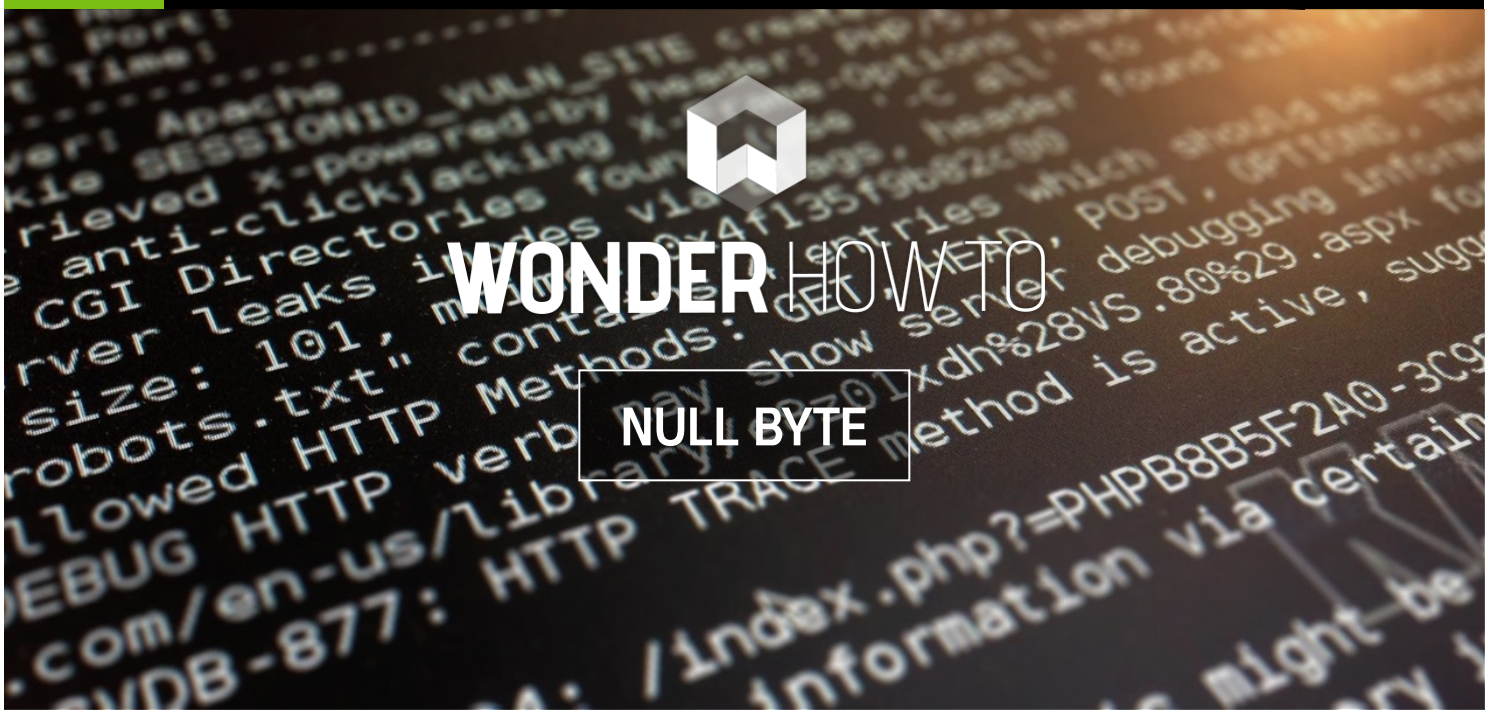


HOW TO

Scan for Vulnerabilities on Any Website Using Nikto

BY KODY 03/14/2019 2:30 AM POPULAR RECON

Before attacking any website, a hacker or penetration tester will first compile a list of target surfaces. After they've used some good recon and found the right places to point their scope at, they'll use a web server scanning tool such as Nikto for hunting down vulnerabilities that could be potential attack vectors.

Nikto is a simple, open-source web server scanner that examines a website and reports back vulnerabilities that it found which could be used to exploit or hack the site. Also, it's one of the most widely used website vulnerabilities tools in the industry, and in many circles, considered the industry standard.

Although this tool is extremely effective, it's *not* stealthy at all. Any site with an intrusion-detection system or other security measures in place will detect that it's being scanned. Initially designed for security testing, stealth was never a concern.

- **Don't Miss:** Use Metasploit's WMAP to Scan Web Apps for Vulnerabilities

The Right Way to Use Nikto

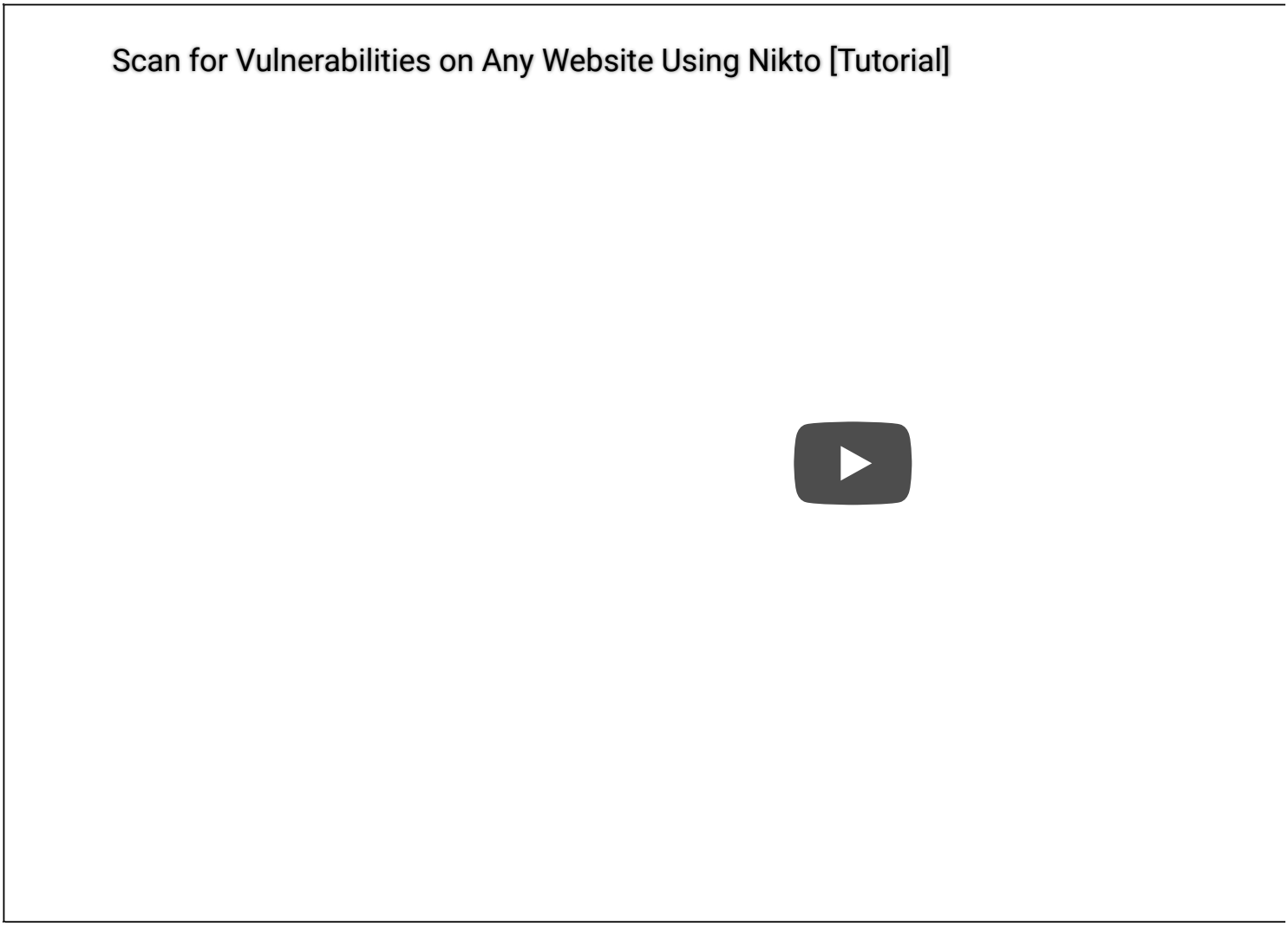
If you just run Nikto by itself on a targeted website, you may not know what to do with the information from the scan. Nikto is actually more like a laser pointer to call in a much larger strike, and you'll see how that plays out in a little bit.

First, let's talk about the target surface. This is pretty much anywhere a hacker will attempt to attack and could include things such as network-exposed printers and a web server. When we get to using Nikto later, we'll need to provide it with one of three different types of information: an IP address for a local service, a web domain to attack, or an SSL/HTTPS website.

Before diving right into a scan with Nikto, it's better to do some additional reconnaissance using an open-source intelligence tool such as Maltego. Tools like this can help build a profile and a more focused list of available targets that should be concentrated on. Once that's done, Nikto can be used to hone in on potential vulnerabilities for targets on the list.

- **Don't Miss:** How to Use Maltego to Fingerprint an Entire Network

If lucky, a vulnerability with a weaponized exploit will be found, meaning there's a tool out there already to take advantage of the weakness. With the appropriate tool, which will automatically exploit the vulnerability, a hacker can gain access to the target to perform any number of behind-the-scenes attacks, like adding code to perform a malicious activity.



Step 1

Install Nikto

If you're running Kali Linux, Nikto comes preinstalled, so you don't have to download or install anything. It'll be located in the "Vulnerability Analysis" category. If you don't have it for some reason, you can get Nikto from its GitHub or just use the **apt install** command.

```
apt install nikto
```

If you're doing this on a Mac, you can use Homebrew to install Nikto.

```
brew install nikto
```

Step 2

Get to Know Nikto

Before you dive into scanning web servers with Nikto, lets you use the **-Help** option to see everything that can be done inside Nikto.

```
nikto -Help
```

```
Options:
       -ask+               Whether to ask about submitting updates
                               yes   Ask about each (default)
                               no    Don't ask, don't send
                               auto  Don't ask, just send
       -Cgidirs+           Scan these CGI dirs: "none", "all", or values like "/cgi/ /cgi-a
       -config+            Use this config file
       -Display+           Turn on/off display outputs:
                               1     Show redirects
                               2     Show cookies received
                               3     Show all 200/OK responses
                               4     Show URLs which require authentication
                               D     Debug output
                               E     Display all HTTP errors
                               P     Print progress to STDOUT
                               S     Scrub output of IPs and hostnames
                               V     Verbose output
       -dbcheck           Check database and other key files for syntax errors
       -evasion+          Encoding technique:
                               1     Random URI encoding (non-UTF8)
                               2     Directory self-reference (/./)
                               3     Premature URL ending
                               4     Prepend long random string
                               5     Fake parameter
                               6     TAB as request spacer
                               7     Change the case of the URL
                               8     Use Windows directory separator (\)
                               A     Use a carriage return (0x0d) as a request spacer
                               B     Use binary value 0x0b as a request spacer
        -Format+           Save file (-o) format:
                               csv   Comma-separated-value
                               htm   HTML Format
                               nbe   Nessus NBE format
                               sql   Generic SQL (see docs for schema)
                               txt   Plain text
                               xml   XML Format
                               (if not specified the format will be taken from the file ext
       -Help              Extended help information
       -host+             Target host
       -404code           Ignore these HTTP codes as negative responses (always). Format is
       -404string         Ignore this string in response body content as negative response
       -id+               Host authentication to use, format is id:pass or id:pass:realm
       -key+              Client certificate key file
       -list-plugins      List all available plugins, perform no testing
       -maxtime+          Maximum testing time per host (e.g., 1h, 60m, 3600s)
       -mutate+           Guess additional file names:
                               1     Test all files with all root directories
```

```
                           2     Guess for password file names
                           3     Enumerate user names via Apache (/~user type requests)
                           4     Enumerate user names via cgiwrap (/cgi-bin/cgiwrap/~us
                           5     Attempt to brute force sub-domain names, assume that t
                           6     Attempt to guess directory names from the supplied dic
       -mutate-options     Provide information for mutates
       -nointeractive      Disables interactive features
       -nolookup           Disables DNS lookups
       -nossl              Disables the use of SSL
       -no404              Disables nikto attempting to guess a 404 page
       -Option             Over-ride an option in nikto.conf, can be issued multiple times
       -output+            Write output to this file ('.' for auto-name)
       -Pause+             Pause between tests (seconds, integer or float)
       -Plugins+           List of plugins to run (default: ALL)
       -port+              Port to use (default 80)
       -RSAcert+           Client certificate file
       -root+              Prepend root value to all requests, format is /directory
       -Save               Save positive responses to this directory ('.' for auto-name)
       -ssl                Force ssl mode on port
       -Tuning+            Scan tuning:
                           1     Interesting File / Seen in logs
                           2     Misconfiguration / Default File
                           3     Information Disclosure
                           4     Injection (XSS/Script/HTML)
                           5     Remote File Retrieval - Inside Web Root
                           6     Denial of Service
                           7     Remote File Retrieval - Server Wide
                           8     Command Execution / Remote Shell
                           9     SQL Injection
                           0     File Upload
                           a     Authentication Bypass
                           b     Software Identification
                           c     Remote Source Inclusion
                           d     WebService
                           e     Administrative Console
                           x     Reverse Tuning Options (i.e., include all except speci
       -timeout+           Timeout for requests (default 10 seconds)
       -Userdbs            Load only user databases, not the standard databases
                           all   Disable standard dbs and load only user dbs
                           tests Disable only db_tests and load udb_tests
       -useragent          Over-rides the default useragent
       -until              Run until the specified time or duration
       -update             Update databases and plugins from CIRT.net
       -useproxy           Use the proxy defined in nikto.conf, or argument http://server:po
       -Version            Print plugin and database versions
       -vhost+             Virtual host (for Host header)
                + requires a value
```

Step 3

Use the Basic Syntax

As you can see from the previous step, Nikto has many options, but for our purposes, we'll stick to the basic syntax as follows. We'll substitute the <IP or hostname> with the actual IP address or hostname sans angle brackets.

```
nikto -h <IP or hostname>
```

However, Nikto is capable of doing a scan that can go after SSL and port 443, the port that HTTPS websites use (HTTP uses port 80 by default). So we're not just limited to scanning old sites, we can do vulnerability assessments on sites that use SSL, which is pretty much a requirement these days to be indexed in search results.

If we know it's an SSL site that we're targeting, we can specify it in Nikto to save some time on the scan by adding **-ssl** to the end of the command.

```
nikto -h <IP or hostname> -ssl
```

Step 4

Scan an SSL-Enabled Website

For example, let's start with scanning pbs.org to see some of the types of information that a Nikto scan will show. After it connects to port 443, we see that there's some useful information about the cipher and a list of other details like that the server is Nginx, but there's not a whole lot of interesting data here for us.

```
nikto -h pbs.org -ssl
```

```
- Nikto v2.1.6
---------------------------------------------------------------------------
- STATUS: Starting up!
+ Target IP:                  54.225.198.196
+ Target Hostname:            pbs.org
+ Traget Port:                443
---------------------------------------------------------------------------
+ SSl Info:                   Subject:        /CN=www.pbs.org
                                      Altnames:       account.pbs.org, admin.pgs.org, dip
                                      jaws..pbs.org, kids.pbs.org, koth-qa.svp.pbs.org, l
                              urs.pbs.org, video.pbs.org, weta-qa.svp.pbs.org, whut-qa.sv
                    Ciphers:  ECDHE-RSA-AES128-GCM-SHA256
                    Issuer:           /C-US/0=Let's Encrypt/CN=Let's Encrypt Authority X3
+ Start Time:                2018-12-05 23:34:06 (GMT-8)
---------------------------------------------------------------------------
+ Server: nginx
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to pro
+ Uncommon header 'x-pbs-fwsrvname' found, with contents: fwcacheproxy1
+ The site uses SSL and the Strict-Transport-Security HTTP header is not defined.
```

```
+ The X-Content-Type-Options header is not set. This could allow the user agent to render t
+ Root page / redirects to: https://www.pbs.org/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ RC-1918 IP address found in the 'x-pbs-appsvrip' header: The IP is "10.137.181.52".
+ Uncommon header 'x-cache-fs-status' found, with contents: EXPIRED
+ Uncommon header 'x-pbs-appsvrname' found, with contents: fwcacheproxy1
+ Uncommon header 'x-pbs-appsvrip' found, with contents: 10.137.181.52
+ Server leaks inodes via ETags, header found with file /pbs.org.zip, fields: 0x5b96537e 0x
+ 7446 requests: 0 error(s) and 10 item(s) reported on remote host
+ End Time:                           2018-12-06 00:30:29 (GMT-8) (3383 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Step 5

Scan an IP Address

Now that we performed a quick scan of a website, let's try using Nikto on a local network to find embedded servers such as a login page for a router or an HTTP service on another machine that's just a server with no website. To get started, let's find our IP address using ifconfig.

```
ifconfig
```

```
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST>  mtu 1500
        inet 192.168.0.48  netmask 0xffffff00  broadcast 192.168.0.255
        inet6 XXXX::XXX:XXXX:XXXX:XXXX%en0  prefixlen 64  secured scopeid 0x8
        ether XX:XX:XX:XX:XX:XX  txqueuelen 1000  (Ethernet)
        inet6 XXXX::XXX:XXXX:XXXX:XXXX%en0  prefixlen 64 autoconf secured
        inet6 XXXX::XXX:XXXX:XXXX:XXXX%en0  prefixlen 64 autoconf temporary
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active

en2: flags=8863<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST>  mtu 1500
        options=60<TS04,TS06>
        ether XX:XX:XX:XX:XX:XX
        media: autoselect <full-duplex>
        status: inactive
```

The IP address we want is the "inet" one. Then we can run **ipcalc** on it to get our network range. If you don't have **ipcalc**, you can install it with **apt install ipcalc**, then try again. The range will be next to "Network," in my case, 192.168.0.0/24.

```
ipcalc 192.168.0.48
```

```
Address:   192.168.0.48         11000000.10101000.00000000. 00110000
Netmask:   255.255.255.0 = 24   11111111.11111111.11111111. 00000000
Wildcard:  0.0.0.255            00000000.00000000.00000000. 11111111
=>
Network:   192.168.0.0/24       11000000.10101000.00000000. 00000000
HostMin:   192.168.0.1          11000000.10101000.00000000. 00000001
HostMax:   192.168.0.254        11000000.10101000.00000000. 11111110
Broadcast: 192.168.0.255        11000000.10101000.00000000. 11111111
Hosts/Net: 254                   Class C, Private Internet
```

Now, we're going to want to run Nmap to find services running in the network range. Let's scan port 80 with our range and tack on **-oG** (grepable output) to extract only the hosts that are up and running, i.e., the ones responding indicating that port 80 is open. Then we'll save everything to a file, which I'm naming **nullbyte.txt**, but could be named anything.

```
nmap -p 80 192.168.0.0/24 -oG nullbyte.txt
```

```
Starting Nmap 7.60 ( https://nmap.org ) at 2018-12-06 00:43 PST
Nmap scan report for 192.168.0.1
Host is up (0.021s latency).

PORT   STATE SERVICE
80/tcp open  http

Nmap scan report for 192.168.0.2
Host is up (0.088s latency).

PORT   STATE SERVICE
80/tcp open  http

Nmap scan report for 192.168.0.4
Host is up (0.032s latency).

PORT   STATE SERVICE
80/tcp open  http

Nmap scan report for 192.168.0.5
Host is up (0.020s latency).

PORT   STATE SERVICE
80/tcp open  http

Nmap scan report for 192.168.0.11
Host is up (0.068s latency).

PORT   STATE  SERVICE
80/tcp closed http

Nmap scan report for 192.168.0.24
Host is up (0.023s latency).
```

```
PORT   STATE  SERVICE
80/tcp closed http

Nmap scan report for 192.168.0.31
Host is up (0.059s latency).

PORT   STATE  SERVICE
80/tcp closed http

Nmap scan report for 192.168.0.48
Host is up (0.030s latency).

PORT   STATE  SERVICE
80/tcp closed http

Nmap scan report for 192.168.0.60
Host is up (0.092s latency).

PORT   STATE  SERVICE
80/tcp closed http

Nmap done: 256 IP addresses (9 hosts up) scanned in 8.92 seconds
```

There's a nice little trick that can send all the up hosts directly to Nikto for scanning. We use **cat** to read the output stored in our **nullbyte.txt** document (or whatever you named it). Then, there's **awk**, a Linux tool that will help search for the following pattern, where **Up** means the host is up and **print $2** means to print out the second word in that line for each, i.e., just the IP address. Then, we send that data to a new file called **targetIP.txt** (or whatever you'd like to name it).

```
cat nullbyte.txt | awk '/Up$/{print $2}' | cat >> targetIP.txt
```

We can now view the contents of our new file with **cat** to see all the IP addresses that have port 80 open.

```
cat targetIP.txt
```

```
192.168.0.1
192.168.0.2
192.168.0.4
192.168.0.5
192.168.0.11
192.168.0.24
192.168.0.31
192.168.0.48
192.168.0.60
```

This is perfect for Nikto because it can easily interpret files like this. So we can send this output over to Nikto with the following command.

```
nikto -h targetIP.txt
```

The results will look similar to the ones we got when performing the SSL scan.

Step 6

Scan an HTTP Website

We've scanned a secure website and an IP address on a local network, and now it's time to go after an unsecured web domain using port 80. For this example, I'm using "afl.com.au," which was not using SSL at the time I performed this scan.

```
nikto -h www.afl.com.au
```

```
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          159.180.84.10
+ Target Hostname:    www.afl.com.au
+ Target Port:        80
+ Start Time:         2018-12-05 21:48:32 (GMT-8)
---------------------------------------------------------------------------
+ Server: instart/nginx
+ Retried via header: 1.1 varnish (Varnish/6.1), 1.1 e9ba0a9a729ff2960a04323bf1833df8.cloud
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to pro
+ Uncommon header 'x-cache' found, with contents: Miss from cloudfront
+ Uncommon header 'x-instart-cache-id' found, with contents: 17:12768802731504004780::1544
+ Uncommon header 'v-cache-hit' found, with contents: Hit
+ Uncommon header 'x-amz-cf-id' found, with contents: Dr-r6OwO5kk9ABt4ejzpc7R7AIF6SuH6kfJH
+ Uncommon header 'x-instart-request-id' found, with contents: 12814413144077601501:BEQ01-
+ Uncommon header 'x-oneagent-js-injection' found, with contents: true
+ Uncommon header 'grace' found, with contents: cache
+ The X-Content-Type-Options header is not set. This could allow the user agent to render
+ Uncommon header 'x-ruxit-js-agent' found, with contents: true
+ Cookie dtCookie created without the httponly flag
+ Server banner has changed from 'instart/nginx' to 'nginx' which may suggest a WAF, load
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Entry '/sites/' in robots.txt returned a non-forbidden or redirect HTTP code (200)
+ Entry '/search/' in robots.txt returned a non-forbidden or redirect HTTP code (200)
+ Entry '*.mobileapp' in robots.txt returned a non-forbidden or redirect HTTP code (400)
+ Entry '*.liveradio' in robots.txt returned a non-forbidden or redirect HTTP code (400)
+ Entry '*.smartmobile' in robots.txt returned a non-forbidden or redirect HTTP code (400)
+ Entry '*.responsive' in robots.txt returned a non-forbidden or redirect HTTP code (400)
+ Entry '/stats?*/' in robots.txt returned a non-forbidden or redirect HTTP code (200)
+ "robots.txt" contains 8 entries which should be manually viewed.
```

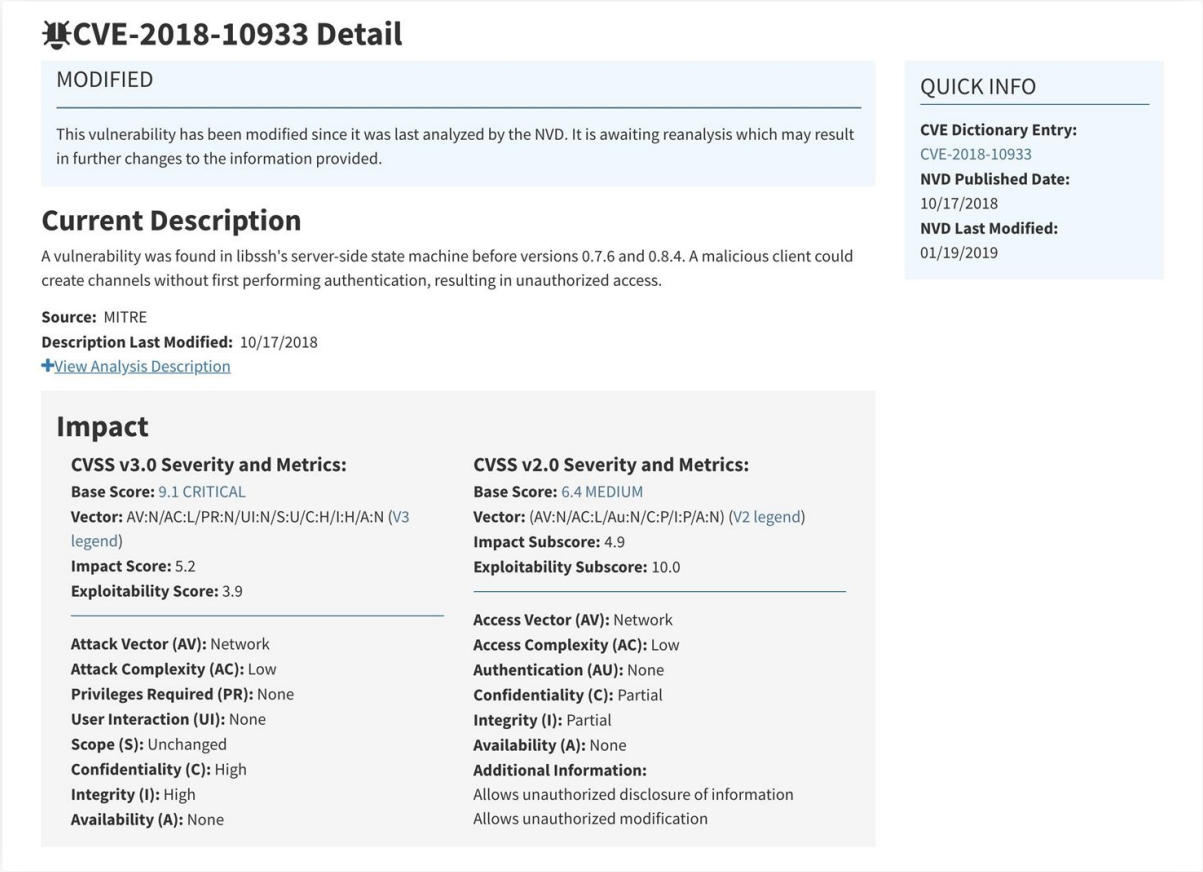
```
+ OSVDB-3092: /sitemap.xml: This gives a nice listing of the site content.
+ OSVDB-3092: /psql_history: This might be interesting...
+ OSVDB-3092: /global/: This might be interesting...
+ OSVDB-3092: /home/: This might be interesting...
+ OSVDB-3092: /news: This might be interesting...
+ OSVDB-3092: /search.vts: This might be interesting...
+ OSVDB-3092: /stats.htm: This might be interesting...
+ OSVDB-3092: /stats.txt: This might be interesting...
+ OSVDB-3092: /stats/: This might be interesting...
+ OSVDB-3092: /Stats/: This might be interesting...
+ OSVDB-3093: /.wwwacl: Contains authorization information
+ OSVDB-3093: /.www_acl: Contains authorization information
+ OSVDB-3093: /.htpasswd: Contains authorization information
+ OSVDB-3093: /.access: Contains authorization information
+ OSVDB-3093: /.addressbook: PINE addressbook, may store sensitive e-mail address contact i
+ OSVDB-3093: /.bashrc: User home dir was found with a shell rc file. This may reveal file
+ OSVDB-3093: /.bash_history: A user's home directory may be set to the web root, the shell
+ OSVDB-3093: /.forward: User home dir was found with a mail forward file. May reveal where
+ OSVDB-3093: /.history: A user's home directory may be set to the web root, the shell hist
+ OSVDB-3093: /.htaccess: Contains configuration and/or authorization information
+ OSVDB-3093: /.lynx_cookies: User home dir found with LYNX cookie file. May reveal cookies
+ OSVDB-3093: /.mysql_history: Database SQL?
+ OSVDB-3093: /.passwd: Contains authorization information
+ OSVDB-3093: /.pinerc: User home dir found with a PINE rc file. May reveal system informat
+ OSVDB-3093: /.plan: User home dir with a .plan, a now mostly outdated file for delivering
+ OSVDB-3093: /.proclog: User home dir with a Procmail rc file. May reveal mail traffic, di
+ OSVDB-3093: /.procmailrc: User home dir with a Procmail rc file. May reveal subdirectorie
+ OSVDB-3093: /.profile: User home dir with a shell profile was found. May reveal directory
+ OSVDB-3093: /.rhosts: A user's home directory may be set to the web root, a .rhosts file
+ OSVDB-3093: /.sh_history: A user's home directory may be set to the web root, the shell h
+ OSVDB-3093: /.ssh: A user's home directory may be set to the web root, an ssh file was re
+ OSVDB-5709: /.nsconfig: Contains authorization information
+ /portal/changelog: Vignette richtext HTML editor changelog found.
+ 7587 requests: 4 error(s) and 55 item(s) reported on remote host
+ End Time:           2018-12-05 22:42:41 (GMT-8) (3249 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Above, we can see that there's a Varnish server and some headers that help indicate how the website is configured. However, the juicier stuff is the directories found which can help snag configuration files that may contain credentials or other things that have been misconfigured and left unintentionally accessible.

The items with the OSVDB prefix are vulnerabilities reported in the Open Source Vulnerability Database (a site that shut down in 2016). It's similar to other vulnerability databases such as SecurityFocus, Microsoft's Technet, and Common Vulnerabilities and Exposures. I prefer to check out the National Vulnerability Database.

While there aren't any major things that can be exploited from this scan, if there was, you can use the CVE reference tool to translate the OSVDB identifier to a CVE entry, so you can use one of the other sites above to read more about the vulnerability.

Let's say we found some worth exploring, such as CVE-2018-10933, a Libssh vulnerability we covered in detail previously. The CVE holds information about what can be exploited, what the severity score is (such as critical), and some other information that can help determine an attack vector. If it something worth using, you can search Metasploit, as someone has already likely developed a weaponized module for it to exploit it more easily.



CVE-2018-10933 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

QUICK INFO

CVE Dictionary Entry: CVE-2018-10933
NVD Published Date: 10/17/2018
NVD Last Modified: 01/19/2019

Current Description

A vulnerability was found in libssh's server-side state machine before versions 0.7.6 and 0.8.4. A malicious client could create channels without first performing authentication, resulting in unauthorized access.

Source: MITRE
Description Last Modified: 10/17/2018
+View Analysis Description

Impact

CVSS v3.0 Severity and Metrics:
Base Score: 9.1 CRITICAL
Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N (V3 legend)
Impact Score: 5.2
Exploitability Score: 3.9

Attack Vector (AV): Network
Attack Complexity (AC): Low
Privileges Required (PR): None
User Interaction (UI): None
Scope (S): Unchanged
Confidentiality (C): High
Integrity (I): High
Availability (A): None

CVSS v2.0 Severity and Metrics:
Base Score: 6.4 MEDIUM
Vector: (AV:N/AC:L/Au:N/C:P/I:P/A:N) (V2 legend)
Impact Subscore: 4.9
Exploitability Subscore: 10.0

Access Vector (AV): Network
Access Complexity (AC): Low
Authentication (AU): None
Confidentiality (C): Partial
Integrity (I): Partial
Availability (A): None
Additional Information:
Allows unauthorized disclosure of information
Allows unauthorized modification

Step 7

Pair Scans with Metasploit

One of the best things about Nikto is that you can actually export information into a format that Metasploit can read when you're performing a scan. To do, just use the commands above to perform the scan, but appending **-Format msf+** to the end of it. The format can help us quickly pair data retrieved with a weaponized exploit.

```
nikto -h <IP or hostname> -Format msf+
```

So, in this guide, we went from determining the target's surface area to finding a vulnerability and then pairing it with a weaponized exploit so we don't have to do all of the work. Since Nikto is

not a stealthy tool, it's wise to perform these types of scans from a VPN, through Tor, or another type of service so that your real IP address is not flagged for suspicious behavior.

Don't Miss: How to Scan Websites for Vulnerabilities Using an Android Phone Without Root

---

- Follow Null Byte on Twitter, Flipboard, and YouTube
- Sign up for Null Byte's weekly newsletter
- Follow WonderHowTo on Facebook, Twitter, Pinterest, and Flipboard

Cover photo by Null Byte

Never Miss a Hacking or Security Guide

New Null Byte in your inbox, every week.

Your Email

GET THE NEWSLETTER