

**Scanafi 2.2.2**

March 2020

|  |
| --- |
|  **Scanafi 2.2** |

# Introduction

**What is Scanafi?**

Scanafi is a lightweight utility that enables you to scan hosts on your internal network for SSL/TLS certificates and potential vulnerabilities.

Scanafi performs network discoveries for certificates on port 443 (default) or a set of well-known ports via SSL/TLS and STARTTLS handshakes. It is available as a single executable file for Windows, Linux, and MacOS operating systems.

Scanafi can operate in two simple modes: “provider” or “standalone.”

The provider mode involves the automatic transmission of certificate discovery results to one of the supported Venafi Provider destinations.

There are three supported Provider destinations:

1. Venafi Trust Protection Platform (TPP) via the WebSDK module: This communication is over https and requires that a TPP server hosting the WebSDK module is accessible. Authentication credentials (username, password) for the Trust Protection Platform are required in order to use this Provider destination.
2. Venafi Cloud for DevOps. This communication is over https, authentication credentials (an API token) for Venafi Cloud are also required to use this Provider destination. One can acquire an API token after completing a successful registration to the Venafi Cloud for DevOps servers (<https://ui.venafi.cloud>).
3. Default: When the provider type is Default, Scanafi is executed and it will scan for certificates. The output is shown on the console or as a JSON. Default is used in a scenario where a user wants to run Scanafi and analyze the output before transferring the results to Venafi Cloud or Venafi Trust Protection Platform.

In the standalone operating mode, certificate discovery results are logged to a standard text file, in JSON format. This file can then be collected for later import to a destination (typically the Venafi Trust Protection Platform using the WebSDK API).

For further reference, review the API endpoint ‘POST/certificates/import’ in the WebSDK API documentation found here: <https://docs.venafi.com/Docs/current/TopNav/Content/SDK/WebSDK/API_Reference/r-SDK-POST-Discovery-Import.php>

# Scanafi System Requirements

1. RHEL 5 (Red Hat Enterprise Linux 5), CentOS 7, Ubuntu 18.04 (bionic), 16.04 (xenial)
2. Windows 2012 server or higher
3. Mac OSx 10.14.x (Mojave)

**What’s new for Scanafi 2.2.2**

**OAuth support:** Scanafi now supports Trust Protection Platform’s OAuth authentication.

# \*Important considerations before installing and using Scanafi 2.2.x

Before you install and use Scanafi version 2.2, please read this list carefully so that you’re aware of the changes that have been made. For more details refer to Operation section and Sample config section.

1. Scanafi parameters from Config file: The current method of executing Scanafi parameters through a CLI command will no longer be supported in Scanafi 2.2. The existing CLI options have mostly been migrated to a single configuration (JSON) file that should help users automate Scanafi in their deployment. For more details refer to “New JSON Configuration Parameters” and Examples section.
2. Scanafi 2.2 won't be able to detect the following vulnerability
	1. Sloth Vulnerability
	2. SPDY/NPN server configuration.

# Operation

Scanafi is a command-line based utility. Options that dictate the operating mode, network segments where certificate discovery is performed, and scan parameters are specified in a configuration JSON file when the program is called from the command line. The configuration JSON file parameters are the same regardless of the version (Windows, Linux, MacOS) being run.

## Command Line Parameters

The following table outlines the arguments available for Scanafi:

| Argument | Description | Example |
| --- | --- | --- |
|  --version--verbose | Print version information and then exitPrint verbose output when possible | --version |
| -h --help | Display help information and then exit. | --help |
|  |
| --dry-run | Outputs information to screen only. Results not published to Venafi Cloud. (Optional) | --dry-run |
| --config path--config\_uri URI--access\_token Access Token--tpp-user username--tpp-user password--api\_key API key--save\_config | Path to a scan configuration file. [required]Specify URI from which to read configuration Specify OAuth access token via command line.Specify TPP username via command lineSpecify TPP password via command lineSpecify Venafi Cloud API key via command lineSave configuration to file | --config c:/Scanafi/ConfigFile |

## New JSON Configuration Parameters

| Old Argument | New json config options | DESCRIPTION |
| --- | --- | --- |
| --threads | { "zone": "some-zone", "id": "12345", }, | The number of threads to use for scanning. Default is 100.(Optional) |
| --zone | Readable zone identifier associated with this scan that corresponds to TPP policy name where you want the certificate uploaded.  |
| --id | Identifier used to label Scanafi jobs |
| --tpp-pass--tpp-user--tppurl | "provider": { "type": "tpp", "config": { "url": "https://your TPP URL goes here", "username": "TPP admin name goes here", "password": "TPP password goes here" }, | Specify the URL, username and password required to authenticate with Trust Protection Platform.A Provider type must be one of: 'default' or 'tpp' [Required]Specify the provider type and the apikey as **either** “Condor” or “VenafiCloud” to send results to Venafi Cloud for DevOps. The apikey must be your DevOps key.Specify the provider type and the apikey as “default” to see test results on a console only.  |
| --apikey | "provider": { "type": "VenafiCloud", "config": { "api\_key": "VenafiCloud API key" }  |
| -default | "provider": { "type": "default", "inputs": [ { "type": "CIDR", "subnet": "192.168.0.1/25", "ports": [443] } |
| --range | "inputs": [ { "type": "CIDR", "subnet": "192.168.0.0/25", "ports": [443] } ] | Perform discovery on network range (IP addresses) in CIDR format. Only a single range is supported. If –range and –hosts are omitted, scan the local network as configured on the system.  |
| --hosts | "inputs": [ { "type": "file", "path": ".\\Tests\\TPP\\TestCase10\\host\_file.csv" } ] | Specify a text file containing IP:Port combinations (one per line) to be scanned.  |
| --iface | Not supported |  |
| --multiport |  "inputs": [ { "type": "CIDR", "subnet": "192.168.0.1/25", "ports": [443,7,9,11,13,15,17,25,37,] }], | Specify ports to Scan. Default is just 443.(Optional) |
| --certsonly | "scan\_configuration": { "ciphers\_scan": true, "certificates\_only\_scan": true, "vulnerability\_scan": true, "server\_config\_scan": false }, | The cert only option was migrated to set of options under scan\_configuration option. The options are Boolean type (True or False. Depending on your requirement you can turn them on/off. The default options.  |
|  |  |  |
| --reportjson |  "outputs”: [ { "type": "file", "path":"\\Users\\Alan\\Documents\\result.json",“report\_json”: true }, { "type": "console",“report\_json”: true }] | Output discovery results to JSON file. |

**How to use Scanafi in Windows**

We’ll describe the procedure for Windows below. The procedure is the same for Mac and Linux.

1. Download the appropriate Scanafi OS version that you want to execute.
2. Build your config file using the example outlined in the document.
3. Launch Powershell or cmd prompt.
4. Run the executable.
5. Type “--config” along with the path for the config file. See the screenshot below.



# Examples

1. **Screenshot of new Scanafi command being executed**

./scanafi --dry-run --config samples/sample\_tpp\_config.json

​

#or

./scanafi --config samples/sample\_default\_config.json



1. **Sample Scanafi Config file where output is printed to the command prompt**

Scenario: Run Scanafi to check or validate scan results on the console, without outputting results to .json or Trust Protection Platform.

{

 "zone":"test zone",

 "id":"test default",

 "scan\_configuration":{

 "certificates\_only\_scan":true

 },

 "provider":{

 "type":"default",

 "inputs":[

 {

 "type":"CIDR",

 "subnet":"192.168.0.1/25",

 "ports":[443

]

 }

 ],

 "outputs":[

 {

 "type":"console"

 }

 ]

 }

}

1. **Sample Scanafi Config file that will output results to the TPP folder called “Test\_Scanafi”**

Scenario: Scanafi will scan internal hosts and ports. Results will be uploaded to a TPP folder specified as the zone’s property. Failing to do so will result in an error and results will not be uploaded.

{

 "zone":"test zone",

 "id":"scanafiv2\_default",

 "scan\_configuration":{

 "certificates\_only\_scan":true

 },

 "provider":{

 "type":"default",

 "inputs":[

 {

 "type":"CIDR",

 "subnet":"192.168.0.1/25",

 "ports":["8009","8010-8015"]

 }

 ],

 "outputs":[

 {

 "type":"file",

 "path":"results.json",

 "report\_json":true

 },

 {

 "type":"console"

 }

 ]

 }

}

1. **Sample Scanafi Config file for OAuth using *access\_token***

Scenario: Sample Scanafi Config file that uses Trust Protection Platform OAuth access token.

Notes:

1. When you provide Trust Protection Platform’s username and password, Scanafi will authenticate using OAuth first.
	1. If it fails, it will fall back to Basic Authentication.
	2. If it is successful, then the OAuth token will be revoked after scan results are uploaded to Trust Protection Platform
2. When it encounters an OAuth access\_token, Scanafi assumes you permitted the access\_token to upload results to Trust Protection Platform.
3. If uploading results fails, Scanafi will try five more times to upload results then will stop.

Refer to this Help topic for set up:

* [POST Authorize/OAuth](https://docs.venafi.com/Docs/current/TopNav/Content/SDK/AuthSDK/r-SDKa-POST-AuthorizeOAuth.php)
	+ Note: in the Example > Request section of the topic, use this code instead where the scope is limited to Certificate:discover.

POST https://tpp.venafi.example/vedauth/authorize/oauth

{

 "client\_id":"scanafi",

 "username":"Admin"

 "password":"$WebSDKPassword",

 "scope":"Certificate:discover",

 "state":"r9GN3vFOwjslBPsyiuB6OA=="

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

{

 "zone":"Policy\\ScanafiResults",

 "id":"ScanafiV2",

 "provider":{

 "type":"tpp",

 "config":{

 "url":"http://192.168.0.203",

 "access\_token":"abcdefghijklmnop=="

 },

 "inputs":[

 {

 "type":"CIDR",

 "subnet":"192.168.0.24/32"

 }

 ]

 }

}

1. **Sample Scanafi Config file for scanning multiple ports**

Scenario: Scan multiple ports for SSL/TLS configuration on all hosts. The scan results will be printed to the standard output and to a file.

{

 "zone": "test zone",

 "id": "scanafiv2\_default",

 "scan\_configuration": {

 "certificates\_only\_scan": true,

 },

 "provider": {

 "type": "default",

 "inputs": [

 {

 "type": "CIDR",

 "subnet": "192.168.0.1/25",

 "ports”: [8009, "8010-8015"]

 }

 ],

 "outputs": [

 {

 "type": "file",

 "path": "results.json",

 "report\_json": true

 },

 {

 "type": "console"

 }

 ]

 }

 }

1. **Sample Scanafi config file for vulnerability scans**
	1. Set certificates-only scan to *false*.
	2. You can enable cipher suites for the scan. By default, this is disabled.
	3. You can enable vulnerabilities for the scan. By default, this is disabled.
	4. Enable scan for additional server configurations

{

 "zone":"test zone",

 "id":"test default",

 "scan\_configuration":{

 "ciphers\_scan":true,

 "certificates\_only\_scan":false,

 "vulnerability\_scan":true,

 "server\_config\_scan":true

 },

 "provider":{

 "type":"default",

 "inputs":[

 {

 "type":"CIDR",

 "subnet":"192.168.0.1/25",

 "ports":[443]

 }

 ],

 "outputs":[

 {

 "type":"console"

 }

 ]

 }

}

# Deployment Strategies

Scanafi is deployed as an executable file and requires no installation on the target host. It does not receive commands or perform work from a centralized platform. Scans that are performed are simply passed as command arguments to the executable.

While it is possible to execute scans manually by copying the executable to a host and executing Scanafi, it is assumed that in most enterprise organizations a more automated management process would be desired. A typical strategy is to identify hosts within network segments to host the Scanafi executable and configure scheduled tasks and/or cron jobs to routinely execute the discovery process.

Another strategy would be to use existing system management frameworks (i.e. BigFix, Tanium, Puppet, Chef, and Microsoft SCCM) already in place to manage, patch, and configure systems in the environment to automatically deploy Scanafi. Discovery would be executed, then Scanafi would automatically be removed after completion.

1. Automated Result Processing

The typical approach is for Scanafi to be configured to execute on a host within the target network segment and upon completion of the scan, send the results directly to Venafi TPP using the WebSDK API. This approach simplifies any additional work required to process result files in order to gain visibility through TPP.

1. Standalone Result File Processing

Another approach is to deploy Scanafi to a host within the targeted network segment and execute it with parameters intended to scan the local subnet then output results to a JSON file. Upon completion, the results file can be sent to a central location for import into Venafi TPP using the WebSDK API. The executable and result file can then be removed from the target machine.

# Troubleshooting

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Log Message** | **Severity** | **Full Handshake  YES/NO** | **Additional Info** |
| 1 | Unable to get protocols for <IP Address> error:00ff0001 | Critical | No, only Client Hello | 0x00FF0001 => unable to connect, which prevents Scanafi from getting protocols and thus this error is reported. |
| 2 | Unable to complete scan for <IP Address> | Critical | No, only Client Hello | If you don't have enough information to proceed with scanning this error is shown, for example, if you cannot retrieve protocols (i.e. #1) you cannot scan for ciphers because those are per protocol and you have 0 protocols. |
| 3 | Received error: Handshake failed when connecting to host  | Critical | No, only Client Hello | **Connection Error**: Handshake can fail for many reasons. It will take effort to classify these situations and report errors for user's eyes instead of log for developers. |
| 4 | Received error: Unable to connect to host | Critical | No, only Client Hello | **Connection Error**: unable to connect, which prevents Scanafi from getting certificates |
| 5 | Received error: Connection timeout connecting to host | Critical | No, only Client Hello | **Connection Error**: connection timeout while connecting to host |
| 6 | Received error: Incomplete parameters provided when connecting to host | Critical | No, only Client Hello | **Connection Error**: incomplete set of parameters provided while connecting to host |
| 7 | Received error: Protocol not supported with host | Critical | No, only Client Hello | **Connection Error**: connection error with invalid list of protocols provided |
| 8 | Received error: Internal error received from host | Critical | No, only Client Hello | **Connection Error**: internal error while connecting to host, no specific reason |
| 9 | No error: successful connection to host | --- |  |  |
| 10 | Invalid code provided | Critical | No, only Client Hello | **Connection Error**: (unknown error) – i.e. unable to process/map error code returned by internal APIs |
|  | **Post Handshake Errors** |  |  | **Protocol – SSLv2, SSLv3, TLS 1.1, TLS 1.2, TLS 1.3 etc** |
| 11 | Ciphers for <IP Address> error: <Error #> | Minor |  | For a protocol negotiated during handshake, unable to retrieve ciphers from a host |
| 12 | Unable to get ECC named curves for <IP Address> error: <Error #> | Minor |  | For a protocol negotiated during handshake, unable to get ECC (Elliptic Curve Cryptography) named curves from a host |
| 13 | Unable to check secure renegotiation support for <IP Address> error: <Error #> | Minor |  | Unable to check if secure renegotiation is supported by a host |
| 14 | Unable to check compression support for <IP Address> error: <Error #> | Minor |  | Unable to check if compression is supported by a host |
| 15 | Unable to check client initiated renegotiation support for <IP Address> error: <Error #> | Minor |  | Unable to check if the client-initiated renegotiation is supported by a host |