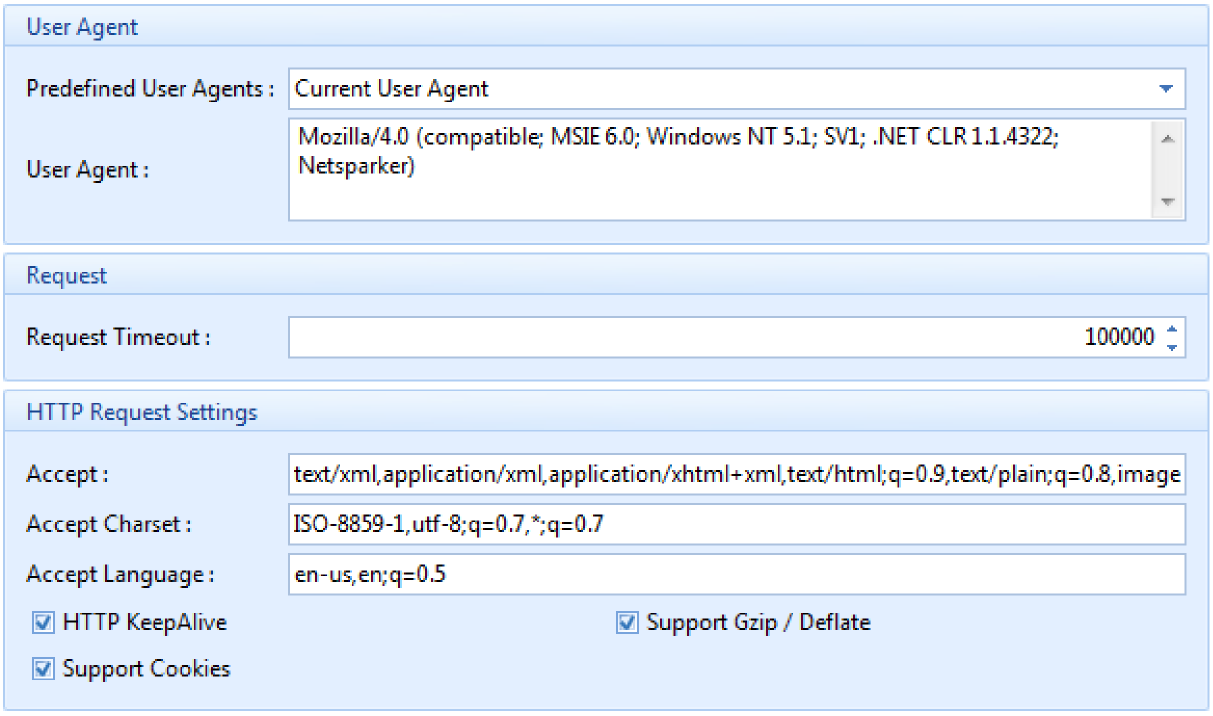
http://example.com/blog/news/welcome-to-my-blog

Placeholder Pattern	RegEx Pattern
/search/{keyword}	/search/(?<keyword>[^/]+)
/blog/{category}/{title}	/blog/(?<category>[^/]+)/(?<title>[^/]+)

HTTP Settings

HTTP Request

HTTP Request settings influence how Netsparker performs HTTP requests during crawling and attacking.


User Agent

Predefined User Agents :	Current User Agent
User Agent :	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; Netsparker)

Request

Request Timeout :	100000

HTTP Request Settings

Accept :	text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image
Accept Charset :	ISO-8859-1,utf-8;q=0.7,*;q=0.7
Accept Language :	en-us,en;q=0.5

- ✓ HTTP KeepAlive
- ✓ Support Gzip / Deflate
- ✓ Support Cookies

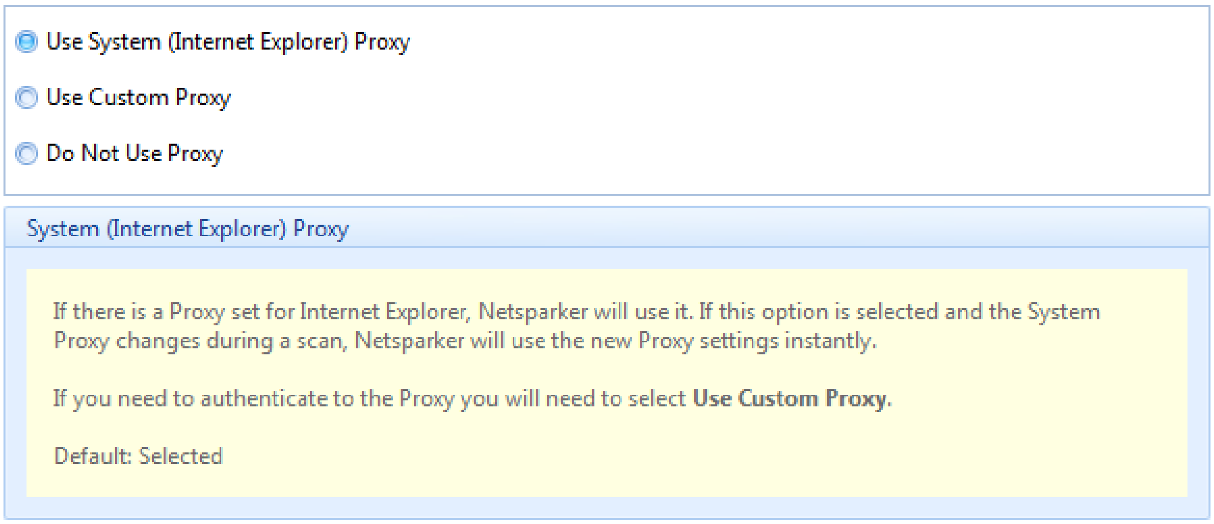
The **User Agent** settings determine the User-Agent header that will be appended to HTTP requests. This setting may either be selected from a pre-defined list of standard user agent strings, or manually entered to achieve a specific custom result.

The **Request Timeout** setting determines the time period (in milliseconds) that Netsparker will wait, after issuing each HTTP request, before treating a lack of response as a timeout.

The settings in the **HTTP Request Settings** section determine the principle characteristics of the HTTP requests issued by Netsparker during crawling and attacking.

Proxy

Proxy settings determine how Netsparker interacts with an upstream proxy. This should not be confused with Netsparker's internal proxy (see crawling settings), which is used to capture the URLs associated with manual page crawling from a browser.


[✓ Use System (Internet Explorer) Proxy]
Use Custom Proxy
Do Not Use Proxy
System (Internet Explorer) Proxy

If there is a Proxy set for Internet Explorer, Netsparker will use it. If this option is selected and the System Proxy changes during a scan, Netsparker will use the new Proxy settings instantly.

If you need to authenticate to the Proxy you will need to select **Use Custom Proxy**.

Default: Selected

Select **Use System Proxy** to configure Netsparker to route its outbound HTTP requests via the local system proxy, as configured within Internet Explorer's settings.

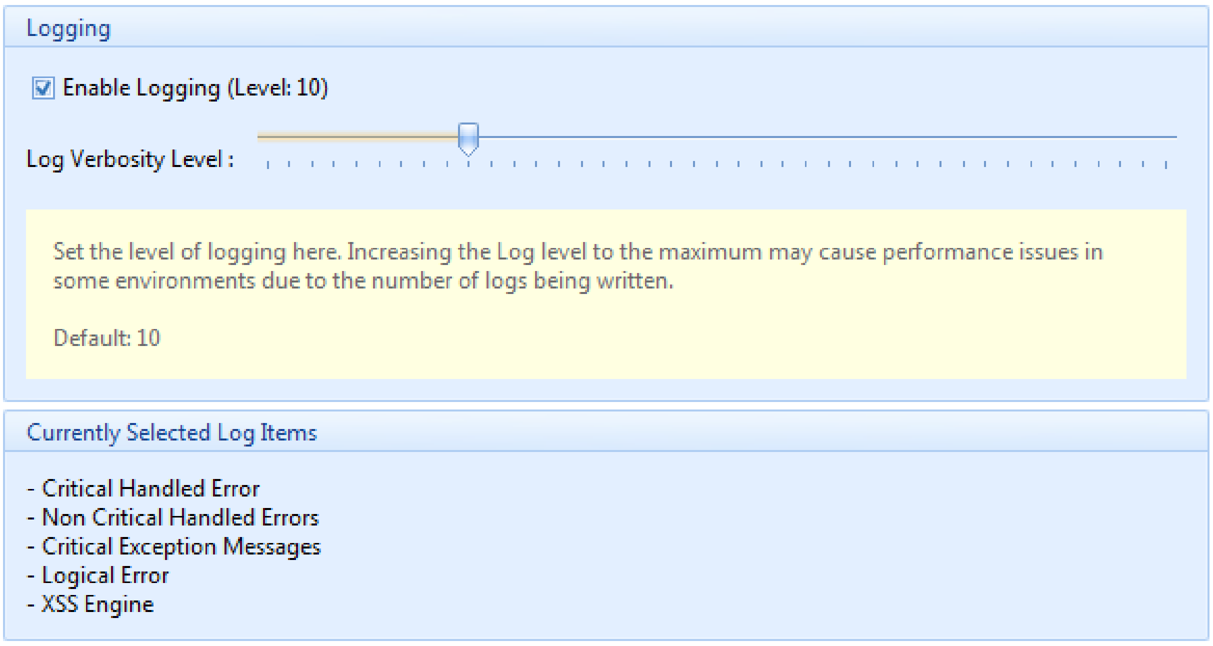
Select **Use Custom Proxy** to specify a custom proxy to be used by Netsparker. If you select this option, you will also be presented with the opportunity to enter authentication credentials for access to the specified proxy.

Select **Do Not Use Proxy** to configure Netsparker to connect directly to the internet, bypassing all local proxy servers.

General Settings

Logging

Logging settings determine whether and in what level of detail Netsparker logs its scanning activity.


Logging

☑ Enable Logging (Level: 10)

Log Verbosity Level :

Set the level of logging here. Increasing the Log level to the maximum may cause performance issues in some environments due to the number of logs being written.

Default: 10

Currently Selected Log Items

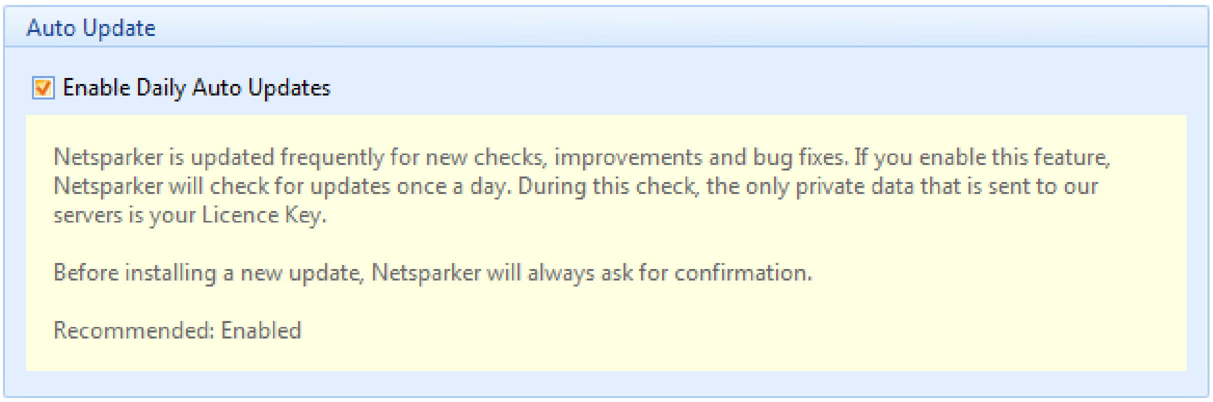
- Critical Handled Error
- Non Critical Handled Errors
- Critical Exception Messages
- Logical Error
- XSS Engine

The **Enable Logging** setting determines whether Netsparker creates log files as it executes scanning.

The **Log Verbosity Level** setting determines the individual line item categories that qualify for inclusion in Netsparker's log files. By moving the configuration slider to the left or right, it is possible to decrease or increase the log detail level. For any given detail level selection, the list of qualifying log item categories will be displayed in the **Currently Selected Log Items** panel.

Auto Update

Auto Update settings influence how Netsparker checks for software updates.


Auto Update

☑ Enable Daily Auto Updates

Netsparker is updated frequently for new checks, improvements and bug fixes. If you enable this feature, Netsparker will check for updates once a day. During this check, the only private data that is sent to our servers is your Licence Key.

Before installing a new update, Netsparker will always ask for confirmation.

Recommended: Enabled

When checked, the **Enable Daily Auto Updates** setting causes Netsparker to connect once daily to its update server to check for the availability of an updated software version.

Support

Getting Support

Netsparker has an easy-to-use and intuitive interface which makes it easy to configure and run a new scan without a guide.

However, we recognize that from time to time, you'll need assistance. By keeping an eye on the Netsparker blog - http://www.netsparker.com/blog/ , you'll have access to tips & tricks and the latest news about Netsparker.

Get in touch with us

We are fanatical about support. You can contact us by email or telephone for any other issue and we will get back to you as soon as possible.

Email:
support@netsparker.com

Phone:
+44 845 686 3001 (weekdays between 09:00 and 17:30 GMT)

Glossary

B

Black-box Scanning

Scanning the security of a system without having the necessary privileges on the target system, in the same way as an attacker would for a web application, black-box scanning means having no source-code, administrative account or access to system hardware etc.

F

False-Positive

False-Positive is when the software reports a vulnerability but, in reality, there is no vulnerability in the scanned web application. Before Netsparker, False-Positives were considered as necessary side effect as there was no software that could eliminate false-positives. Netsparker is the first tool that doesn’t report false-positives.

False-Negative

When a tool misses a vulnerability that actually exists in the application, it’s called as false-negative.

W

Web Application Security Scanner

This is an automated tool which is designed to find vulnerabilities in web applications.