Cybersecurity Tools

م.د.ابراهيم محمد الحليمه

2025

What is vulnerability scanning?

Vulnerability scanning is commonly considered to be the most efficient way to check your site against a huge list of known vulnerabilities - and identify potential weaknesses in the security of your applications. Vulnerability scanning can be used as part of a standalone assessment, or as part of a continuous overall security monitoring strategy.

What is a web vulnerability scanner?

Vulnerability scanners are automated tools that scan web applications to look for security vulnerabilities. They test web applications for common security problems such as cross-site scripting (XSS), SQL injection, and cross-site request forgery (CSRF).

More capable scanners may be able to delve further into an application by utilizing more advanced techniques. Pioneering application system testing techniques mean that Burp Scanner, the engine powering Burp Suite application security testing products, can find vulnerabilities many other scanners would miss, including asynchronous SQL injection and blind SSRF for instance.

How does a web vulnerability scanner work?
Web vulnerability scanners work by automating several processes. These include application spidering and crawling, discovery of default and common content, and probing for common vulnerabilities.

There are two primary approaches to vulnerability scanning - passive, and active. A passive scan performs non-intrusive checks, simply looking at items to determine if they are vulnerable. You can visualize this method by imagining encountering a door, but not touching it to see if it's open or locked. If the door is closed, that marks the end of that branch of your investigation.

An active scan on the other hand, is a simulated attack on your site in order to access vulnerabilities as they would appear to an outsider. If you visualize this as a door, the fact that it may be closed would not present a dead-end. Instead, your investigation would push you to test the door, perhaps pick the lock, or even force entry.

Some scan types also involve authentication, whereby the scanner uses access permissions to establish if there are further open or closed "doors" within the application. Some scanners are able to acquire these access permissions themselves, and some will need them provided prior to testing.

The scanner will then produce a report of varying detail, depending on the type of scan performed. This report usually includes the specific request and response that the application used to diagnose each reported vulnerability, enabling a knowledgeable user to manually investigate and confirm the bug's existence.

> These software programs scan web applications to identify security vulnerabilities including cross-site scripting, SQL injection, and path traversal. Examples of tools include Burp Suite, Nikto, Paros Proxy, and SQLMap.

What are the common vulnerabilities detected by automated scanning?

Several categories of common vulnerabilities can be detected by scanners with a degree of reliability. Some scanners can detect a wider range of vulnerabilities, for example if their logic is more frequently updated. Regular updates can play a big part in maintaining your security posture - once a vulnerability becomes public, it's also public to hackers. This is something to consider when selecting your vulnerability scanning tool.

Vulnerabilities reliably detected by run-of-the-mill scanners include, but are not necessarily limited to:

Reflected cross-site scripting (XSS)

Automated scanners typically send test strings containing HTML markup and search the responses for these strings, enabling them to detect basic XSS flaws.

Straightforward directory listings

This type of vulnerability can be identified by requesting the directory path, and looking for a response containing text that looks like a directory listing.

Directory traversal

Some path traversal vulnerabilities can be detected by submitting a traversal sequence targeting a known file, and searching the response for the appearance of this file.

Some command injection vulnerabilities

These types of vulnerability can often be detected by injecting a command that causes a time delay, or echoes a specific string into the application's response.

SQL injection

This allows an attacker to interfere with queries that an app makes to its database. This can sometimes be detected using basic payloads designed to cause recognizable error messages.

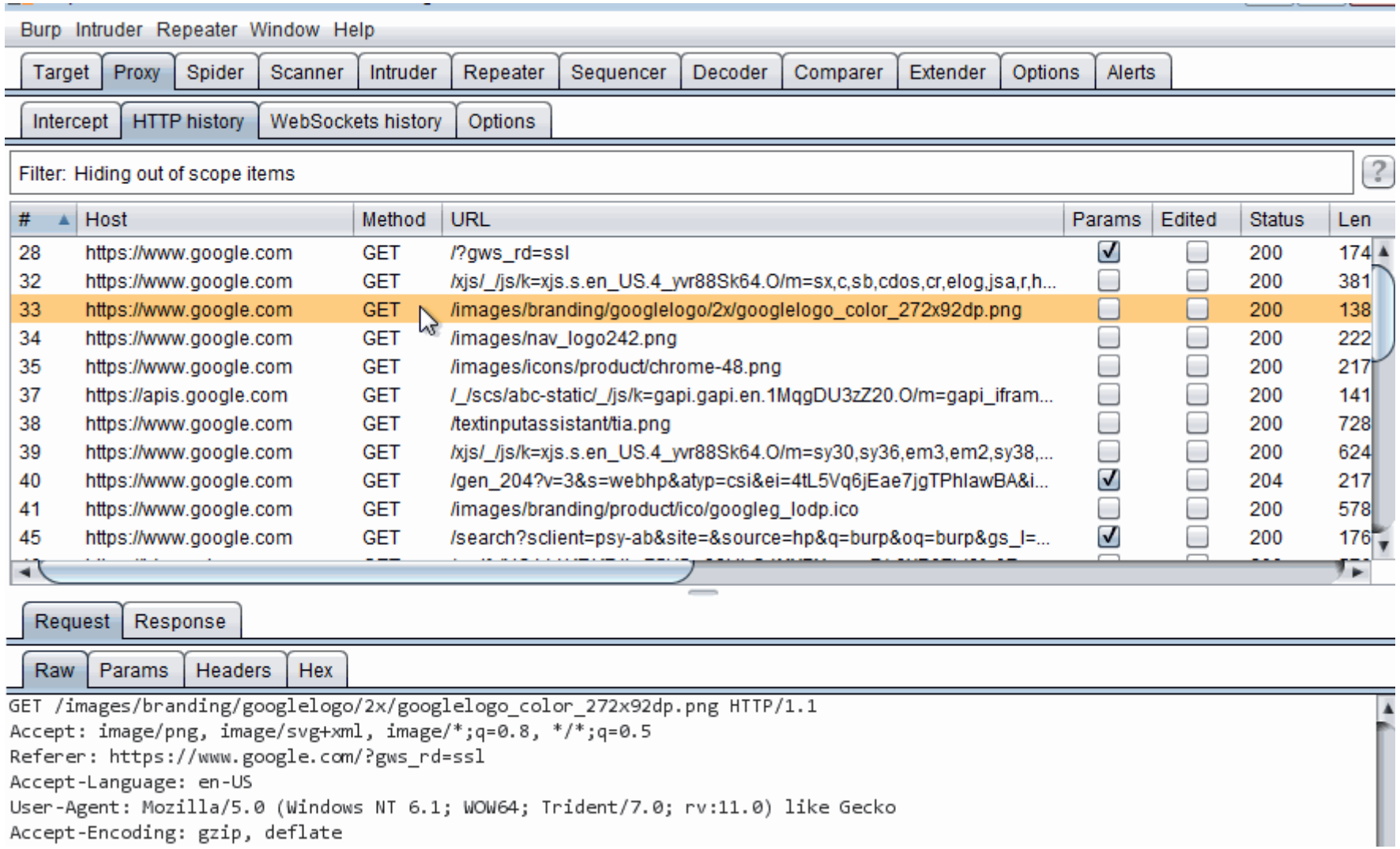
Open redirection

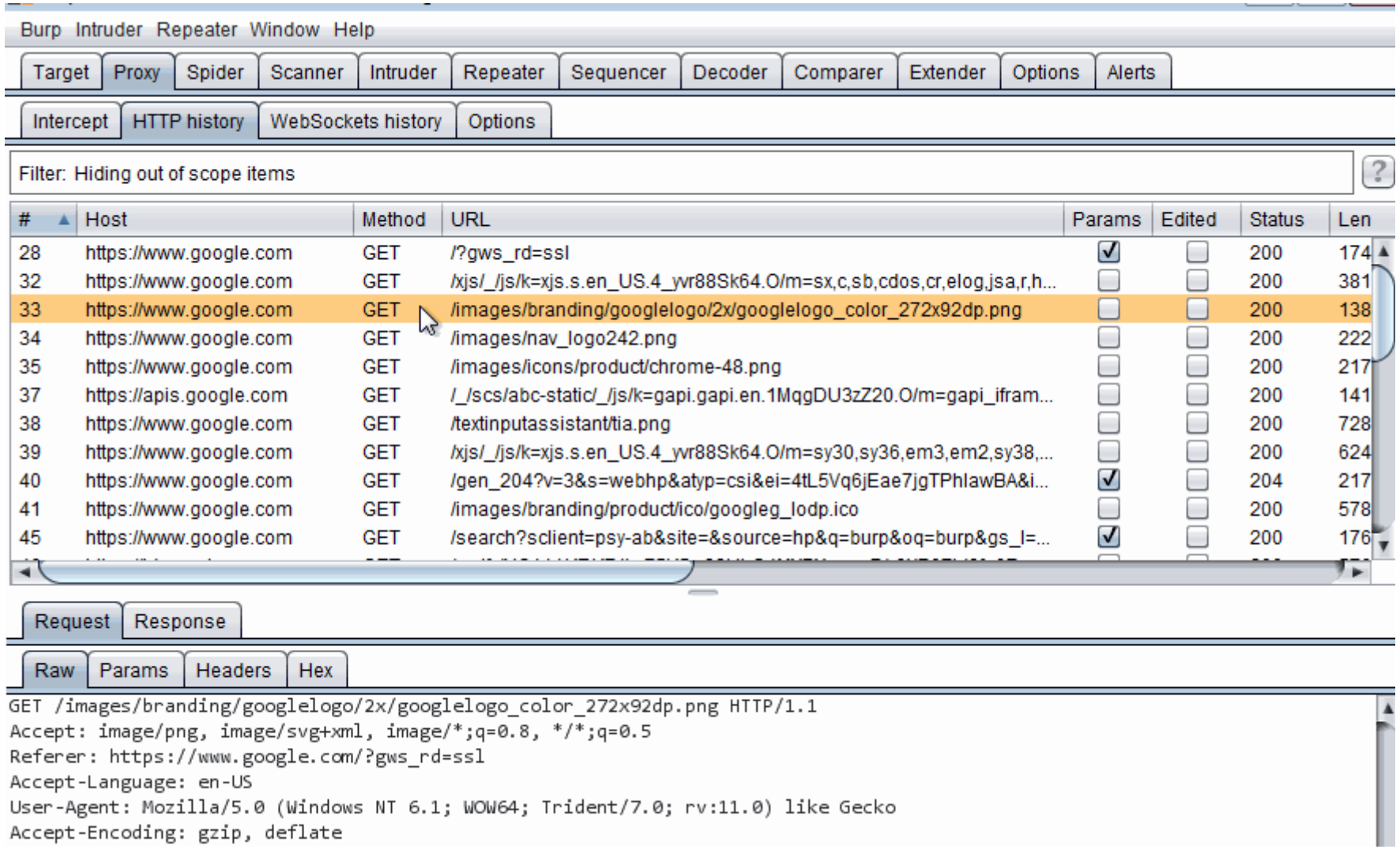
A scanner tests for these vulnerabilities by submitting payloads, designed to test whether a parameter can cause redirection to an arbitrary external domain.

What is the best vulnerability scanner?

There are no true benchmarks for evaluating a vulnerability scanner, as each one will usually have its own strengths and weaknesses depending on your use case. Bear in mind that even if a vendor presents benchmarking criteria for their scanner, this data has the potential to lean heavily in their favor. Whatever your use case, it's important to select the type of scanner that comes packaged the way you need it - so you can hit the ground running.

PortSwigger's application security testing products both use the same underlying web vulnerability scanner - Burp Scanner. Whether you want software designed for an individual tester looking to improve workflows, or enterprises wanting to scale and automate, there's a Burp Suite for everyone.


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Intercept HTTP history WebSockets history Options
Filter: Hiding out of scope items
Host Method URL Params Edited Status Len
28 https://www.google.com GET /?gws_rd=ssl 200 174
32 https://www.google.com GET /xjs/_/js/k=xjs.s.en_US.4_yvr88Sk64.O/m=sx,c,sb,cdos,cr,elog,jsa,r,h... 200 381
33 https://www.google.com GET /images/branding/googlelogo/2x/googlelogo_color_272x92dp.png 200 138
34 https://www.google.com GET /images/nav_logo242.png 200 222
35 https://www.google.com GET /images/icons/product/chrome-48.png 200 217
37 https://apis.google.com GET /_/scs/abc-static/_/js/k=gapi.gapi.en.1MqgDU3zZ20.O/m=gapi_ifram... 200 141
38 https://www.google.com GET /textinputassistant/tia.png 200 728
39 https://www.google.com GET /xjs/_/js/k=xjs.s.en_US.4_yvr88Sk64.O/m=sy30,sy36,em3,em2,sy38,... 200 624
40 https://www.google.com GET /gen_204?v=3&s=webhp&atyp=csi&ei=4tL5Vq6jEae7jgTPhlawBA&i... 204 217
41 https://www.google.com GET /images/branding/product/ico/googleg_lodp.ico 200 578
45 https://www.google.com GET /search?sclient=psy-ab&site=&source=hp&q=burp&oq=burp&gs_l=... 200 176
Request Response
Raw Params Headers Hex
GET /images/branding/googlelogo/2x/googlelogo_color_272x92dp.png HTTP/1.1
Accept: image/png, image/svg+xml, image/*;q=0.8, */*;q=0.5
Referer: https://www.google.com/?gws_rd=ssl
Accept-Language: en-US
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Options Alerts
Intercept HTTP history WebSockets history Options
Filter: Hiding out of scope items
Host Method URL Params Edited Status Len
28 https://www.google.com GET /?gws_rd=ssl 200 174
32 https://www.google.com GET /xjs/_/js/k=xjs.s.en_US.4_yvr88Sk64.O/m=sx,c,sb,cdos,cr,elog,jsa,r,h... 200 381
33 https://www.google.com GET /images/branding/googlelogo/2x/googlelogo_color_272x92dp.png 200 138
34 https://www.google.com GET /images/nav_logo242.png 200 222
35 https://www.google.com GET /images/icons/product/chrome-48.png 200 217
37 https://apis.google.com GET /_/scs/abc-static/_/js/k=gapi.gapi.en.1MqgDU3zZ20.O/m=gapi_ifram... 200 141
38 https://www.google.com GET /textinputassistant/tia.png 200 728
39 https://www.google.com GET /xjs/_/js/k=xjs.s.en_US.4_yvr88Sk64.O/m=sy30,sy36,em3,em2,sy38,... 200 624
40 https://www.google.com GET /gen_204?v=3&s=webhp&atyp=csi&ei=4tL5Vq6jEae7jgTPhlawBA&i... 204 217
41 https://www.google.com GET /images/branding/product/ico/googleg_lodp.ico 200 578
45 https://www.google.com GET /search?sclient=psy-ab&site=&source=hp&q=burp&oq=burp&gs_l=... 200 176
Request Response
Raw Params Headers Hex
GET /images/branding/googlelogo/2x/googlelogo_color_272x92dp.png HTTP/1.1
Accept: image/png, image/svg+xml, image/*;q=0.8, */*;q=0.5
Referer: https://www.google.com/?gws_rd=ssl
Accept-Language: en-US
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate