Headless Scanner
User Guide
Version 1.2

May 18, 2023
# Table of Contents

About this Guide ........................................................................................................... 5
  About Qualys ............................................................................................................... 5
  Qualys Support .......................................................................................................... 5

Getting Started with Qualys Headless Scanner......................................................... 7
  Headless Scanner Background and Architecture ....................................................... 7

How does the Headless Scanner Work........................................................................ 9
  Headless Scanner Operation ...................................................................................... 9
  Headless Scanner Deployment Scenario ................................................................... 10

Headless Scanner Docker Image ................................................................................. 11
  Headless Scanner Docker Image Name ..................................................................... 11
  Accessing the Headless Scanner Docker image ....................................................... 12

Headless Scanner Operational Requirement ............................................................. 13

Configurations ............................................................................................................. 13
  Headless Scanner Input ............................................................................................. 13
    Scan Specification File .......................................................................................... 13
    Job Type ................................................................................................................ 14
    Scan Definition .................................................................................................... 14
    Reporting Options ................................................................................................. 15
    Scan Options ....................................................................................................... 19
    Callback Service ................................................................................................. 29
    Virtual Hosts Section .......................................................................................... 30
    Authentication Records Related Options .......................................................... 30
  SCAP/STIG Content Scan ....................................................................................... 31
  SCAP Scan Configuration ......................................................................................... 31
  SCAP Scan Output ................................................................................................... 32
    SSH Credentials Support .................................................................................... 34
    Custom Root Certificate Authorities ................................................................. 38
    Custom Shell Scripts .......................................................................................... 38
    Custom Shell Scripts Specification Parameters ............................................... 39
    Sample Scan Specifications ............................................................................... 42
    Customizing Headless Scanner Input and Output .............................................. 49

Launching Headless Scanner ...................................................................................... 50
  Running Headless Scanner with Separate Vulnerability Signatures ..................... 51
  Running Headless Scanner with a Read-Only File-system ....................................... 52
  Running Headless Scanner with Non-root User .................................................... 52
Headless Scanner Output ................................................................. 56
Scan Job Status File ........................................................................... 56
Message Section ................................................................................ 57
Results Section .................................................................................. 58
Error Reporting ................................................................................. 58
Scan Results File ............................................................................... 60
Troubleshooting Scenarios ................................................................. 61

Headless Scanner KBXML ................................................................. 65
Extract KBXML from Headless Scanner Image .................................. 65
KBXML File Attributes ..................................................................... 65

Headless Scanner Support for AWS ECS ......................................... 75
Headless Scanner AWS Fargate Deployment ...................................... 75
Headless Scanner ECS Deployment .................................................. 75
Launching a Scan ............................................................................... 81
Qualys Headless Scanner Docker Repository ..................................... 85
Custom MetaData Injection ............................................................... 85
Secondary Report .............................................................................. 88

Appendix A ......................................................................................... 91

Appendix B ......................................................................................... 113
About this Guide

This guide introduces Qualys Headless Scanner. Learn how you can perform vulnerability assessments using a standalone scanner tool. This guide caters to an advanced audience.

About Qualys

Qualys, Inc. (NASDAQ: QLYS) is a pioneer and leading provider of cloud-based security and compliance solutions. The Qualys Cloud Platform and its integrated apps help businesses simplify security operations and lower the cost of compliance by delivering critical security intelligence on demand and automating the full spectrum of auditing, compliance and protection for IT systems and web applications.

Founded in 1999, Qualys has established strategic partnerships with leading managed service providers and consulting organizations including Accenture, BT, Cognizant Technology Solutions, Deutsche Telekom, Fujitsu, HCL, HP Enterprise, IBM, Infosys, NTT, Optiv, SecureWorks, Tata Communications, Verizon and Wipro. The company is also a founding member of the Cloud Security Alliance (CSA).

For more information, please visit www.qualys.com.

Qualys Support

Qualys is committed to providing you with the most thorough support. Through online documentation, telephone help, and direct email support, Qualys ensures that your questions can be answered in the fastest time possible. We support you 7 days a week, 24 hours a day. Access support information at www.qualys.com/support/.
Getting Started with Qualys Headless Scanner

The Qualys Headless Scanner is designed to help organizations identify vulnerabilities. It allows you to perform vulnerability assessments without connecting to the rest of the Qualys backend platform. A Headless Scanner allows easy deployment of a network vulnerability assessment solution in a multitude of isolated network segments.

Having isolated network segments requires PCPs for each segment. Also, the geographic location of network segments might restrict access to outsiders. As a result, a docker containerized network vulnerability assessment solution became necessary.

Headless Scanner Background and Architecture

A Headless Scanner is a docker image that combines a scanning engine and vulnerability signatures in one container. This lets you quickly set up a functioning scanner solution with minimal uptime.

![Headless Scanner Architecture](image)

The Headless Scanner creates a docker container from the supplied docker image to perform the scan. The container performs the network target scan based on your provided Scan Specification file.

Upon completion of the scan, the result is returned to the container, and the container exits. With this basic operation, the Scan Specification file can be supplied from a cloud-based message queue, and the results can be stored in a cloud-based storage bucket.
Here is a component diagram of the Headless Scanner:

![Component Diagram]

For information on the Scan Specification file, see Scan Specification File.

Running a scanning solution in a container has some common limitations to scanning through a NAT device. Another potentially serious limitation is the inability to scan IPv6 targets because Docker containers don’t have an IPv6 NAT solution.
How does the Headless Scanner Work

The following sections describe the Headless Scanner workflow and how the Headless Scanner operates with input and output files.

Headless Scanner Operation

With its minimalistic interface, the Headless Scanner can be easily integrated into a variety of environments. The workflow for a Headless Scanner is described in this section.

The Headless Scanner operates with input and output files. The input file contains the complete description of a single scan job that the Headless Scanner is expected to process. Once a scan job is processed, the result of the scan job is stored as output files.

You can integrate Headless Scanner with your existing workflows and technologies to achieve a working solution. Using input and output files reduces the complexity of the Headless Scanner's interaction with any cloud technology.

Here is a high-level flow diagram showing the Headless Scanner workflow:
Headless Scanner Deployment Scenario

Here is one method for automating large and complex network scans using a Headless Scanner:

You create and provide the green components. Qualys provides the red component. The orange components are provided by you and Qualys, if needed.

Using this deployment scenario, you can maintain a scan task or target queue that needs to be processed. A number of Headless Scanner containers can be created based on the queue size. These instances can consume the scan target queue. The Headless Scanners can generate results for each target, which can be queued for processing. Results can be post-processed to remove irrelevant targets, re-queue failed scans, and add results to the vulnerability management database.

**Note:** The Headless Scanner image needs to be periodically updated to keep vulnerability detections up-to-date. Qualys publishes updates to the Headless Scanner image periodically. You can pull the new image directly from the Qualys Docker Registry or host a local registry to avoid redundancy.
The Headless Scanner is distributed as a docker image. There are several Headless Scanner images available. Each variation of the Headless Scanner image is designed to deliver specific features.

<table>
<thead>
<tr>
<th>Docker Image</th>
<th>Capabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsengine:&lt;VERSION&gt;</td>
<td>A lightweight docker image delivers the remote VM scanning capability.</td>
</tr>
<tr>
<td>hsengine:&lt;VERSION&gt;-kbxml</td>
<td>Headless Scanner image with Qualys vulnerability knowledge base. <strong>Note:</strong> We have deprecated this image and made the kbxml file part of the hsengine image.</td>
</tr>
<tr>
<td>hsengine-vulnsigs:&lt;VERSION&gt;</td>
<td>Vulnerability signatures as a docker image.</td>
</tr>
</tbody>
</table>

**Headless Scanner Docker Image Name**

The Headless Scanner docker image has a name in the following format:

hsengine:<VERSION>

The <VERSION> tag has the following format and meaning:

<VERSION_MAJOR>,<VERSION_MINOR>,<VERSION_PATCH>,<VULNSIGS_VERSION_ID>-<VERSION_BUILD>

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERSION_MAJOR</td>
<td>This is incremented whenever an incompatible change is made to the image.</td>
</tr>
<tr>
<td>VERSION_MINOR</td>
<td>This is incremented whenever a new compatible version of the Headless Scanner is released.</td>
</tr>
<tr>
<td>VERSION_PATCH</td>
<td>This is incremented whenever a new release is cut to address bug fixes.</td>
</tr>
<tr>
<td>VULNSIGS_VERSION_ID</td>
<td>The third component of the VULNSIGS version.</td>
</tr>
<tr>
<td>VERSION_BUILD</td>
<td>A Build number (the number after the symbol '-')</td>
</tr>
</tbody>
</table>

For example, If the Headless Scanner version is 1.2.0 and the VULNSIGS version is 2.4.797-2, the version assigned to Headless Scanner can be 1.2.0.797b2-1, i.e., hsengine:1.2.0.797b2-1 any subsequent builds with the two above components can get 1.2.0.797b2-2, 1.2.0.797b2-3, etc.
Accessing the Headless Scanner Docker image

The Headless Scanner image is available in DockerHub as a private image. You can access the image by opening a DockerHub account and requesting access. Refer to the following Headless Scanner DockerHub screenshot.

The Headless Scanner images are signed in order to guarantee the integrity of the image. You can use a verification method to ensure that the delivery pipeline has not modified the artifacts by the delivery manager.

To verify the signature, the following command can be used:

docker trust inspect --pretty qualys/hsengine:0.7.9.760b2-1

The following is the output of the command:

Signatures for qualys/hsengine:0.7.9.760b2-1

<table>
<thead>
<tr>
<th>SIGNED TAG</th>
<th>DIGEST</th>
<th>SIGNERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7.9.760b2-1</td>
<td>ba24815f962c36a6a4c496836b9b5e52a73391bf22d7a56f5a652ad678d4c752</td>
<td>qdevops24</td>
</tr>
</tbody>
</table>

List of signers and their keys for qualys/hsengine:0.7.9.760b2-1

<table>
<thead>
<tr>
<th>SIGNER</th>
<th>KEYS</th>
</tr>
</thead>
<tbody>
<tr>
<td>qdevops24</td>
<td>c5c865eab6a6</td>
</tr>
</tbody>
</table>
Headless Scanner Operational Requirement

The Headless Scanner container can be launched with assigned quotas for memory and CPU. The following table can be used as a guideline for planning resource allocation for Headless Scanner.

<table>
<thead>
<tr>
<th>Memory Quota</th>
<th>1GB</th>
<th>2GB</th>
<th>4GB</th>
<th>6GB</th>
<th>8GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Parallel Targets</td>
<td>24</td>
<td>45</td>
<td>90</td>
<td>135</td>
<td>180</td>
</tr>
</tbody>
</table>

Configurations

The following sections elaborate on the required configurations you must use to set up the Headless Scanner to perform the vulnerability scan.

Headless Scanner Input

The Headless Scanner input is an XML file called Scan Specification file, that describes the scan.

The Scan Specification file contains scan parameters, which include:
- scan targets
- scan performance settings
- scan behavior modifications
- target authentication information

Scan Specification File

A Scan Specification file is a complete definition of a single scan. It contains the following sections:
- Job Type
- Scan definition
- Scan Options
- Authentication Information
Appendix A contains the XSD schema of the Scan Specification.

When you start the scanning, the Headless Scanner container expects to find the Scan Specification file in a specific location in the container file system:

/usr/share/qualys/scan_spec.xml

Each scan job launched using the Headless Scanner is assigned a job ID that can be used to identify that scan job. The scan job ID is based on the scan job start time and the process ID of the Headless Scanner engine. The scan job ID looks like the following:

2020-08-13-05-43-38zutc-16

During the processing of the scan job, the Headless Scanner creates the directory under /usr/share/qualys with the name of the scan job ID, where all the scan job artifacts can be generated. This includes a results file and any other troubleshooting artifacts the scan job may produce.

Sample Scan Specification

Click Sample Scan Specification to know more details about scan specifications that you can use in various scenarios.

Job Type

The only job type currently supported is “scan”.

Scan Definition

The scan definition section describes the scan targets and scan type in the SCAN/CONF/TYPE element. Currently, the supported scan types are “vm”, “scap” and “pci”.

- vm: The value corresponds to vulnerability management scanning.
- pci: The value corresponds to Payment Card Industry (PCI) scanning module.
- scap: The value corresponds to SCAP content processing. See SCAP/STIG Content Scan section for details.

Targets are specified by listing IPv4 addresses, or address ranges.

Here is an example for scan definition section:

<SCAN>
  <TARGETS>
    <TARGET>
      <IP>10.20.31.80</IP>
      <IP>10.20.31.77</IP>
      <IP>10.20.32.190</IP>
      <IP>10.10.25.245</IP>
    </TARGET>
  </TARGETS>
</SCAN>
Target types currently supported are:
- IPv4
- IPv6
- FQDN

For FQDN targets, the Headless Scanner resolves the FQDN into an IPv4 address at the start of the scan and uses the first address throughout the entire scan. Also, a virtual host entry is automatically added to the scan configuration to facilitate the HTTP for the evaluation of FQDN targets.

## Reporting Options

### Metadata Options

This section gives detailed information you can use to control the report generated by the Headless Scanner. By default, the generated report contains no vulnerability meta-data such as vulnerability title, severity, etc. The report can be expected to be integrated into a workflow to enhance the vulnerability report with all necessary meta-data for human consumption outside of the Headless Scanner. You can implement this approach to allow various use cases for the Headless Scanner.

However, there are cases, such as a CI/CD pipeline, where the speed of report consumption is very important. For these types of use cases, it is possible to instruct Headless Scanner to provide some limited meta-data in the generated vulnerability report.

Since reported meta-data can be repeated for each instance of detected vulnerabilities and different use cases might use different pieces of meta-data, the report can be customized to include only the useful meta-data components. The report customization can be optionally configured using the `<REPORT>` section of the scan specification.

Here is an example of the REPORT section:

```xml
<REPORT>
  <META>
    <TITLE>1</TITLE>
    <SEVERITY>1</SEVERITY>
    <CVSS>1</CVSS>
    <CVSS3>1</CVSS3>
  </META>
</REPORT>
```
In this section, the Headless Scanner is being instructed to include all available at this point meta-data for vulnerabilities in the report.

Here is an example of a report generated with these settings:

```xml
<VULN>
  <META>
    <TITLE>SSL Certificate - Invalid Maximum Validity Date Detected</TITLE>
    <SEVERITY>2</SEVERITY>
    <CVSS>6.4</CVSS>
    <CVSS3>6.5</CVSS3>
  </META>
  <QID>38685</QID>
  <FLAGS>v</FLAGS>
  <PORT>5432</PORT>
  <PROTOCOL>tcp</PROTOCOL>
  <SSL>1</SSL>
  <RESULT>Certificate #0 ******************************** is valid for more than 825 days</RESULT>
  <INFO>
    <CERTIFICATE fingerprint="E8238BF9A2BD8EC9E4B182CD7A3AE7EEF50B7EA83E11F82ABB8A32103DC037A6"/>
  </INFO>
</VULN>
```

The following is the list of available meta-data along with severity details:

<table>
<thead>
<tr>
<th>Metadata</th>
<th>Description</th>
<th>Tag</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLE</td>
<td>Qualys assign this information and corresponds to the title of the vulnerability QID</td>
<td>&lt;TITLE&gt;</td>
<td>1</td>
</tr>
<tr>
<td>SEVERITY</td>
<td>Qualys assign this information, provides severity levels or rankings for vulnerabilities</td>
<td>&lt;SEVERITY&gt;</td>
<td>1</td>
</tr>
<tr>
<td>CVSS</td>
<td>MITRE assigns this information as the CVSS base score of the vulnerability</td>
<td>&lt; CVSS&gt;</td>
<td>1</td>
</tr>
<tr>
<td>CVSS3</td>
<td>MITRE assigns this information as the CVSSv3 base score of the vulnerability</td>
<td>&lt; CVSS3&gt;</td>
<td>1</td>
</tr>
</tbody>
</table>

**Note:** The entire REPORT tag can be omitted, in which case no meta-data can be present in the report generated by Headless Scanner.
Vulnerability Filtering Options

The Headless Scanner report format includes information about vulnerabilities.

As an example, let’s consider the following sequence.

1. A scan is launched, and a vulnerability XYZ has been reported as present on the target on UDP port 500.
2. A second scan is launched but with a limited number of UDP ports. As a result, the XYZ vulnerability is not present in the report. This doesn’t mean that the vulnerability has been fixed.
3. The vulnerable software has been updated on the target, and the XYZ vulnerability is no longer present.
4. A third scan is launched against the target, and the scan reports that vulnerability XYZ is not present on the target. The XYZ vulnerability can be reported with the ‘c’ flag (CANCEL). This allows a ticketing system to close the ticket and confirm that the vulnerability is no longer applicable.

It is possible to disable reporting of all vulnerabilities with the ‘c’ (CANCEL) flag.

The following section of the REPORT tag does precisely:

```xml
<REPORT>
  <QID_EXCLUSIONS>
    <CANCELLED>true</CANCELLED>
  </QID_EXCLUSIONS>
</REPORT>
```

Some vulnerabilities identified by the Headless Scanner are for inactive Linux kernel packages. Since vulnerabilities of inactive kernels do not apply to the target machine’s current state, Headless Scanner can suppress vulnerabilities for inactive kernels.

```xml
<REPORT>
  <QID_EXCLUSIONS>
    <INACTIVE_KERNEL>true</INACTIVE_KERNEL>
  </QID_EXCLUSIONS>
</REPORT>
```

Supported Report Format Option

It is possible to instruct the Headless Scanner to produce output in various formats.

OUTPUT_FORMAT- The Headless Scanner supports the following possible values and output formats.

- `json` - The Headless Scanner can produce JSON file output for each of the targets.
• xml- The Headless Scanner can produce XML files for target host results.

• json_and_xml- The Headless Scanner can produce both JSON and XML files for target host results. Please note that scan_status.xml file can refer to JSON files.

Here is an example of the configuration, followed by a snippet of the JSON formatted output.

```
<REPORT>
  <OUTPUT_FORMAT>json</OUTPUT_FORMAT>
</REPORT>

"scan_header": {
  "report_version": 13,
  "ip_address": "10.20.31.186",
  "dns_hostname": "",
  "nb_hostname": "",
  "mac_address": "",
  "os": "Linux fc11-31-186 2.6.29.6-213.fc11.i586 #1 SMP Tue Jul 7 20:45:17 EDT 2009 i686",
  "cpe": "",
  "start_time": "1657724608",
  "end_time": "1657724684",
  "devicetype": "Unknown"
},
"vulns": [
{
  "qid": 82040,
  "flags": "",
  "results": "#table cols="3"
ICMP_Reply_Type Triggered_By Additional_Information\nEcho_(type=0_code=0) Echo_Request
Echo_Reply\nUnreachable_ (type=3_code=3) UDP_Port_69
Port_Unreachable\nTime_Stamp_ (type=14_code=0) Time_Stamp_Request
15:04:05 GMT\nUnreachable_ (type=3_code=3) UDP_Port_7
Port_Unreachable\nIP_with_High_Protocol
Protocol_Unreachable_ (type=3_code=3) UDP_Port_5930
Port_Unreachable\nUnreachable_ (type=3_code=3) UDP_Port_53
Port_Unreachable\nUnreachable_ (type=3_code=3) UDP_Port_7170
Port_Unreachable\nUnreachable_ (type=3_code=3) UDP_Port_65535
Port_Unreachable\nUnreachable_ (type=3_code=3) UDP_Port_2
Port_Unreachable\nUnreachable_ (type=3_code=3) UDP_Port_65534
Port_Unreachable\n",
  "title": "ICMP Replies Received",
```
"cvss_base": "",
"cvss_temporal": "",
"cvss3_base": "",
"cvss3_temporal": "",
"severity": 1,
"category": "Ig",
"description": "ICMP (Internet Control and Error Message Protocol) is a protocol encapsulated in IP packets. ICMP's principal purpose is to provide a protocol layer that informs gateways of the inter-connectivity and accessibility of other gateways or hosts. We have sent the following types of packets to trigger the host to send us ICMP replies:
- Echo Request (to trigger Echo Reply)
- Timestamp Request (to trigger Timestamp Reply)
- Address Mask Request (to trigger Address Mask Reply)
- UDP Packet (to trigger Port Unreachable Reply)
- IP Packet with Protocol >= 250 (to trigger Protocol Unreachable Reply)
Listed in the "Result" section are the ICMP replies that we have received.

"workaround": "",
"solution": ""
}
]
}

Scan Options

The scan options specify scan performance parameters. This section corresponds to the options profile section in the traditional QualysGuard interface.

This section is intentionally modeled after the QualysGuard API Options Profile to allow automatic conversion of existing Options Profiles into Scan Options for Scan Specification file.

Click Sample Scan Options to see the sample example of scan options.

The scan options elements are described in the following sub-sections.

Scan Parameters

The scan parameters define the scope of the scan for all targets.

This section has the following structure:

- PORTS
- TCP_PORTS
### Configurations

#### Headless Scanner Input

- **TCP_PORTS_TYPE** with possible values

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Port Lists (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>Assign empty list as list of TCP ports to be probed.</td>
<td>N/A</td>
</tr>
<tr>
<td>custom</td>
<td>Assign the list specified in TCP_PORTS_ADDITIONAL section as the list of TCP ports to be probed.</td>
<td>N/A</td>
</tr>
<tr>
<td>light</td>
<td>Assign a common list of 160 ports as the list of TCP ports to be probed.</td>
<td>160</td>
</tr>
<tr>
<td>standard</td>
<td>Assign an extended common list of 1900 ports as the list of TCP ports to be probed.</td>
<td>1900</td>
</tr>
<tr>
<td>full</td>
<td>Assign all of the ports from 1-65535 as the list of TCP ports to be probed. (Full setting may increase scan time and is not recommended for Class C or larger networks)</td>
<td>1-65535</td>
</tr>
</tbody>
</table>

**Light TCP Ports View list:**


**Standard TCP Ports View List:**

Configurations
Headless Scanner Input

21

3000-3002, 3006-3007, 3010-3011, 3020, 3047-3049, 3080, 3127-3128, 3141-3145, 3180-3181,
3205, 3232, 3260, 3264, 3267-3269, 3279, 3306, 3322-3325, 3333, 3340, 3351-3352, 3355,
3372, 3389, 3421, 3454-3457, 3689-3690, 3700, 3791, 3900, 3984-3986, 4000-4002, 4008-4009,
4080, 4092, 4100, 4103, 4105, 4107, 4132-4134, 4144, 4242, 4321, 4333, 4434-4454,
4500-4501, 4567, 4580, 4626, 4651, 4660-4663, 4672, 4899, 4903, 4950, 5000-5005, 5009-
5011, 5020-5021, 5031, 5050, 5053, 5080, 5100-5101, 5145, 5150, 5190-5193, 5222, 5236,
5300-5305, 5321, 5400-5402, 5432, 5510, 5520-5521, 5530, 5550, 5554-5558, 5569,
5599-5601, 5631-5632, 5634, 5650, 5678-5679, 5713-5717, 5729, 5742, 5745, 5755, 5757,
5766-5767, 5800-5802, 5900-5902, 5977-5979, 5997-6053, 6080, 6103, 6110-6112, 6123, 6129,
6141-6149, 6253, 6346, 6387, 6389, 6400, 6455-6456, 6499-6500, 6515, 6543, 6558, 6588,
6660-6670, 6672-6673, 6699, 6767, 6771, 6776, 6789, 6831, 6883, 6912, 6939, 6969-6970,
7000-7021, 7070, 7080, 7099-7100, 7121, 7161, 7174, 7200-7201, 7300-7301, 7306-7308, 7395,
7426-7431, 7491, 7511, 7777-7778, 7781, 7789, 7895, 7938, 7999-8020, 8023, 8032, 8039,
8080-8082, 8090, 8100, 8181, 8192, 8200, 8383, 8403, 8443, 8450, 8484, 8500, 8732, 8765,
8886-8894, 8910, 9000-9002, 9005, 9043, 9080, 9090, 9098-9100, 9400, 9443, 9495, 9535,
9570, 9872-9876, 9878, 9889, 9989-10002, 10005, 10007, 10080-10082, 10101, 10202, 10204,
10520, 10607, 10666, 11000-11002, 11004, 11223, 12000-12002, 12076, 12223, 12287, 12345-
12346, 12361-12362, 12456, 12468-12469, 12631, 12701, 12753, 13000, 13333, 14237-14238,
15858, 16384, 16660, 16959, 16969, 17000, 17007, 17300, 18000, 18181-18186, 18190-18192,
18194, 18209-18210, 18231-18232, 18264, 19541, 20000-20001, 20011, 20034, 20200, 20203,
20331, 21544, 21554, 21845-21849, 22222, 22273, 22289, 22305, 22321, 22555, 22800, 22951,
23456, 23476-23477, 25000-25009, 25252, 25793, 25867, 26000, 26208, 26274, 26409, 27000-
27009, 27374, 27665, 29369, 29891, 30029, 30100-30102, 30129, 30303, 30999, 31336-31337,
31339, 31554, 31666, 31785, 31787-31788, 32000, 32768-32790, 33333, 33567-33568, 33911,
34324, 37651, 40412, 40421-40423, 42424, 44337, 47557, 47806, 47808, 49400, 50000-50001,
50505, 50766, 51102, 51107, 51112, 53001, 54320-54321, 57341, 60008, 61439, 61466, 62078,
65000, 65301, 65512

• TCP_PORTS_ADDITIONAL [OPTIONAL]

• HAS_ADDITIONAL boolean: The value true indicates that the list of ports in ADDITIONAL_PORTS element should be added to the list of TCP ports to be probed during the scan. The value of false indicates that no additional ports should be added.

• ADDITIONAL_PORTS: A comma separated list of TCP ports.

• THREE_WAY_HANDSHAKE boolean: The value true instructs the scanning engine to use full TCP handshake during probing as opposed to raw SYN packets. This option is useful when scanning through NAT devices.

• UDP_PORTS

• UDP_PORTS_TYPE with possible values

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Port Lists (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>Assign empty list as list of UDP ports to be probed.</td>
<td>N/A</td>
</tr>
</tbody>
</table>

21
## Configurations

### Headless Scanner Input

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>custom</strong></td>
<td>Assign the list specified in UDP_PORTS_ADDITIONAL section as the list of UDP ports to be probed.</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>light</strong></td>
<td>Assign a common list of 30 ports as the list of UDP ports to be probed.</td>
<td>30</td>
</tr>
<tr>
<td><strong>standard</strong></td>
<td>Assign a common list of 180 ports as the list of UDP ports to be probed.</td>
<td>180</td>
</tr>
<tr>
<td><strong>full</strong></td>
<td>Assign all the ports from 1-65535 as the list of UDP ports to be probed.</td>
<td>1-65535</td>
</tr>
</tbody>
</table>

**Light UDP Ports View List:**

7, 13, 17, 19, 37, 53, 67-69, 111, 123, 135, 137, 161, 177, 407, 464, 500, 517-518, 520, 1434, 1645, 1701, 1812, 2049, 3527, 4569, 4665, 5036, 5060, 5632, 6502, 7778, 15345

**Standard UDP Ports View List:**


- **UDP_PORTS_ADDITIONAL [OPTIONAL]**
  - HAS_ADDITIONAL boolean- The value true indicates that the list of ports in ADDITIONAL_PORTS element should be added to the list of UDP ports to be probed during the scan. The value of false indicates that no additional ports should be added.
  - ADDITIONAL_PORTS- A comma separated list of UDP ports.

- Adaptive Scanning- Set USE_AUTH_FOR_PORTLISTS to a Boolean value. The value true means to instruct the Headless Scanner engine to use the SSH authentication information to obtain precise TCP and UDP ports that are active on the target and adjust the list of ports the scanner is going to probe. As an example, if the scan was configured with 1-1024 ports to be probed, and the scan also had Unix authentication information supplied, then the scanner will first attempt to use SSH to log in to the scanner, fetch a list of ports that
have active services and use that list instead of 1-1024 list supplied originally to the scanner. If the authentication information is not provided or login to the target fails, then the port list remains unchanged (1-1024 in the example case).

For example, if the scan was configured with 1-1024 ports to be probed, and the scan also contained Unix authentication information, then the scanner can first attempt to use SSH to log in to the scanner, fetch a list of ports with active services and use that list instead of the list of 1-1024 provided initially to the scanner. The port list remains unchanged if authentication information is not provided or login fails.

- **SCAN_DEAD_HOSTS** boolean- The value true means that the scanner should attempt to run all tests even if the target doesn't seem to be alive. The default setting is to skip full evaluation of such targets.

### Performance Parameters

The scan performance section under the PERFORMANCE tag has the following structure:

- **PARALLEL_TARGETS**- positive integer- The maximum number of targets that can be scanned simultaneously by the Headless Scanner. The default value can be 10 if this setting is missing.

- **TOTAL_PROCESSES**- positive integer- The maximum number of probes to be run in parallel per target.

- **HTTP_PROCESSES**- positive integer- The maximum number of HTTP probes to be run in parallel per target.

- **PACKET_DELAY**- enum- This is the delay between groups of packets sent to each scanned host. A short delay means that packets are sent more frequently. A long delay means that packets are sent less frequently. Each performance level has been assigned a delay time appropriate for the performance level.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>No delay</td>
</tr>
<tr>
<td>Short</td>
<td>0.3 second delay</td>
</tr>
<tr>
<td>Medium</td>
<td>0.6 second delay</td>
</tr>
<tr>
<td>Long</td>
<td>1 second delay</td>
</tr>
<tr>
<td>Maximum</td>
<td>2 second delay</td>
</tr>
</tbody>
</table>

- **PORT_SCANNING_AND_HOST_DISCOVERY**- This setting determines the aggressiveness (parallelism) of port scanning and host discovery at the port level. Lowering the intensity level has the effect of serializing port scanning and host discovery. This is useful for certain network conditions like cascading firewalls and lower scan prioritization on the network.
Port scanning and host discovery typically place a maximum load on firewall state tables during a scan. We recommend that you reduce the intensity level if you are scanning through a firewall. Unauthenticated scans see more of a performance difference using this option. The following are the possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>25%</td>
</tr>
<tr>
<td>Low</td>
<td>50%</td>
</tr>
<tr>
<td>Medium</td>
<td>75%</td>
</tr>
<tr>
<td>Normal</td>
<td>100%</td>
</tr>
</tbody>
</table>

In order to understand this control, split the scan into three phases discovery, port-scanning, and general VM scanning.

In the discovery phase, the scanner attempts to determine if the host is alive by sending various probes like ICMP and TCP connection requests.

In the port-scanning phase, the scanner sends a large number of TCP SYN packets to a large number of ports. As mentioned above, the first two phases affect firewalls the most because of the large number of parallel connection requests. This setting can be used to temporarily limit the number of parallel IPs that can be engaged in a scan but only in the first two phases.

The following table illustrates the functionality through examples:

<table>
<thead>
<tr>
<th>Intensity Level</th>
<th>Number of parallel IPs to be scanned</th>
<th>Host Discovery Divisor</th>
<th>Effective number of parallel IPs in discovery and port-scanning phases</th>
<th>Effective number of parallel IPs in the scanning phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal (100%)</td>
<td>60</td>
<td>1</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>Medium (75%)</td>
<td>60</td>
<td>3</td>
<td>20</td>
<td>60</td>
</tr>
<tr>
<td>Low (50%)</td>
<td>60</td>
<td>6</td>
<td>10</td>
<td>60</td>
</tr>
<tr>
<td>Minimum (25%)</td>
<td>60</td>
<td>12</td>
<td>5</td>
<td>60</td>
</tr>
</tbody>
</table>

**LOAD_BALANCER_DETECTION**

boolean- With enabled, the scanner can attempt to identify load balancers and the number of Web servers behind them.

**VULNERABILITY_LIST**

This is a way to limit the vulnerability scope of the scan to a selected list of vulnerability detections to limit the scan duration. The vulnerability detections can be specifically included in the scan report by listing them as a comma-separated list as the value of the SELECTED_QIDS element. Or, vulnerability detections can be excluded in a similar format using the EXCLUDED_QIDS element.
Here is an example:

```xml
<VULNERABILITY_LIST>
    <SELECTED_QIDS>105297,115263,105053,38307,45038</SELECTED_QIDS>
    <EXCLUDED_QIDS>38117</EXCLUDED_QIDS>
</VULNERABILITY_LIST>
```

**PASSWORD_BRUTE_FORCING**

Select the level and scope of password brute forcing performed by scans.

- **THROTTLE_INTERVAL** - It is an optional interval in seconds to wait between brute force attempts. This can be used to avoid account lockups for devices that have protection against brute force.

The value is an integer in the range of 0-60.

Here is an example of how you can define a list of weak passwords that can be specified for the scan.

```xml
<PASSWORD_BRUTE_FORCING>
    <THROTTLE_INTERVAL>1</THROTTLE_INTERVAL>
    <SYSTEM>
        <HAS_SYSTEM>1</HAS_SYSTEM>
        <SYSTEM_LEVEL>Exhaustive</SYSTEM_LEVEL>
    </SYSTEM>
    <CUSTOM_LIST>
        <CUSTOM>
            <ID>1002</ID>
            <TITLE>FTP-weak</TITLE>
            <TYPE>FTP</TYPE>
            <LOGIN_PASSWORD>L:anonymous,P:none</LOGIN_PASSWORD>
        </CUSTOM>
        <CUSTOM>
            <ID>1001</ID>
            <TITLE>SSH-weak</TITLE>
            <TYPE>SSH</TYPE>
            <LOGIN_PASSWORD>L:root,P:toor</LOGIN_PASSWORD>
        </CUSTOM>
        <CUSTOM>
            <ID>1003</ID>
            <TITLE>Windows-weak</TITLE>
            <TYPE>Windows</TYPE>
            <LOGIN_PASSWORD>L:administrator,P:admin</LOGIN_PASSWORD>
        </CUSTOM>
    </CUSTOM_LIST>
</PASSWORD_BRUTE_FORCING>
```
ADDL_CERT_DETECTION
You can use this option to instruct the scan engine to find certificates in more locations using authentication, e.g., Apache, Tomcat, Jboss, Java KeyStore, and Windows IIS. The authentication is required.

LITE_OS_SCAN
When this option is selected, and QID 45017 is included in the scan, OS detection testing is reduced during the host discovery phase before vulnerability testing. Telnet, msrpc, ntp, and other expensive OS detection methods are excluded.

CUSTOM_HTTP_HEADER
When authorized scans run, a custom value can be set to drop defenses (such as logging, IPs, etc.) Here is an example of how a custom HTTP header and a value can be set:

    <CUSTOM_HTTP_HEADER>
    <VALUE>Qualys_Scan</VALUE>
    <DEFINITION_KEY>Auth</DEFINITION_KEY>
    <DEFINITION_VALUE>b23afe3b438a20812cd3ffe3</DEFINITION_VALUE>
    </CUSTOM_HTTP_HEADER>

HOST_ALIVE_TESTING
When this option is enabled, scan testing report hosts found alive during the discovery or port scanning process. Another testing cannot occur.

WEB_CRAWL
This option disables web crawling during the scan. The possible values are true, false, 0, and 1. If web crawling is disabled, web server discovery and fingerprinting can occur, but web-specific modules cannot do further scanning.

WEB_REQUEST_LIMIT
This option limits the number of web requests. The scanner cannot perform further web evaluation once it reaches the limit indicated in this field. QID 87382 can be posted to notify you that the scanner didn't complete all the web application-related tests.
Additional Scan Parameters

- HOST_DISCOVERY

- TCP_PORTS
  - STANDARD_SCAN
    Host Discovery TCP Ports View list:
    21-23, 25, 53, 80, 88, 110-111, 135, 139, 443, 445
  - TCP_ADDITIONAL [OPTIONAL]
    - HAS_ADDITIONAL boolean: The value true indicates that the list of ports in ADDITIONAL_PORTS element should be added to the list of TCP ports to be probed during host discovery phase of the scan. The value of false indicates that no additional ports should be added.
    - ADDITIONAL_PORTS: A comma separated list of TCP ports.

- UDP_PORTS
  - STANDARD_SCAN
    Host Discovery UDP Ports View list:
    53, 111, 135, 137, 161, 500
  - CUSTOM_PORT [OPTIONAL]
    - BLOCK_RESOURCES- In some cases, network scanning has an adverse effect on individual servers or networks due to legacy or vulnerable software causing unexpected results. A TCP connection to an old service could leave the service unresponsive. In these cases, it may be advisable to instruct the scanner to avoid accessing some ports on all or some of the targets. This can be achieved by using the BLOCK_RESOURCES section.
    - BLOCK_RESOURCES- This is the main element of this section.
      - IP_LIST: This element contains a comma-separated list of IP addresses and IP ranges in the form x.x.x.x-y.y.y.y or a single special entry ‘target’. The target keyword indicates that the blocked ports should be applied to all scanned targets.
      - PORT_LIST: This element should contain a comma-separated list of ports that the scanner should avoid.

The blocking logic applies on a set defined by the cartesian product of IP_LIST and PORT_LIST.

Here is an example:
<ADDITIONAL>
...
<BLOCK_RESOURCES>
  <IP_LIST>target</IP_LIST>
  <PORT_LIST>443</PORT_LIST>
</BLOCK_RESOURCES>
...
</ADDITIONAL>

- PACKET_OPTIONS
  - IGNORE_FIREWALL_GENERATED_TCP_RST - When this option is enabled, the scan engine will ignore firewall-generated TCP RST packets.
  - IGNORE_ALL_TCP_RST - When this option is enabled, the scanner can ignore TCP RST packets from any source for host discovery purposes. With the option turned off, TCP RST packets received from the target are considered indicators that the target is an alive and functioning host.
  - IGNORE_FIREWALL_GENERATED_TCP_SYN_ACK - When this option is enabled, the scan engine can ignore firewall-generated TCP SYN-ACK packets.
  - NOT_SEND_TCP_ACK_OR_SYN_ACK_DURING_HOST_DISCOVERY - When this option is enabled, the scan engine cannot send TCP ACK or SYN-ACK packets during host discovery.

Note: For scans, this option applies only if the 3-way handshake option is disabled.

Overall Scan Control
The following options allow control over some aspects of the entire scan.

- SCAN_DURATION_LIMIT - This option allows an over-scan duration upper limit to be specified. The value of this option is in seconds. The scan cannot be allowed to proceed longer than the specified amount of seconds. When the scan is interrupted due to the time limit enforcement, the scan status can contain indicators that some targets have not been fully evaluated. The entries for targets that were canceled due to time limitations can look like this:

  <TARGET id="2001:470:8418:141f::a14:1fba" status="error">
  <ERROR code="4">
    <![CDATA[scan cancelled]]>
  </ERROR>
  </TARGET>
This behavior can change in the near future to have a more consistent format:

```xml
<TARGET id="2001:470:8418:141f::a14:1fba" status="cancelled">
  <ERROR code="4">
    <![CDATA[
      scan cancelled
    ]]>  
  </ERROR>
</TARGET>
```

### Callback Service

Some vulnerability tests require the ability of the target system to connect back to the scanner to confirm the presence of the vulnerability. However, typically due to security concerns, inbound connections to the scanner are not allowed.

The Headless Scanner can be configured to interact with a callback service to facilitate testing for such exposures. The callback service interaction workflow is illustrated in the following sequence diagram.

Headless Scanner can be optionally configured to use a callback service in the scan specification file using the following structure:

- **TYPE**: It specifies the type of callback service. Currently, the only supported value is ‘tsunami’. This refers to the tsunami callback service open-source project. In the future, support for other services can be added.
- **RECORD_URL**: This should contain the URL that should be sent to the target. The callback service can record any attempt to access the URL.
- **POLL_URL**: This should contain the scanner’s URL to check if the target has accessed the URL.
- **DNS_PREFIX**: (optional) Some protocols require complex interaction between the client and the server. For these protocols, the callback service uses FQDN-based callback detection. The DNS_PREFIX setting is used to generate one-time FQDNs for verifying callback attempts. If a name is resolved, the target intended to connect to the callback service.
Configurations
Headless Scanner Input

- `POLL_REQUEST_COUNT`: (optional) a number of requests that the scanner should check with callback service for a callback confirmation. The maximum value for this setting is 10.

- `POLL_REQUEST_TIMEOUT`: (optional) a timeout in milliseconds that each request to the callback service should wait for a response. The maximum value for this setting is 60000.

Here is an example:

```
<SCAN>
...
<CALLBACK_SERVICE>
  <TYPE>tsunami</TYPE>
  <RECORD_URL>http://192.41.13.116:8881</RECORD_URL>
  <POLL_URL>http://127.0.0.1:8880</POLL_URL>
</CALLBACK_SERVICE>
</SCAN>
```

Virtual Hosts Section
It is possible to provide HTTP virtual host information to help the scanner correctly assess HTTP servers and applications hosted by them.

Virtual host mappings can be provided in the following way:

```
<VIRTUAL_HOST_LIST>
  <VIRTUAL_HOST>
    <IP>10.10.10.121</IP>
    <PORT>80</PORT>
    <FQDN>webfront.acme.com</FQDN>
  </VIRTUAL_HOST>
</VIRTUAL_HOST_LIST>
```

Authentication Records Related Options
Normally Headless Scanner engine requires that authentication records are strictly associated with only one target. This means that while scanning a target, only one set of credentials can be used. This is done to avoid causing account lockout during the scan by multiple incorrect authentication attempts.

However, in some cases, targets can change IP addresses, and it is impossible to associate credentials with the target IP address. Also, the account lockout policy is forgiving enough to allow a few invalid authentication attempts. In these cases, it is possible to relax the strict one-to-one association requirement. The following block instructs the engine not to require strict mapping of authentication records to target IPs.
<AUTHENTICATION_OPTIONS>
    <MULTIPLE_AUTH_RECORDS>true</MULTIPLE_AUTH_RECORDS>
</AUTHENTICATION_OPTIONS>

SCAP/STIG Content Scan

Qualys Headless Scanner can be configured to process SCAP 1.1/1.2 content for evaluating policy compliance of targets. SCAP content is a self-contained expression of security policy rules, methods for checking compliance with the policy, and a definition of the scope of the security policy. Security policies are defined by XCCDF, checking mechanisms by OVAL, and score by CPE. These components can be combined into a SCAP data stream or referred to separately. The Scan Specification file can be configured to provide either an SCAP data stream or individual files and evaluate security policies.

SCAP Scan Configuration

SCAN/CONF/TYP e element must contain a scap value to instruct the Qualys Headless Scanner to process SCAP contents.

SCAP content can be specified in two major methods:

- SCAP data-stream
- Individual component files

A reference to the SCAP content and the XCCDF profile is required in both cases in the child QSCAP element of HEADLESS_SCANNER_JOB.

For the SCAP data-stream, the QSCAP_CONTENT element's name attribute should have a scap-ds value, and the type attribute should have a file name for the data stream. The Scan Specification file and the SCAP data-stream XML file should be in the same directory.

<QSCAP>
    <XCCDF_PROFILE>xccdf_gov.nist_profile_xccdf_gov.nist.validation.unixFile
        Test_profile_file</XCCDF_PROFILE>
    <QSCAP_CONTENT name="scap-ds" type="filename">unix_file_test-
datastream.xml</QSCAP_CONTENT>
</QSCAP>

To specify SCAP content by its components, four or five QSCAP_CONTENT elements can be used. Each QSCAP_CONTENT should refer to a component of SCAP.

The XCCDF file should be specified by the QSCAP_CONTENT element with a name attribute containing the xccdf value.
The CPE dictionary file should be specified by the QSCAP_CONTENT element with a name attribute containing the cpe-dictionary value.

The CPE OVAL file should be specified by the QSCAP_CONTENT element with a name attribute containing the cpe-oval value.

The main OVAL file should be specified by the QSCAP_CONTENT element with a name attribute containing the oval value.

Optionally a secondary OVAL file can be specified. It should be specified by the QSCAP_CONTENT element with a name attribute containing the other-oval value.

```xml
<QSCAP>
  <XCCDF_PROFILE>xccdf_org.cisecurity.benchmarks_profile_Level_1_-
  _Workstation</XCCDF_PROFILE>
  <QSCAP_CONTENT name="xccdf"
type="filename">sql.xccdf.xml</QSCAP_CONTENT>
  <QSCAP_CONTENT name="oval"
type="filename">sql.oval.xml</QSCAP_CONTENT>
  <QSCAP_CONTENT name="cpe-dictionary" type="filename">openscap-cpe-
dict.xml</QSCAP_CONTENT>
  <QSCAP_CONTENT name="cpe-oval" type="filename">openscap-cpe-
oval.xml</QSCAP_CONTENT>
</QSCAP>
```

### SCAP Scan Output

Headless Scanner processes the input SCAP content and produces XML results with the results of the XCCDF policy evaluation. Each rule is that evaluation results are produced along with supporting evidence.

Headless Scanner will support the various output formats in future versions, including JSON, native XCCDF, and SCAP 1.2.

```xml
<?xml version="1.0"?>
<HOST>
  <REPORT_REF>qscap/1671465475.39</REPORT_REF>
  <REPORT_VERSION>13</REPORT_VERSION>
  <TRACKING_METHOD>ip</TRACKING_METHOD>
  <IP_ADDRESS>10.xx.xx.xx</IP_ADDRESS>
  <MAC_ADDRESS>00:xx:xx:xx:xx:xx</MAC_ADDRESS>
  <OS>CentOS Linux 7.2.1511</OS>
  <DEVICETYPE>Unix</DEVICETYPE>
  <CPE>cpe:/o:centos:centos_linux:7.2.1511::</CPE>
  <TRUSTED_SCANNING>
```
<TYPE>SSH</TYPE>
<STATUS>1</STATUS>
.AUTH_NAME>sample auth record</AUTH_NAME>
<PORT>22</PORT>
<ERROR_MSG>Completed login for host=10.xx.xx.xx, user=root</ERROR_MSG>
</TRUSTED_SCANNING>
<COMPLIANCE>
<DP>
<K>xccdf_moc.elpmaxe.www_rule_1</K>
<V>2</V>
<E>
<d ident="oval:x:def:1" r="2">
<AND r="2">
<t ident="oval:x:tst:1" r="2">
<a><![CDATA[result IS ]]]></a>
<i id="oval:1:obj:1" flag="complete" item_name="sql57_item">
 <version>5.7</version>
 <result datatype="record">
  <field name="BANNER" datatype="string">Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production</field>
  <field name="BANNER_FULL" datatype="string">Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production</field>
  <field name="BANNER_LEGACY" datatype="string">Version 19.17.0.0.0</field>
 </result>
 </i>
</t>
</AND>
</d>
</DP>
</COMPLIANCE>
</HOST>

This section contains authentication credentials that you can use for the scanning engines to perform a target assessment of your organizational environment.
Here is a sample authentication section.

```xml
<AUTHS>
  <NT>
    <RECORD>
      <AUTH_NAME>VzNLVEVTVC0x</AUTH_NAME>
      <AUTH_ID>18861</AUTH_ID>
      <USERNAME>YWRtaW5pc3RyYXRvcg==</USERNAME>
      <IP_RANGE>10.20.30.1-10.255.255.255</IP_RANGE>
      <PASSWORD><!-- BASE64 ENCODED password--></PASSWORD>
    </RECORD>
    <RECORD>
      <AUTH_NAME>bWwtMms4eDY0ZGM=</AUTH_NAME>
      <DOMAIN>bWwtMms4eDY0ZGM=</DOMAIN>
      <USERNAME>YWRtaW5pc3RyYXRvcg==</USERNAME>
      <PASSWORD><!-- BASE64 ENCODED password--></PASSWORD>
      <HOSTID>TRUE</HOSTID>
      <AUTH_ID>1234</AUTH_ID>
    </RECORD>
  </NT>
</AUTHS>

SSH Credentials Support

This section elaborates on username password credentials, private key-based SSH authentication, and openSSH certificate-based SSH authentication.

The following is the structure of the SSH2 authentication record in a BNF format.

```xml
<SSH2>
  <IP_RANGE>
  <AUTH_ID> (optional) base64 encoded short authentication record identifier
  <AUTH_NAME> (optional) base64 encoded authentication record name
  <PORTS> (optional) a comma separated list of ports that the record should be attempted
  <TARGET_TYPE> (optional) base64 encoded device type indicator
  <USERNAME> base64 encoded username
  <USERPASS> (only used for SSH-"password", SSH-"keyboard-interactive", telnet, rlogin)
  <PW_INFO> (see below)

```
<CLEARTEXT> (BOOL optional) If set to TRUE will allow the scanner to send the password over non-encrypted channel (telnet)

<PASSWORD2> (optional, Cisco only)

<PW_INFO> (structure defined below) this password is used for 'enable' command

<SUDO_INFOS> (optional)

<SUDO_INFO> (optional, only used for root escalation, may be repeated)

<PW_INFO> (optional, structure defined below) used for sudo escalation

<CMD> (optional) base64 encoded command to be used for sudo escalation

<PRIV_KEYS> (optional)

<CERTPK_INFO> (defined below, may be repeated)

<KRB5> (optional)

<REALM> base64 encoded Kerberos Realm

<KDC> (optional) base64 encoded FQDN of the KDC

<PW_INFO> (optional, defined below, Kerberos password, mutually exclusive with CERTPK_INFO)

<CERTPK_INFO> (optional, defined below, Kerberos PKINIT user cert, mutually exclusive with PW_INFO)

<ROOT_CERTS> (optional)

<CERT> (repeatable)

<CERTPK_INFO>

<PRIVATE_KEY> (optional)

<PW_INFO> (optional, defined below, pass phrase of private key)

<CERT> (optional)

<PW_INFO>

<PASSWORD> (optional) base64 encoded password
**Username Password Credentials**

Usernames and passwords are the most primitive types of credentials in which authentication is based on secret passwords.

Credentials of this type look like this:

```xml
<SSH2>
  <RECORD>
    <AUTH_ID>MTAyMDMyMjAzNw==</AUTH_ID>
    <AUTH_NAME>MTAyMDMyMjAzN19hdXRo</AUTH_NAME>
    <CLEARTEXT>false</CLEARTEXT>
    <PORTS>22</PORTS>
    <IP_RANGE>10.20.32.203</IP_RANGE>
    <USERNAME>ZWMyLXVzZXI=</USERNAME> <!-- base64 encoded username -->
    <USERPASS>
      <PW_INFO>
        <PASSWORD>ZWMyLXVzZXI=</PASSWORD> <!-- base64 encoded password -->
      </PW_INFO>
    </USERPASS>
  </RECORD>
</SSH2>
```

**Private Key Based SSH Authentication**

```xml
<SSH2>
  <RECORD>
    <AUTH_ID>Mjg2MDAy</AUTH_ID>
    <AUTH_NAME>U2FtcGxlIHByaXZhdGUga2V5IGF1dGgcmVjb3Jk</AUTH_NAME>
    <USERNAME>ZWMyLXVzZXI=</USERNAME>
    <PRIV_KEYS>
      <CERTPK_INFO>
        <PRIVATE_KEY>LS0tLS1...</PRIVATE_KEY> <!-- base64 encoded private key -->
      </CERTPK_INFO>
    </PRIV_KEYS>
  </RECORD>
</SSH2>
```

Base64 encoding is required for the private key.
You can obtain the private key's base64 encoded value by using the following command:

```
openssl base64 -A -in ~/.ssh/id_rsa
```

### OpenSSH Certificate Base SSH Authentication

```
<SSH2>
  <RECORD>
    <AUTH_ID>MTQzOTgx</AUTH_ID>
    <AUTH_NAME>QXV0aCBJRCAxNDUyMDM=</AUTH_NAME>
    <CLEARTEXT>false</CLEARTEXT>
    <PORT>22</PORT>
    <IP_RANGE>10.xx.xx.xx</IP_RANGE>
    <USERNAME>ZWMyLXVzZXI=</USERNAME>
    <PRIV_KEYS>
      <CERTPK_INFO>
        <PRIVATE_KEY>LS0....t</PRIVATE_KEY>
        <CERT>c3NoL....z</CERT>
      </CERTPK_INFO>
    </PRIV_KEYS>
  </RECORD>
</SSH2>
```

The private key and the certificate values are base64 encoded.

The OpenSSH certificate to be base64 encoded usually has the following form:

```
ssh-rsa-cert-v01@openssh.com
AAAABHNZ+aC1yc2EtY2YdC12MDFAb3BlbnNzZjbi20AAAAgG16/6YBocI3/nH+5NNeGYeMr
cslk0GyGfdOepk0/r6AAAADAQABAAMAAABQC4Ci8P10sya9csUbZQETkT5NxFDxGmVeFczl
dcRjkSRyCzhdpkjyAaWR8tB_kvMattr+466Nu7q+13vTkGyCC53sPdPMHbDH3geYi8l7aZGG
KFMGgILVMjWwl8OWalFblSOFPTnTjCjGZjyDvCPlcKOijjCa77bFo/jmpR4P4RbrYGDXF8dgo
+rc84FrN4Y4zUzy6W7WqWOjr9g459TCAmfU2eOtFM2E910KJuP5byBb7mnA6ekcQsc
tlEdQBFohwZ9JZfnLx4uhILTc14UV6Ym5W7RnpCQzkZff9pc0JiGN7j4XVBDF1P4z/ErTQ
CGkzKe1309aFmRAAAAAAAAAAABCIy2QAAAAAYYAGxvY2FsaG9zdC5xd
WFseXMuY29TAAAAGIU/jAAAAGYhZQEmAAAAAAAADGgAAAAATZWNkc2Et2hLMIuaXN0cDI1NgAAAAAhuaXN0cD11NgaAEEEEEfc6GVgyquX+0nMaW3RWH6r9x
TZNZ8nyY1uwT79FR2cWO04nNiBKe+rGCq7AzAQ883QhSMeFTE5AVjwcAAAAAGMAA
ATZWNkc2Et2hLMIuaXN0cDI1NgAAAAAEGAAxkWD78c2n2R7YThCyD1TKYTmGj4j
awegl41LCYgrAAAAAgD2PpTUhqq3tYeQxeX0KFvvT3mDB6VFz0YINz0bYsOPQ=
```

```bash
sshuser@workstation.qualys.com
```
Target Types
SSH authentication records may contain special indicators instructing the scan engine to use commands that apply to the target type while logging into the target system using SSH.

The default command set used by the scan engine is a subset of the Linux/Unix command set. When the authentication record contains a target type indicator, such as BIG-IP for F5 devices, the scanner can use commands that apply to F5 BIG-IP systems.

The list of supported device types are listed in the following table:

<table>
<thead>
<tr>
<th>Device Type String</th>
<th>Device Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A10</td>
<td>A10 devices</td>
</tr>
<tr>
<td>BIG-IP</td>
<td>F5 BIG-IP devices</td>
</tr>
<tr>
<td>NetApp_PC</td>
<td>NetApp Device</td>
</tr>
<tr>
<td>FORTIANALYZER</td>
<td>Fortinet Forti Analyzer</td>
</tr>
<tr>
<td>CISCO_ISE</td>
<td>Cisco Identity Services Engine</td>
</tr>
<tr>
<td>HP_COMWARE</td>
<td>HP Comware Software</td>
</tr>
</tbody>
</table>

The device type string is included in the SSH authentication record as the value of SSH_DEVICE_TYPE element in base64 encoded format.

Custom Root Certificate Authorities
To supply a list of custom root CA certificates ROOT_CA_CERTS element can be used with a list of base64 encoded PEM certificates CERT elements.

Here is an example.

Custom Root CA List

```
<ROOT_CA_CERTS>
  <ROOT_CA>LS0tLS1CR....</ROOT_CA>
  <ROOT_CA>LS0tLS1CR...</ROOT_CA>
</ROOT_CA_CERTS>
```

Custom Shell Scripts
Custom shell scripts are supported to allow the collection of custom data points from scan targets. Multiple custom shell scripts can be specified in the scan specification file to be executed on the target, and the output of scripts can be captured and reported in the scan results.

Here is a sample input for specifying custom shell script:

```
<CUSTOM SHELL SCRIPTS>
```
<SHELL_SCRIPT id="1" title="Collect /etc/passwd file">
  <META>
    <DESCR> Collect contents of /etc/passwd file in order to identify if any users ... </DESCR>
    <APPLY_OS>.*Linux</APPLY_OS>
    <EXCLUDE_OS>RedHat</EXCLUDE_OS>
  </META>
  <SCRIPT shell="/bin/bash" contents="text|base64">
    <![CDATA[cat /etc/passwd]]>
  </SCRIPT>
</SHELL_SCRIPT>

<SHELL_SCRIPT id="2" title="Collect IPv4 Forwarding state">
  <META>
    <DESCR> Collect the status of IPv4 packet forwarding </DESCR>
    <APPLY_OS>.*Linux</APPLY_OS>
    <EXCLUDE_OS>Fedora</EXCLUDE_OS>
  </META>
  <SCRIPT shell="/bin/bash" contents="text|base64">
    <![CDATA[cat /proc/sys/net/ipv4/ip_forward]]>
  </SCRIPT>
</SHELL_SCRIPT>
</CUSTOM_SHELL_SCRIPTS>

**Custom Shell Scripts Specification Parameters**

**Title**
The title is supposed to be a one line simple title assigned to the script.

**Description**
The description of the custom script contains a larger text or HTML based description of the custom shell script that can help someone process the output of the custom script execution.

**Apply OS**
This field indicates that the custom shell script should be run only on operating systems that satisfy this field's regular expression. This field is optional. If the field is not present, then the script should be run on all target OS not excluded by the exclusion filter.

**Exclude OS**
This field indicated the operating systems where the script should be skipped because it may not be applicable. This field is optional. If the field is not present, the script has no exclusions.

**Note:** If the script execution is excluded due to one of the filters, the QID1208 results cannot contain any results for that shell script.

**Shell Script**

The custom script can be provided in this section. The id attribute contains an identifier to distinguish multiple custom scripts. The shell attribute identifies the shell to be used for custom script execution. Currently, only /bin/bash is supported.

The contents of the custom shell script element can be base64 encoded to avoid collision with XML keywords and delimiters.

**Output**

The results of custom shell scripts can be reported as results of QID 1208. The results look like the input with some additional elements.

Here is an example:

Custom Shell Script Output -

```xml
<CUSTOM_SHELL_SCRIPTS>
  <SHELL_SCRIPT id="1" title="Collect /etc/passwd file">
    <META>
      <DESCR>Collect contents of /etc/passwd file in order to
```
identify if any users ...
</DESCR>

<APPLY_OS>.*Linux</APPLY_OS>
<EXCLUDE_OS>RedHat</EXCLUDE_OS>
</META>

<OS>Ubuntu 18.04.6 LTS</OS>
<USER>root</USER>

<SCRIPT shell="/bin/bash" contents="text|base64">
 <![CDATA[cat /etc/passwd]]>
</SCRIPT>

<STDOUT><![CDATA[
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/bin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/bin:/sbin/nologin
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/sbin/nologin
dbus:x:81:81:System message bus:/sbin/nologin]]>
</STDOUT>

<STDERR/>
</SHELL_SCRIPT>

<SHELL_SCRIPT id="2" title="Collect IPv4 Forwarding state">

<DESCR>Collect the status of IPv4 packet forwarding</DESCR>

<APPLY_OS>.*Linux</APPLY_OS>
<EXCLUDE_OS>Fedora</EXCLUDE_OS>
</META>

<OS>Ubuntu 18.04.6 LTS</OS>

41
Sample Scan Specifications

This section lists scan specifications that can use in various scenarios. Each scenario has certain specifics that the scan specification can be adapted for.

CI/CD

The scan time is expected when the Headless Scanner is utilized in a CI/CD workflow. To arrange the scan to be complete in the shortest amount of time, the probe(module) parallelism can be increased while the target parallelism is maintained low. This assumes that the CI/CD pipeline is scanning one target at a time.

The following scan specification can be used:

CI/CD Scan Job:

```xml
<?xml version="1.0"?>
<HEADLESS_SCANNER_JOB>
  <JOB_TYPE>scan</JOB_TYPE>
  <SCAN>
    <TARGETS>
      <TARGET>
        <IP>10.0.0.168</IP>
      </TARGET>
    </TARGETS>
    <CONF>
      <TYPE>vm</TYPE>
    </CONF>
  </SCAN>
  <OPTION_PROFILE>
    <SCAN>
      <PORTS>
        <TCP_PORTS>
          <TCP_PORTS_TYPE>standard</TCP_PORTS_TYPE>
        </TCP_PORTS>
      </PORTS>
    </SCAN>
  </OPTION_PROFILE>
</HEADLESS_SCANNER_JOB>
```
<THREE_WAY_HANDSHAKE>1</THREE_WAY_HANDSHAKE>
</TCP_PORTS>
<UDP_PORTS>
  <UDP_PORTS_TYPE>none</UDP_PORTS_TYPE>
  <HAS_ADDITIONAL>1</HAS_ADDITIONAL>
  <ADDITIONAL_PORTS>53</ADDITIONAL_PORTS>
</UDP_PORTS_ADDITIONAL>
</UDP_PORTS>
<AUTHORITATIVE_OPTION>1</AUTHORITATIVE_OPTION>
</PORTS>
<SCAN_DEAD_HOSTS>1</SCAN_DEAD_HOSTS>

<PERFORMANCE>
  <PARALLEL_SCALING>1</PARALLEL_SCALING>
  <OVERALL_PERFORMANCE>Custom</OVERALL_PERFORMANCE>
  <PARALLEL_TARGETS>1</PARALLEL_TARGETS>
  <PROCESSES_TO_RUN>
    <TOTAL_PROCESSES>40</TOTAL_PROCESSES>
    <HTTP_PROCESSES>20</HTTP_PROCESSES>
  </PROCESSES_TO_RUN>
  <PACKAGE_DELAY>Minimum</PACKAGE_DELAY>
</PERFORMANCE>

<PORT_SCANNING_AND_HOST_DISCOVERY>Minimum</PORT_SCANNING_AND_HOST_DISCOVERY>
</PERFORMANCE>

<LOAD_BALANCER_DETECTION>1</LOAD_BALANCER_DETECTION>

<SYSTEM>
  <HAS_SYSTEM>1</HAS_SYSTEM>
  <SYSTEM_LEVEL>Standard</SYSTEM_LEVEL>
</SYSTEM>

<CUSTOM_LIST>
  <CUSTOM>
    <ID>3001</ID>
    <TITLE><![CDATA[123]]></TITLE>
    <TYPE>FTP</TYPE>
    <LOGIN_PASSWORD><![CDATA[L:temp,P:123123123]]></LOGIN_PASSWORD>
  </CUSTOM>
</CUSTOM_LIST>
Network-wide Scan

When a Headless Scanner performs network scans of the network of targets, the trade-off is between scan duration and resources committed to the scan. The following scan performance controls of the Headless Scanner can be used to tune the scan for available resources.

Number of parallel targets-
HEADLESS_SCANNER_JOB/OPTION_PROFILE/SCAN/PERFORMANCE/PARALLEL_TARGETS

Number of parallel probes per target-
HEADLESS_SCANNER_JOB/OPTION_PROFILE/SCAN/PERFORMANCE/PROCESSES_TO_RUN

**Note:** Assigning at least 4GB of RAM memory to the Headless Scanner docker container makes it possible to perform a scan of 40 targets in parallel with 20 probes per target. The scan specification with configuration:

```xml
<?xml version="1.0"?>
<HEADLESS_SCANNER_JOB>
  <JOB_TYPE>scan</JOB_TYPE>
  <SCAN>
    <TARGETS>
      <TARGET>
        <IP>10.0.0.168</IP>
      </TARGET>
    </TARGETS>
    <CONF>
      <TYPE>vm</TYPE>
    </CONF>
  </SCAN>
  <OPTION_PROFILE>
    <SCAN>
      <PORTS>
        <TCP_PORTS>
          <TCP_PORTS_TYPE>standard</TCP_PORTS_TYPE>
          <THREE_WAY_HANDSHAKE>1</THREE_WAY_HANDSHAKE>
        </TCP_PORTS>
      </PORTS>
    </SCAN>
  </OPTION_PROFILE>
</HEADLESS_SCANNER_JOB>
```
<UDP_PORTS>
  <UDP_PORTS_TYPE>none</UDP_PORTS_TYPE>
  <UDP_PORTS_ADDITIONAL>
    <HAS_ADDITIONAL>1</HAS_ADDITIONAL>
    <ADDITIONAL_PORTS>53</ADDITIONAL_PORTS>
  </UDP_PORTS_ADDITIONAL>
</UDP_PORTS>

<AUTHORITATIVE_OPTION>1</AUTHORITATIVE_OPTION>

<PERFORMANCE>
  <PARALLEL_SCALING>1</PARALLEL_SCALING>
  <OVERALL_PERFORMANCE>Custom</OVERALL_PERFORMANCE>
  <PARALLEL_TARGETS>40</PARALLEL_TARGETS>
  <PROCESSES_TO_RUN>
    <TOTAL_PROCESSES>20</TOTAL_PROCESSES>
    <HTTP_PROCESSES>10</HTTP_PROCESSES>
  </PROCESSES_TO_RUN>
  <PACKET_DELAY>Minimum</PACKET_DELAY>
  <PORT_SCANNING_AND_HOST_DISCOVERY>Minimum</PORT_SCANNING_AND_HOST_DISCOVERY>
</PERFORMANCE>

<AUTHS>
  <LOAD_BALANCER_DETECTION>1</LOAD_BALANCER_DETECTION>
  <ADDL_CERT_DETECTION>1</ADDL_CERT_DETECTION>
  <LITE_OS_SCAN>1</LITE_OS_SCAN>
  <HOST_ALIVE_TESTING>1</HOST_ALIVE_TESTING>
</AUTHS>

Authenticated Scans

If Headless Scanner has access to authentication credentials for targets, the scan results can have a much better quality vulnerability report. To provide authentication credentials AUTHS section of the scan specification can be used.

AUTHS consists of various authentication types, such as SSH, NT, MSSQL, SNMP, etc. Each authentication section consists of a list of authentication records of the authentication type.
Here are few examples of authentication records:

**SSH Authentication:**

```xml
<SSH>
  <RECORD>
    <AUTH_NAME>U2FtcGxlIGF1dGggcmVjb3Jk</AUTH_NAME>
    <USERNAME>c2NhbnVzZXI=</USERNAME> <!-- scanuser -->
    <PASSWORD>MWdCc2prSWs3</PASSWORD> <!-- 1gBsjkIk7 -->
    <CLEARTEXT>true</CLEARTEXT>
    <IP_RANGE>10.0.0.0-10.0.0.255</IP_RANGE>
    <HOSTID>1</HOSTID>
    <AUTH_ID>OQ==</AUTH_ID>
  </RECORD>
  <RECORD>
    <AUTH_ID>MTQzOTgx</AUTH_ID>
    <AUTH_NAME>VW5peCBBdXRoIHdpdGggU1VETyBvcHRpb24=</AUTH_NAME>
    <CLEARTEXT>false</CLEARTEXT>
    <IP_RANGE>10.0.0.1-10.255.255.255</IP_RANGE>
    <RSA_PK>...BASE64 ENCODED PRIVATE RSA KEY...</RSA_PK>
    <SUDO>true</SUDO>
    <USERNAME>...</USERNAME> <!-- base64 encoded non-root username -->
  </RECORD>
</SSH>
```

**SSH Version 2 Authentication record example:**

```xml
<SSH2>
  <RECORD>
    <AUTH_ID>MTQzOTkx</AUTH_ID>
    <AUTH_NAME>VW5peCBBdXRoIFNTSDI=</AUTH_NAME>
    <USERPASS>
      <PW_INFO>
        <PASSWORD>...</PASSWORD>  <!-- base64 encoded password -->
      </PW_INFO>
      <CLEARTEXT>false</CLEARTEXT>
    </USERPASS>
    <IP_RANGE>10.10.10.1-10.10.10.255</IP_RANGE>
    <USERNAME>...</USERNAME> <!-- base64 encoded non-root username -->
  </RECORD>
  <RECORD>
    <AUTH_ID>MTQ1MjAz</AUTH_ID>
  </RECORD>
</SSH2>
```
<AUTH_NAME>VW5peCBBdXRoIHdpdGggU1VETyBhbmQgQ3VzdG9tIGNvbW1hbmQ=</AUTH_NAME>

<USERNAME>...</USERNAME> <!-- non root username -->
<IP_RANGE>192.168.1.10</IP_RANGE>
</USERPASS>

<PW_INFO>
  <PASSWORD>...</PASSWORD> <!-- base64 encoded password -->
</PW_INFO>
</USERPASS>
</SUDO_INFOS>
</RECORD>
</SSH2>

<NT>

<RECORD>
  <AUTH_NAME>U2FtcGxlIE5UIGF1dGggcmVjb3JkICMx</AUTH_NAME>
  <USERNAME>YWRtaW4=</USERNAME> <!-- admin -->
  <PASSWORD>MWdCc2prSWs3</PASSWORD>  <!-- 1gBsjkIk7 -->
  <DOMAIN>Zm9vYmFy</DOMAIN> <!-- foobar -->
  <HOSTID>FALSE</HOSTID>
  <AUTH_ID>MA==</AUTH_ID>
</RECORD>

<RECORD>
  <AUTH_NAME>U2FtcGxlIE5UIGF1dGggcmVjb3JkICMy</AUTH_NAME>
  <USERNAME>c2NhbnVzZXI=</USERNAME>  <!-- scanuser -->
  <PASSWORD>MWdCc2prSWs3</PASSWORD>  <!-- 1gBsjkIk7 -->
  <DOMAIN>YWNtZS5jb20=</DOMAIN>  <!-- acme.com -->
  <HOSTID>FALSE</HOSTID>
  <AUTH_ID>MQ==</AUTH_ID>
</RECORD>
</NT>
The following is the complete example with AUTHS section:

```xml
<?xml version="1.0"?>
<HEADLESS_SCANNER_JOB>
  <JOB_TYPE>scan</JOB_TYPE>
  <SCAN>
    <TARGETS>
      <TARGET>
        <IP>10.0.0.168</IP>
      </TARGET>
    </TARGETS>
    <CONF>
      <TYPE>vm</TYPE>
    </CONF>
  </SCAN>
  <OPTION_PROFILE>
    <SCAN>
      <PORTS>
        <TCP_PORTS>
          <TCP_PORTS_TYPE>custom</TCP_PORTS_TYPE>
          <TCP_PORTS_ADDITIONAL>
            <HAS_ADDITIONAL>1</HAS_ADDITIONAL>
            <ADDITIONAL_PORTS>22</ADDITIONAL_PORTS>
          </TCP_PORTS_ADDITIONAL>
          <THREE_WAY_HANDSHAKE>1</THREE_WAY_HANDSHAKE>
        </TCP_PORTS>
        <UDP_PORTS>
          <UDP_PORTS_TYPE>none</UDP_PORTS_TYPE>
          <UDP_PORTS_ADDITIONAL>
            <HAS_ADDITIONAL>1</HAS_ADDITIONAL>
            <ADDITIONAL_PORTS>53</ADDITIONAL_PORTS>
          </UDP_PORTS_ADDITIONAL>
        </UDP_PORTS>
        <AUTHORITATIVE_OPTION>1</AUTHORITATIVE_OPTION>
      </PORTS>
      <SCAN_DEAD_HOSTS>1</SCAN_DEAD_HOSTS>
    </SCAN>
    <PERFORMANCE>
      <PARALLEL_SCALING>1</PARALLEL_SCALING>
      <OVERALL_PERFORMANCE>Custom</OVERALL_PERFORMANCE>
      <PARALLEL_TARGETS>40</PARALLEL_TARGETS>
      <PROCESSES_TO_RUN>
```
Customizing Headless Scanner Input and Output

The Headless Scanner input file name, location and scanner output directory can be customized using environment variables. There are two environment variables you can use for customizing the Headless Scanner.

• SCAN_INPUT_FILE: If it is present, it’s expected to point to the scan specification file.

• SCAN_OUTPUT_DIR: If it is present, it tells the Headless Scanner where to generate the output/result files.

Here’s an example:
docker run \
  -it \

Configurations

Launching Headless Scanner

In the above example the input file headless_vm_spec.xml can be read from current directory and the output will be generated in /tmp/custom directory.

Launching Headless Scanner

The entry point of the Headless Scanner docker image is a shell script that performs preparatory steps for launching scans. The script expects to find the Scan Specification file as /usr/share/qualys/scan_spec.xml.

Here is one way that by having a scan_spec.xml file in the local directory, the Headless Scanner can be launched:

docker run -it
   --rm \  
   --ulimit core=-1 \  
   --security-opt seccomp=unconfined \  
   --name demo_scanner \  
   -v $(pwd):/usr/share/qualys/ \  
   hengine:0.2.0-22

In this case, the Headless Scanner container can find the scan_spec.xml file and process it. Upon exit, the scan job results can be created in the same directory under the scan job ID subdirectory.
Here is the layout of the /usr/share/qualys directory when the Headless Scanner completes the scan job:

```
usr
    share
        qualys
            scan_spec.xml
            2020-08-18-11-43-02zutc-1
            10.10.10.248.xml
            10.10.10.249.xml
            10.10.10.250.xml
            10.10.10.252.xml
            scan_status.xml
```

### Running Headless Scanner with Separate Vulnerability Signatures

The Headless Scanner image contains two main components: the engine and the vulnerability signatures.

The changes to the engine are happening at a much slower rate as compared to signature changes. Therefore it is beneficial to separate the signature releases from the Headless Scanner engine to allow for a much higher frequency of updates.

The signature is released as a separate docker image. However, the image is just a container for the signatures. The image itself cannot be used to run any scans.

To use vulnerability signatures from a hsengine-vulnsigs image, the following steps can be followed:

1. Obtain the signatures.
   ```bash
docker pull qualys/hsengine-vulnsigs:2.5.560-3
   ```

2. Create a non-running container to obtain the signatures.
   ```bash
docker create --name sigs qualys/hsengine-vulnsigs:2.5.560-3
   ```

3. Copy the signatures from the container.
   ```bash
docker cp sigs:/usr/local/qualys /tmp
   ```

4. Provide the signatures to the Headless Scanner container.
   ```bash
docker run -it --rm \
    --name headless_plus_updates \ 
    -v /tmp/headless_scan/config:/usr/share/qualys \ 
    -v /tmp/qualys/VULNSIGS-2.5.560-3:/usr/local/qualys/VULNSIGS-2.5.560-3 \ 
```
Configurations
Launching Headless Scanner

```
-v
/tmp/qualys/VULNSIGS_VERSION:/usr/local/qualys/VULNSIGS_VERSION \
qualys/hsengine:0.6.300b2-1
```

5. Remove the container for signatures.

```
docker rm -f sigs
```

6. This can launch the Headless Scanner container and while scanning, the 2.5.560-3 version of the signatures can be used.

Running Headless Scanner with a Read-Only File-system

In the few cases where running Headless Scanner in a container with a Read-Only file system is desirable, the command to launch the container can be modified to use the --read-only flag. The --read-only flag effectively removes the container's write layer and mounts the container's root file-system as a read-only, prohibiting writes to locations other than the specified volumes for the container.

However, Headless Scanner needs some portion of the file-system to be writable for its operational needs, such as caches, temporary artifacts, etc. These portions of the file-system can be explicitly mounted with Read-Write access.

The final command may look like the following:

```
docker run --read-only \
- it \
-- rm \
-- ulimit core=-1 \
-- security-opt seccomp=unconfined \
-- name vm_scanner \
-- net=host \
-v $(pwd)/:/usr/share/qualys \
-v /usr/local/qualys -v /tmp -v /run \
art-hq.intranet.qualys.com:5001/qualys/headless/hsengine:0.6.568b2-1
```

Running Headless Scanner with Non-root User

Due to security considerations, running the Headless Scanner container with non-root user privileges is desirable. At the same time, the Headless Scanner process needs to have elevated privileges to perform operations like binding to ports below 1024, opening and using raw ip sockets, etc.
User namespace mapping allows the Docker daemon to create an isolated namespace that looks and feels like a root namespace. However, the root user inside of this namespace is mapped to a non-privileged uid on the Docker Host. This means containers can effectively have root privilege inside the user namespace but no privileges on the Docker Host.

**Enable the User Namespace**

The subordinate UID and GID ranges must be associated with a user. For this to work, you can create any user of your choice or use the existing username.

1. Execute the below command to create a ’dockremap’ (default) or any other user, for example, the dockertest user on the host.

   useradd -M -s /bin/false dockremap
   useradd -M -s /bin/false dockertest

2. To verify if the user is created, run the following command.

   id dockremap
   uid=552(dockremap) gid=552(dockremap) groups=552(dockremap)

   id dockertest
   uid=1005(dockertest) gid=1005(dockertest) groups=1005(dockertest)

3. The namespace remapping is handled on the host using two files, /etc/subuid and /etc/subgid. You need to add the mappings of the user into these files.

   echo "dockremap:500000:65536" >> /etc/subuid
   echo "dockremap:500000:65536" >> /etc/subgid

   or

   echo "dockertest:231072:65536" >> /etc/subuid
   echo "dockertest:231072:65536" >> /etc/subgid

The UID 0 in the containers can map to UID 500000 on the host, and subsequent UIDs can map to the next 64k of UIDs on the host.

**Note:** Existing UIDs on the host must not fall into the range of UIDs chosen (check by calculating 500000+ 65536 - 1)
Enable usersns-remap on the daemon

The user namespace remapping or usersns-remap makes use of the Linux user namespace to re-map the root user within the container to a less-privileged user in the host machine. In this way, the container can run as a root, which is mapped to a user with no privileges on the host.

Enabling usersns-remap effectively masks existing image and container layers and other Docker objects within /var/lib/docker/. This is because Docker needs to adjust the ownership of these resources and store them in a subdirectory within /var/lib/docker/. Also, if you disable usersns-remap, you can't access any of the resources created while it was enabled.

- The docker installation must be cleaned by removing any previous image. The output of "docker image ls" has to be empty.

- You can configure the daemon using the daemon.json configuration file. Then edit the daemon.json file located under "/etc/docker/" (If the file does not exist, create it).

Add to the daemon.json the following entry.

```
cat > /etc/docker/daemon.json <<-EOT
{
   "usersns-remap": "default",     # or "usersns-remap": "dockertest"
   "group": "dockerroot"
}
EOT
```

By using the "default" value, docker can use the "dockremap" user and group to make the remapping.

'group' allows any member of group 'dockerroot' to control docker.

**Note:** For anyone added to this group is effectively being trusted with root privileges.

The User namespaces are not enabled on the kernel by default in Centos/Redhat 7. Execute the following steps to enable them.

```
grubby --args="namespace.unpriv_enable=1 user_namespace.enable=1" --update-kernel="$(grubby --default-kernel)"
echo "user.max_user_namespaces=15076" >> /etc/sysctl.conf reboot
sudo systemctl start docker
```

1. Check if the user namespace is enabled.
Configurations

Launching Headless Scanner

Enabled:

sh-4.2# docker info | grep Root
Docker Root Dir: /var/lib/docker/500000.500000   \[ Docker root directory is pointing to user 500000 \]

Disabled:

sh-4.2# docker info | grep Root
Root Rotation In Progress: false
Docker Root Dir: /home/docker

2. Verify that a namespaced directory exists within /var/lib/docker/ named with the UID and GID of the namespaced user, owned by that UID and GID, and not group-or-world-readable. Some of the subdirectories are still owned by root and have different permissions.

sudo ls -ld /var/lib/docker/500000.500000/
```
```
drwx------ 11 500000 500000 11 Jun 21 21:19 /var/lib/docker/500000.500000/
```

Limitation of Using User Namespace:

Refer to all the limitations mentioned on the Docker portal.

Prior to executing the docker command

- Permission issue: If there are any locations on the Docker host where the unprivileged user needs to write, you need to adjust the permissions of those locations accordingly.

Note: You can modify the permissions of the directories/files only after configuring and restarting the Docker.

After enabling the user namespace, so either try changing the permission of the current working directory to 757 or set the user and group to 500000.

`chmod 757 /home/bkhare/hsengine_newbuildplan/scanengine/tests/scan_specs/VM_Remote_Reg_XP`

- There is a limitation that the host’s network namespace cannot be shared when user namespaces are enabled, so you need to remove (--net=host) from the docker command if specified.

Docker run Command in the User Namespace Environment

After enabling the user namespace, you can run the docker command to launch the headless scanner.
Here is an example to run the command:

docker run -it \
  --rm \
  --ulimit core=-1 \
  --security-opt seccomp=unconfined \
  --name scanner \
  -v $(pwd):/usr/share/qualys/ \
  art-hq.intranet.qualys.com:5001/qualys/headless/hsengine:0.6.561b2-1

Docker run command with the root user in the user namespace environment:

To disable user namespaces for a specific container, add the --userns=host flag to the docker commands.

Here is an example to run the command:

docker run -it \
  --rm \
  --ulimit core=-1 \
  --security-opt seccomp=unconfined \
  --name scanner \
  --userns=host \
  -v $(pwd):/usr/share/qualys/ \
  art-hq.intranet.qualys.com:5001/qualys/headless/hsengine:0.6.561b2-1

Disabling the User Namespace

Delete all the entries from the /etc/docker/daemon.json file and reboot the system. Use the following command to reboot the system:

reboot

Headless Scanner Output

Scan Job Status File

Scan job status file is an XML file which describes the status of the scan job.

Here is a sample scan job status file:

<SCAN SCAN_ID="2020-08-13-05-43-38zutc-16" status="success">
  <MESSAGE>
    <INFO name="vulnsigs_version">VULNSIGS-2.4.954-2</INFO>
  </MESSAGE>
</SCAN>
The SCAN_ID attribute of the root element SCAN is the scan reference assigned to the scan job.

The status attribute of the root element conveys the overall scan status. Possible values for the status attribute are:
- running
- success
- error

The overall scan status is considered "success" if all the requirements of the scan engine have been met when the container was created. Even if individual hosts encountered errors during the scan, the over-scan status would still be reported as "success." Cases where the overall scan status cannot be reported as "success" are when the expected Scan Specification file is missing or doesn't validate against the Scan Specification schema.

The scan job status file has two sections:
- Scan meta-information in the MESSAGE tag
- Scan results in the RESULTS tag

**Message Section**

The MESSAGE tag in the Scan Job-status file contains scan job meta-information such as scan duration, number of alive hosts, vulnerability signatures version used during the scan, etc.
Results Section

The RESULTS tag contains a list of TARGET tags, each of which describes the scan result details of the IPv4 target. The "id" attribute of the TARGET element contains the IPv4 address of the target. The "status" attribute indicates the scan status of the target. The possible value of status attributes are:

- started
- up
- alive
- down
- done
- error
- canceled

Error Reporting

In cases where the scan encounters error conditions during the scan, the errors can be reported in the scan status file. The scan errors can be split into two categories: overall scan errors and individual host scan errors.
Overall scan errors include conditions like missing or malformed scan specification files etc. These errors can be reported with the status attribute of SCAN element in the scan_status.xml file.

Here is an example of a malformed scan specification file:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SCAN id="2022-08-18-16-10-51zutc-1" status="failed">
  <MESSAGE><![CDATA[invalid scan specification file. Exiting!]]></MESSAGE>
  <DETAILED_MESSAGE><![CDATA[/usr/share/qualys/scan_spec.xml:1: parser error : Document is empty
  ^
  /usr/share/qualys/scan_spec.xml:1: parser error : Start tag expected, '<' not found
  ]]></DETAILED_MESSAGE>
  <RESULTS>
  </RESULTS>
</SCAN>
```

The MESSAGE and DETAILED_MESSAGE elements contain details about the error condition.

In case of individual host errors, the overall scan status can be set to success, while the host result's status attribute can be set to error for all hosts that have encountered an error during the scan.

Here is an example:

```xml
<SCAN SCAN_ID="2020-08-10-19-35-04zutc-14" status="success">
  <MESSAGE>...</MESSAGE>
  <RESULTS>
    <TARGET id="10.20.31.38" status="error">
      <ERROR code="2000">
        <![CDATA[Error: Scan did not complete]]>
      </ERROR>
    </TARGET>
  </RESULTS>
</SCAN>
```
Scan Results File

When the target scan completes, the results of individual targets can be stored in the scan job directory under `<IPv4>.xml` name, where the IPv4 is the IPv4 address of the target. Appendix B has the XSD schema of the results XML file.
Troubleshooting Scenarios

The following are the troubleshooting scenarios with possible resolutions:

• **Enable troubleshooting logs and tcpdump files generation**

To enable troubleshooting logs and tcpdump file generation, add the following DEBUG block in the REPORT element. The logs and tcpdump files are generated for each target and stored in the output directory.

```
<REPORT>

  ...

  <DEBUG>
    <LOG>true</LOG>
    <TRAFFIC_DUMP>true</TRAFFIC_DUMP>
  </DEBUG>

</REPORT>
```

To enable the core file dumps when a crash is reported, add the HARVEST_CORE block in DEBUG block in the REPORT element:

```
<REPORT>

  ...

  <DEBUG>
    <LOG>true</LOG>
    <HARVEST_CORE>true</HARVEST_CORE>
    <TRAFFIC_DUMP>true</TRAFFIC_DUMP>
  </DEBUG>

</REPORT>
```
# Troubleshooting Scans

<table>
<thead>
<tr>
<th>Stage</th>
<th>Action</th>
<th>Feedback</th>
<th>Expected Outcome</th>
<th>Possible Failures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launching a scan</td>
<td>Deploy a ECS task.</td>
<td>Check the Task status using AWS ECS describe-tasks</td>
<td>&quot;lastStatus&quot;: &quot;RUNNING&quot;, &quot;connectivity&quot;: &quot;CONNECTED&quot;, &quot;healthStatus&quot;: &quot;UNKNOWN&quot;, ...</td>
<td>inaccessible image, missing or invalid scan specification file, improper access to S3 resources</td>
</tr>
<tr>
<td>Active Scan</td>
<td>Inspect S3 output bucket</td>
<td>The task status should be &quot;RUNNING&quot; state.</td>
<td>SCAN/@status=&quot;running&quot;</td>
<td></td>
</tr>
<tr>
<td>Completed Scan</td>
<td>Check scan_status.xml</td>
<td>SCAN/@status=&quot;success&quot;. Results should be available in individual files.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Failed Scan</td>
<td>Check scan_status.xml</td>
<td>SCAN/@status=&quot;failed&quot;</td>
<td>malformed scan specification file, scan specification file with incorrect values</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Evidence</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>The scan task gets to</td>
<td>Check permissions of the S3 bucket to ensure that</td>
<td>the &quot;RUNNING&quot; state, but the scan results in S3 bucket is empty</td>
</tr>
<tr>
<td>the &quot;RUNNING&quot; state,</td>
<td>the role that the ECS task is running has permis-</td>
<td></td>
</tr>
<tr>
<td>but the scan results</td>
<td>sion to write to the S3.</td>
<td></td>
</tr>
</tbody>
</table>
## Troubleshooting Scenarios

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Evidence</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>The scan task fails to reach the &quot;RUNNING&quot; state</td>
<td>The task status as returned by AWS CLI has .lastStatus not equal to RUNNING</td>
<td>Check the AWS Fargate task messages. This could be a matter of missing image or failed access to one or more resources.</td>
</tr>
<tr>
<td>The scan completes, but scan_status.xml file SCAN/@status contains &quot;failed&quot;</td>
<td></td>
<td>Check the scan_status.xml file's MESSAGE and DETAILED_MESSAGE elements for details about the failure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>There could be the number of causes for the failure:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Missing or mistyped scan specification file:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The scan_status.xml file will indicate that the input scan specification file is missing:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;SCAN id=&quot;2023-04-07-14-38-12zutc-1&quot; status=&quot;failed&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;MESSAGE&gt;Missing scan config file. Exiting!&lt;/MESSAGE&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/RESULTS&gt; &lt;/RESULTS&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;/SCAN&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Invalid scan specification file- This could be malformed XML or incorrect values for one or more scan specifications. The DETAILED_MESSAGE element will point to the violating section of the XML.</td>
</tr>
</tbody>
</table>

---

63
Headless Scanner KBXML

The Headless Scanner image incorporates the Qualys knowledge base in the form of an XML file called KBXML, which provides detailed information about vulnerabilities. The Qualys Knowledge base categorizes these vulnerabilities into confirmed, potential, and information-gathered data. Each vulnerability has a unique identifier referred to as QID.

The Qualys Knowledge base is continuously updated to support new vulnerabilities and maintain the existing ones, ensuring you have access to the latest information to take necessary measures to address any vulnerabilities in your systems.

Extract KBXML from Headless Scanner Image

You must follow the following steps to fetch the kbxml file from the image.

1. Obtain the headless scanner image.
   
   ```bash
docker pull qualys/hsengine:latest
   ```

2. Create a non-running container.
   
   ```bash
docker create --name scanner qualys/hsengine:latest
   ```

3. Copy the kbxml file from the container.
   
   ```bash
docker cp scanner:/usr/local/qualys/ml/xml/kbxml.xml /tmp
   ```

4. Remove the container.
   
   ```bash
docker rm -f scanner
   ```

The KBXML file can be referred at location /tmp/kbxml.xml

KBXML File Attributes

- **QID** - It corresponds to Qualys identifier to identify the unique vulnerability.
- **CHANGELOG** - It corresponds to tracking the code changes in the existing vulnerability QID.

The new log entry can be seen under CHANGELOG/LOG tag whenever an attribute is modified.

changeDate tag is the date when the attributes are modified with the modification details.
Change Log:

<VULN lastModified="2022-03-11T10:20:07+00:00">
<QID>178844</QID>
<CHANGE_LOG>
  <LOG changeDate="2021-10-28T10:05:54+00:00"><![CDATA[Real-time threat indicator "Privilege_Escalation" added.]]></LOG>
  <LOG changeDate="2021-10-27T18:03:34+00:00"><![CDATA[Real-time threat indicator "High_Data_Loss" added.]]></LOG>
  <LOG changeDate="2021-10-27T18:04:07+00:00"><![CDATA[Real-time threat indicator "Denial_of_Service" added.]]></LOG>
  <LOG changeDate="2021-10-27T18:01:38+00:00"><![CDATA[Real-time threat indicator "High_Lateral_Movement" added.]]></LOG>
  <LOG changeDate="2021-10-27T18:02:59+00:00"><![CDATA[Real-time threat indicator "Easy_Exploit" added.]]></LOG>
</CHANGE_LOG>

• PRODUCT: It provides information about the product associated with the QID, like VM, SCA, WAS, etc

• CATEGORY: It corresponds to VULNS which means vulnerability.

• SEVERITY: It provides severity levels or ranking for vulnerabilities.

<table>
<thead>
<tr>
<th>Severity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity 5</td>
<td>It is most urgent as it presents greater risks and needs to be addressed immediately.</td>
</tr>
<tr>
<td>Severity 4,3</td>
<td>The medium Risk involves the potential compromise of access to host information, use of services, etc.</td>
</tr>
<tr>
<td>Severity 2,1</td>
<td>This is the least urgent as it involves the disclosure of sensitive information.</td>
</tr>
</tbody>
</table>

• CVE: It contains information about the Common Vulnerability and Exposures (CVE) IDs associated with vulnerability.

• PROPERTIES: It depicts the QID properties. Here is the information about the properties field.
CVSS scores are generated based on the CVSS specification.

- **BASE_METRIC** - It refers to static attributes inherited in vulnerability, computed by the vendor, and once set, is not expected to change.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESSVECTOR (Local/Adjacent/Network)</td>
<td>This metric measures how remote an attacker can attack a target to exploit the target’s vulnerability, i.e., physical access, local network, or remote networks.</td>
</tr>
<tr>
<td>ACCESSCOMPLEXITY (High/Medium/Low)</td>
<td>This metric specifies the attacks complexity from high to low specialized access conditions required to exploit the vulnerability.</td>
</tr>
<tr>
<td>AUTHENTICATION (Multiple/Single/None)</td>
<td>This metric specifies the number of times an attacker must authenticate to the target system to exploit the vulnerability.</td>
</tr>
<tr>
<td>CONFIDENTIALITYIMPACT (None/Partial/Complete)</td>
<td>This metric specifies the impact on confidentiality of a successful exploit of the vulnerability on the target system.</td>
</tr>
<tr>
<td>INTEGRITYIMPACT (None/Partial/Complete)</td>
<td>This metric specifies the impact on the integrity of a successful exploit of the vulnerability on the target system.</td>
</tr>
<tr>
<td>AVAILABILITYIMPACT (None/Partial/Complete)</td>
<td>This metric specifies the impact on the availability of a successful exploit of the vulnerability on the target system.</td>
</tr>
<tr>
<td>BASESCORE</td>
<td>The Static CVS score of the CVE.</td>
</tr>
</tbody>
</table>
• **TEMPORAL_METRIC**- The value of temporal metrics changes over the lifetime of the vulnerability as exploits are developed, disclosed, and automated and as mitigations and fixes are made available.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXPLOITABILITY</td>
<td>This metric measures the vulnerability being attacked and is typically based on the current state of exploit techniques, exploit code availability, or active, &quot;in-the-wild&quot; exploitation.</td>
</tr>
<tr>
<td>REMEDIATIONLEVEL</td>
<td>The less official and permanent a fix, the higher the vulnerability score.</td>
</tr>
<tr>
<td>REPORTCONFIDENCE</td>
<td>The degree of confidence in the vulnerability and the credibility of the known technical details.</td>
</tr>
<tr>
<td>TEMPORALSCORE</td>
<td>This equation combines the temporal metrics with the base score to produce a temporal score ranging from 0 to 10.</td>
</tr>
</tbody>
</table>

• **ENVIRONMENTAL_METRIC**- The environmental metrics use the base and current temporal score to assess the severity of a vulnerability in the context of how the vulnerable product or software is deployed. This measure is calculated subjectively, typically by affected parties.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLLATERALDAMAGEPOTENTIAL</td>
<td>The collateral damage potential (CDP) metric measures the potential loss or impact on physical assets such as equipment (and lives) or the financial impact upon the affected organization if the vulnerability is exploited.</td>
</tr>
<tr>
<td>TARGETDISTRIBUTION</td>
<td>The target distribution (TD) metric measures the proportion of vulnerable systems in the environment.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CONFIDENTIALITY REQUIREMENT</td>
<td>Three different metrics assess the specific security requirements for confidentiality (CR), integrity (IR), and availability (AR), allowing the environmental score to be fine-tuned according to the users' environment.</td>
</tr>
<tr>
<td>INTEGRITY REQUIREMENT</td>
<td></td>
</tr>
<tr>
<td>AVAILABILITY REQUIREMENT</td>
<td></td>
</tr>
<tr>
<td>ENVSCORE</td>
<td>This equation combines the environmental metrics with the temporal score to produce an ecological score ranging from 0 to 10.</td>
</tr>
</tbody>
</table>

- THREAT_INTEL- It corresponds to Vulnerability associated threads
- QID_RELEASE_DATE- It corresponds to the time when vulnerability QID was released.
- QID_MODIFIED_DATE- It corresponds to the time when vulnerability QID was modified.
- TITLE- It corresponds to the title of the vulnerability QID.
- DESCRIPTIONS- It corresponds to the description of the vulnerability QID.
- SOLUTIONS- It corresponds to providing the lists of solutions available from vendors to fix the vulnerability.
- CONSEQUENCES- These correspond to the impact due to the exploited vulnerability.
- VULN_PATCH- It gives the vendor-supplied patches available to fix the vulnerability.
  - PATCHAVAILABLE- Yes/No
  - PATCH_RELEASE_DATE- It corresponds to when the patch was released if it is available.
  - VENDOR_SEVERITY- It corresponds to the severity of the patch.
  - PATCHES- It corresponds to the list of patches.
- VENDOR_REFERENCES- This is a list of the various software vendor websites.
KBXML Sample

<VULN lastModified="2023-02-28T16:00:12+00:00">
  <QID>991431</QID>
  <CHANGE_LOG>
    <LOG changeDate="2023-01-06T10:04:36+00:00"><![CDATA[Real-time threat indicator "Privilege_Escalation" added.]]></LOG>
    <LOG changeDate="2023-02-28T15:43:18+00:00"><![CDATA[CVSS V2 temporal score updated from "2.6" to "3.0"]]></LOG>
    <LOG changeDate="2023-02-28T15:43:18+00:00"><![CDATA[CVSS V3 temporal score updated from "4.7" to "5.0"]]></LOG>
  </CHANGE_LOG>
  <PRODUCTS>
    <PRODUCT id="16" name="SCA" lastModified="2023-01-04T00:00:00+00:00"/>
  </PRODUCTS>
  <CATEGORY>Vuln</CATEGORY>
  <SEVERITY>4</SEVERITY>
  <DOMAINVULN>SCA</DOMAINVULN>
  <CVE>
    <CVE_ID>CVE-2022-25203</CVE_ID>
  </CVE>
  <SF_ID>0</SF_ID>
  <AUTHENTICATION>
    <AUTH name="unix" flag="u" value="true"/>
    <AUTH name="windows" flag="n" value="false"/>
    <AUTH name="oracle" flag="o" value="false"/>
    <AUTH name="snmp" flag="m" value="false"/>
    <AUTH name="webscan" flag="w" value="false"/>
    <AUTH name="db2" flag="d" value="false"/>
    <AUTH name="vmware" flag="e" value="false"/>
    <AUTH name="mssql" flag="q" value="false"/>
    <AUTH name="httpbasic" flag="b" value="false"/>
    <AUTH name="noauth" flag="r" value="false"/>
    <AUTH name="formauth" flag="h" value="false"/>
    <AUTH name="sybase" flag="k" value="false"/>
  </AUTHENTICATION>
  <PROPERTIES>
    <PROP name="upgradable" flag="v" value="false"/>
    <PROP name="inferred" flag="i" value="false"/>
    <PROP name="pci" flag="p" value="false"/>
    <PROP name="destructive" flag="z" value="false"/>
  </PROPERTIES>
</VULN>
<PROP name="cisconac" flag="c" value="false"/>
<PROP name="nacagent" flag="a" value="false"/>
<PROP name="audit" flag="t" value="false"/>
<PROP name="listeningport" flag="l" value="false"/>
<PROP name="eol" flag="f" value="false"/>
<PROP name="qidfilter" flag="x" value="false"/>
</PROPERTIES>
<QID_PROPERTIES>
<PROPERTIES_VALUE id="512795" value="unx" lastModified="2023-01-04T09:40:17+00:00"/>
</QID_PROPERTIES>
<PRIVATE_SOURCES>
<PRIVATE_SOURCE source="SCA" published="0" lastModified="2023-01-04T09:40:04+00:00"/>
</PRIVATE_SOURCES>
<CVSS lastModified="2023-02-28T16:00:01+00:00">
<Base_Metric>
<ACCESSVECTOR text="Network" value="3"/>
<ACCESSCOMPLEXITY text="Medium" value="2"/>
<AUTHENTICATION text="Single" value="2"/>
<CONFIDENTIALITYIMPACT text="None" value="1"/>
<INTEGRITYIMPACT text="Partial" value="2"/>
<AVAILABILITYIMPACT text="None" value="1"/>
<BASESCORE>3.5</BASESCORE>
</BASE_Metric>
<TEMPORAL_Metric>
<EXPLOITABILITY text="Unproven" value="1"/>
<REMEDIATIONLEVEL text="Unavailable" value="4"/>
<REPORTCONFIDENCE text="Confirmed" value="3"/>
<TEMPORALSCORE>3</TEMPORALSCORE>
</TEMPORAL_Metric>
</CVSS>
</NIST_CVSS lastModified="2023-01-04T09:40:02+00:00"/>
<ACCESSVECTOR text="Network" value="3"/>
<ACCESSCOMPLEXITY text="Medium" value="2"/>
<AUTHENTICATION text="Single" value="2"/>
<CONFIDENTIALITYIMPACT text="None" value="1"/>
<INTEGRITYIMPACT text="Partial" value="2"/>
<AVAILABILITYIMPACT text="None" value="1"/>
<BASESCORE>3.5</BASESCORE>
</NIST_CVSS>

<CVSS_V3 lastModified="2023-02-28T16:00:09+00:00">
  <BASE_METRIC_V3>
    <ATTACKVECTOR text="Network" value="1"/>
    <ATTACKCOMPLEXITY text="Low" value="1"/>
    <PRIVILEGESREQUIRED text="Low" value="2"/>
    <USERINTERACTION text="Required" value="2"/>
    <SCOPE text="Changed" value="2"/>
    <CONFIDENTIALITYIMPACT text="Low" value="2"/>
    <INTEGRITYIMPACT text="Low" value="2"/>
    <AVAILABILITYIMPACT text="None" value="1"/>
    <BASESCORE>5.4</BASESCORE>
  </BASE_METRIC_V3>
  <TEMPORAL_METRIC_V3>
    <EXPLOITCODEMATUREITY text="Unproven" value="1"/>
    <REMEDIATIONLEVEL text="Unavailable" value="4"/>
    <REPORTCONFIDENCE text="Confirmed" value="3"/>
    <TEMPORALSOCORE>5</TEMPORALSOCORE>
  </TEMPORAL_METRIC_V3>
</CVSS_V3>

<NIST_CVSS_V3 lastModified="2023-01-04T09:40:16+00:00">
  <ATTACKVECTOR text="Network" value="1"/>
  <ATTACKCOMPLEXITY text="Low" value="1"/>
  <PRIVILEGESREQUIRED text="Low" value="2"/>
  <USERINTERACTION text="Required" value="2"/>
  <SCOPE text="Changed" value="2"/>
  <CONFIDENTIALITYIMPACT text="Low" value="2"/>
  <INTEGRITYIMPACT text="Low" value="2"/>
  <AVAILABILITYIMPACT text="None" value="1"/>
  <BASESCORE>5.4</BASESCORE>
</NIST_CVSS_V3>

<THREAT_INTEL>
  <THREAT threatIntelID="13"
    ID="246765"
Java (Maven) Security Update for com.sonymobile.jenkins.plugins.teamviews:team-views (GHSA-mv5c-724f-3fq7)

Jenkins Team Views Plugin 0.9.0 and earlier does not escape team names, resulting in a stored cross-site scripting (XSS) vulnerability exploitable by attackers with Overall/Read permission.

Successful exploitation of this vulnerability could lead to a security breach or could affect integrity, availability, and confidentiality.

No patch available.

Vendor Reference:
- GHSA-mv5c-724f-3fq7
  - URL: https://github.com/advisories/GHSA-mv5c-724f-3fq7
Headless Scanner KBXML

</VENDOR_REFERENCES>
</VULN>
Headless Scanner Support for AWS ECS

Headless Scanner AWS Fargate Deployment

Qualys Headless Scanner is a standalone network vulnerability scanner packaged in a Docker container. Typically, you have deployed Qualys Headless Scanner on a virtual machine or a cloud VM instance. But all modern cloud providers have native container service, a more natural method for Qualys Headless Scanner deployment.

Qualys Headless Scanner operation hinges on the fundamental principle that the scanner lifetime spans the duration of a single scan. This means that the scanner instance can be used for only one scan. The Headless Scanner container cannot be re-used to launch multiple scans in parallel or sequentially. This principle dictates that the entire scope of the scan needs to be defined and provided to the Headless Scanner. Also, when the scan is completed, the Headless Scanner exits and produces scan results.

Headless Scanner ECS Deployment

The following steps are required to run the Qualys Headless Scanner in the AWS ECS environment.

1. Define the Access/Role for the Headless Scanner.
2. Setup the Input & Output of the Headless Scanner.
3. Headless Scanner image needs to be made available.
4. Define the ECS cluster/task.

The following steps need to perform to set up all the necessary components.

1. Navigate to the IAM service in the AWS console. Select Roles from the left-hand side menu and click the Create role.
Headless Scanner Support for AWS ECS

2. Select **AWS service** for the type. The S3 and Elastic Container Service for the Use case and click **Next**.

3. In Add Permissions step, add the following policies.
   - AmazonS3FullAccess
   - AmazonECSTaskExecutionRolePolicy
4. In the Role details page, provide ecsTaskExecution for the role name and click the Create role to create the role.

5. Navigate to the S3 service in the AWS console and create an S3 bucket that can be used for storing scan specification files.

6. Create another S3 bucket that can be used for storing scan results.

The S3 bucket for storing the results can be reused for multiple scans, or a new one for each scan can be created. This depends on the strategy for launching scans.
7. The Headless Scanner image that can be used for this task is published in Qualys’ ECR as a private repository.

The image can be shared with you upon request. You need to provide your AWS account number to Qualys so that the Headless Scanner images can be shared with that AWS account.

You have the following access to the ECR repository:
- `ecr:BatchCheckLayerAvailability`
- `ecr:BatchGetImage`
- `ecr:GetDownloadUrlForLayer`

8. Navigate to the ECS service in the AWS console.

9. Create a Fargate cluster, which can be used for hosting Qualys Headless Scanners.

i) While creating a cluster, provide a cluster name. Refer to the following screenshot.
ii) specify the VPC that the cluster can operating in.

![Networking](image)

iii) For infrastructure, leave all the checkboxes blank for Fargate-type deployment.

![Infrastructure](image)

10. Create a Task that can encompass a single scan.

i) Provide a descriptive name for the task.

![Configure task definition and containers](image)

ii) In the next section, provide a name for the container running the Headless Scanner.

Supply the Qualys Headless Scanner image URL. Ideally, this should be in the same region as the cluster. You need to remove any port mapping for the container, and two environment variables need to be created in the environment variables sections.

- **SCAN_INPUT_FILE** variable should be configured to point to the scan specification file in s3 uri format.
- **SCAN_OUTPUT_DIR** variable should be configured to point to the s3 bucket which can be used to store scan results.
iii) Click **Next** to proceed further.

iv) In the task environment configuration step, select the following settings:
- Operating System: Linux/X86_64
- CPU: 4vCPU
- Memory 15GB
- Task role: ecsTaskExecutionRole (create above)
- Task Execution Rule: ecsTaskExecutionRole (create above)
**Note:** The CPU, memory, and storage should suit the scan to perform the scan task. Use one vCPU for each 10-15 target hosts and 1Gb for every 15 hosts; for a scan with 40 parallel IPs, 4 vCPUs and 4Gb (rounded up) should be used.

![Configure environment, storage, monitoring, and tags](image)

v) Select the storage as 50GiB.

![Storage - optional](image)

vi) Review all the settings and click **Next > Create**.

**Launching a Scan**

To launch a scan using Headless Scanner in Fargate the newly created ECS task must be deployed in newly created ECS cluster.

1. Navigate to **ECS Service** in the AWS console.
2. Select **Task definitions** in the left-hand side menu.
3. Select the task that was created for deploying the Headless Scanner in step #6 of the setup section and enable the checkbox of the task revision. This can activate the deploy button, click **Deploy > Run task**.

4. In the task creation step, select the cluster that was created for Headless Scanner deployment.

5. Select the correct network where the container can be deployed (VPC and subnets) and click **Create**.
View Scan Result

Once the scanner completes the scanning, the result can be viewed based on `OUTPUT_FORMAT` in the S3 bucket.

1. When `<OUTPUT_FORMAT>xml</OUTPUT_FORMAT>`. The result file is uploaded as `<IP>.xml`.

2. When `<OUTPUT_FORMAT>json</OUTPUT_FORMAT>`. The result file is uploaded as `<IP>.json`.

3. When `<OUTPUT_FORMAT>json_and_xml</OUTPUT_FORMAT>` or `<OUTPUT_FORMAT>xml_and_json</OUTPUT_FORMAT>`. The result files are uploaded as `<IP>.xml` and `<IP>.json`. 

Monitoring Scan Lifecycle

When launching a Qualys Headless Scanner task in ECS, the task status conveys if the task has been successfully deployed or not. The above information can be accessed through the AWS console or AWS CLI.

The following are the steps given for ECS task transitions:

1. The status of the task can be obtained using the following command.

   ```bash
   aws ecs describe-tasks --cluster hs-cluster --task afdf485ad7a34fb3ad3043da77e51baa | jq ".[].[].lastStatus" "RUNNING"
   ```

   A successfully launched scan can complete and produce a scan_status.xml file in the S3 bucket that was specified for output. The scan status can be one of these values: "running," "post_processing," "success," or "error."
"success" and "failed" are final states, while "running" and "post_processing" are transitional states between "running" and "success" or "failed."

Qualys Headless Scanner Docker Repository

Qualys Headless Scanner is published in AWS ECR private repository. You can access the private repository by providing your AWS account ID to Qualys. Qualys can grant access to the ECR repository for active users only.

The Qualys Headless Scanner docker image can be accessed by pulling the image using the following URL.

docker pull 531xxx.dkr.ecr.us-east-1.amazonaws.com/hsengine/hsengine:latest

**Note:** The URL needs to be modified for that region in order to pull the image in other AWS regions.

The name template for the image looks like the following:

531xxx.dkr.ecr.<Region-ID>.amazonaws.com/hsengine:latest

Where the <Region-ID> should be replaced by the AWS region id, the region id can be obtained using the AWS CLI command.

aws ec2 describe-regions |jq ".Regions[].Endpoint"

Custom MetaData Injection

Headless Scanner can be instructed to provide customer-supplied data tokens in the result files in order to facilitate no lookup results parsing. This option can be used in cases where scan results from multiple different network ranges are aggregated together into a single storage area. While parsing the results, there could be ambiguity as to which network the results belong to. To resolve these types of ambiguities, you can supply data tokens which can be copied into the results files.

The structure of custom meta-data sections is as follows:

- CUSTOM_METADATA- This is the main tag for this section.
Headless Scanner Support for AWS ECS

- **SCAN**: This sub-section contains data tokens that are intended to be copied only into scan_status.xml file and are relevant for the overall scan.
  - **DATA**: This element describes the data token in key-value pair model. The KEY is a unique attribute that names the data token, while the value of DATA element is the data token. The value of the key can be lowercase/uppercase ASCII letters, _ (underline), and digits. The hyphen or dash character - is not allowed in a variable name in the Bash shell.

- **HOST**: This sub-section contains data tokens that are intended to be copied only into individual target results.
  - **DATA**: This element describes the data token in key-value pair model. The KEY is a unique attribute that names the data token, while the value of DATA element is the data token.
  - **type**: possible values: mapped, static. If the type is mapped
  - **lang**: possible values: jq. This identified the engine that is used to process the map file
  - **update_frequency (optional)**: freshness of the mapping file in seconds. This setting indicates that if the cache file modification time is less than the specified value, the mapping file cannot be retrieved from remote location (S3, etc.). Default value is 300 seconds.

Here is an example:

```xml
<REPORT>
...<CUSTOM_METADATA>
  <SCAN>
   <DATA key="scan-metadata">Scan requested by SecOps</DATA>
   <DATA key="launched by">jdoe@acme.com</DATA>
  </SCAN>
  <HOST>
   <DATA key="network">Eng Network</DATA>
  </HOST>
  </CUSTOM_METADATA>
<REPORT>
```

The jq retrieves certain records or attribute-value pairs from the JSON file. jq lets you define variables using expression and use $ syntax to substitute the variable value into the expression.

Here is an example to understand the jq variable substitution.
In the above example, `ip` can match the objects where the value of the `ip` field equals the value of `$TARGET_IP` variable or `site_id`. However, `"${site_id}"` is not a valid jq variable substitution, considered as a literal string. The final jq command will fetch the ethernet address in the JSON file with an `ip` field equal to 10.20.31.38.

Here is another example of injecting the metadata through a JSON file:

```xml
<REPORT>
...
<CUSTOM_METADATA>
....
<HOST>
  <DATA key="site_id2">SITE-1</DATA>
  <DATA type="mapped" lang="jq" key="mac_address" update_frequency="300">s3://hs-scanspec/$site_id2.json?[]|select(.ip==$TARGET_IP and .whid==$site_id2)|.ethernet</DATA>
</HOST>
</CUSTOM_METADATA>
</REPORT>
```

The key can substitute the filename. Here the value of key `$site_id2` can be replaced by `SITE-1`.

In the following example, the expected mapping `SITE-1.json` file format could look like this:

```json
[
  {
    "ethernet" : "00:01:02:00:01:01",
    "ip" : "192.168.0.0",
    "whid" : "SITE-1"
  },
  {
    "ethernet" : "00:01:02:00:01:02",
    "ip" : "192.168.0.1",
    "whid" : "SITE-1"
  },
  ...
]
```
The filter specified can select the record with the matching IP address and the .whid to extract the .ethernet field from the above SITE-1.json file. Here the TARGET_IP is a variable that the Headless Scanner supplies.

The resulting metadata can look this:

```xml
<HOST>
  ....
  <CUSTOM_METADATA>
    <DATA key="site-id">safe-site 1</DATA>
    <DATA key="site-id2">safe-site 2</DATA>
    <DATA key="mac_address">00:01:02:00:01:01</DATA>
  </CUSTOM_METADATA>
</HOST>
```

**Secondary Report**

Qualys Headless Scanner can be instructed to produce a secondary scan results report which is intended for processing by operators with limited access to privileged information. The secondary scan result report is a subset of the full report normally produced by the Headless Scanner. You can have the ability to control which sections of the full report are included in the secondary report.

The following environment variable needs to be customized to enable this feature.

- **SECONDARY_OUTPUT_DIR**- This is an environment variable that points to the s3 URI where the secondary report should be generated. In the future, we will add the support to specify the local disk path as well.

Here is an example:

```bash
docker run \
  -it \
  --rm \n  --ulimit core=-1 \n  --security-opt seccomp=unconfined \n  --name vm_scanner \n  --net=host \n  -v $(pwd)/:/var/qualys \n  -v /tmp:/usr/share/qualys/ \n  --env SCAN_INPUT_FILE="/var/qualys/headless_vm_spec.xml" \n  --env SCAN_OUTPUT_DIR="custom" \n  --env SECONDARY_OUTPUT_DIR="secondary" \
```
In the above example, the secondary output can be generated in /tmp/secondary directory. The structure of the secondary report sections is as follows:

- **FILE_SUFFIX**: This element is a string that will be appended to the IP XML result file under SECONDARY_OUTPUT_DIR. For example, `<IP>_discovery.xml` file.
- **INCLUDED_HEADERS**: This element is a comma-separated list of header elements in the full report which should be included in the secondary report. If this element is omitted from the scan specification file, all headers are included.
- **INCLUDED_QIDS**: This is a comma-separated list of QIDs from the full report, which should be included in the secondary report. If this element is omitted from the scan specification file, all QIDs are included in the secondary report.

Here is an example of specification file:

```xml
<REPORT>
  ...
  <SECONDARY_REPORT>
    <FILE_SUFFIX>discovery</FILE_SUFFIX>
    <INCLUDED_QIDS>6,45038</INCLUDED_QIDS>
    <INCLUDED_HEADERS>IP_ADDRESS,OS,DEVICETYPE</INCLUDED_HEADERS>
  </SECONDARY_REPORT>
</REPORT>
```

The resulting secondary report can be created with the filename "<IP>_discovery.xml" in the specified S3 bucket.

```xml
<HOST>
  <IP_ADDRESS>10.20.32.160</IP_ADDRESS>
  <OS>EulerOS / Ubuntu / Fedora / Tiny Core Linux / Linux 3.x / IBM / FortiSOAR</OS>
  <DEVICETYPE>Unix</DEVICETYPE>
  <VULN>
    <QID>6</QID>
    <RESULT>
      #table IP_address Host_name 10.20.32.160 No_registered_hostname
    </RESULT>
  </VULN>
</HOST>
```
Headless Scanner Support for AWS ECS

<QID>45038</QID>
<Result>
</Result>
<Info>
<Duration>246</Duration>
<Startime>1682341758</Startime>
<Stoptime>1682342004</Stoptime>
</Info>
</Vuln>
</Host>
Appendix A

This is the XSD schema of the Scan Specification file.

```xml
<?xml version="1.0"?>
<!--
Revision: 0.7.0
Date: 02/03/22
Author: tgevorgyan@qualys.com

TODO: Add restrictions enforced by QWEB.
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- +++++++++++++++ QBOOLEAN +++++++++++++++++++ -->
  <xs:simpleType name="QBOOLEAN">
    <xs:restriction base="xs:string">
      <xs:enumeration value="TRUE" />  
      <xs:enumeration value="True" />  
      <xs:enumeration value="true" /> 
      <xs:enumeration value="1" />   
      <xs:enumeration value="FALSE" /> 
      <xs:enumeration value="False" /> 
      <xs:enumeration value="false" /> 
      <xs:enumeration value="0" /> 
    </xs:restriction>
  </xs:simpleType>
  <!-- +++++++++++++++ PortsTypeEnum +++++++++++++++++++ -->
  <xs:simpleType name="PortsTypeEnum">
    <xs:restriction base="xs:string">
      <xs:enumeration value="none" /> 
      <xs:enumeration value="custom" /> 
      <xs:enumeration value="light" /> 
      <xs:enumeration value="standard" /> 
      <xs:enumeration value="full" /> 
    </xs:restriction>
  </xs:simpleType>
  <!-- +++++++++++++++ IntensityEnum +++++++++++++++++++ -->
  <xs:simpleType name="IntensityEnum">
    <xs:restriction base="xs:string">
      <xs:enumeration value="light" /> 
      <xs:enumeration value="standard" /> 
      <xs:enumeration value="full" /> 
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```
Appendix A

<x:simpleType name="BasicIGEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="all" />
    <xs:enumeration value="register" />
    <xs:enumeration value="netblockonly" />
    <xs:enumeration value="none" />
  </xs:restriction>
</xs:simpleType>

<!-- +++++++++++++++ PerformanceEnum +++++++++++++++++++ -->
<x:simpleType name="PerformanceEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Low" />
    <xs:enumeration value="Normal" />
    <xs:enumeration value="High" />
    <xs:enumeration value="Custom" />
  </xs:restriction>
</xs:simpleType>

<!-- +++++++++++++++ PacketDelayEnum +++++++++++++++++++ -->
<x:simpleType name="PacketDelayEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Minimum" />
    <xs:enumeration value="Short" />
    <xs:enumeration value="Medium" />
    <xs:enumeration value="Long" />
    <xs:enumeration value="Maximum" />
  </xs:restriction>
</xs:simpleType>

<!-- +++++++++++++++ IntegerListType +++++++++++++++++++ -->
<xs:simpleType name="IntegerListType">
  <xs:restriction base="xs:string">
    <xs:pattern value="[0-9]+( *, *[0-9]+ *)*/">
  </xs:restriction>
</xs:simpleType>

<!-- +++++++++++++++ AdditionalPortsType +++++++++++++++++++ -->
<xs:complexType name="AdditionalPortsType">
  <xs:sequence>
    <xs:element name="HAS_ADDITIONAL" type="xs:boolean"/>
    <xs:element name="ADDITIONAL_PORTS">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:pattern value=".*"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<!-- +++++++++++++++ UDPPortsProfileType +++++++++++++++++++ -->
<xs:complexType name="UDPPortsProfileType">
  <xs:sequence>
    <xs:element name="UDP_PORTS_TYPE" type="PortsTypeEnum"/>
    <xs:element name="UDP_PORTS_ADDITIONAL" type="AdditionalPortsType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<!-- +++++++++++++++ HostDiscoUDPPortsType +++++++++++++++++++ -->
<xs:complexType name="HostDiscoUDPPortsType">
  <xs:sequence>
    <xs:element name="STANDARD_SCAN" type="xs:boolean" minOccurs="0"/>
    <xs:element name="CUSTOM_PORT" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<!-- +++++++++++++++ TCPPortsProfileType +++++++++++++++++++ -->
<xs:complexType name="TCPPortsProfileType"/>
Appendix A

```xml
<xsd:sequence>
  <xsd:element name="TCP_PORTS_TYPE" type="PortsTypeEnum"/>
  <xsd:element name="TCP_PORTS_ADDITIONAL" type="AdditionalPortsType" minOccurs="0"/>
  <xsd:element name="THREE_WAY_HANDSHAKE" type="xs:boolean"/>
</xsd:sequence>
</xsd:complexType>

<!-- +++++++++++++++ PortsProfileType +++++++++++++++++++ -->
<xsd:complexType name="PortsProfileType">
  <xsd:sequence>
    <xsd:element name="TCP_PORTS" type="TCPPortsProfileType"/>
    <xsd:element name="UDP_PORTS" type="UDPPortsProfileType"/>
    <xsd:element name="AUTHORITATIVE_OPTION" minOccurs="0" type="xs:boolean"/>
  </xsd:sequence>
</xsd:complexType>

<!-- +++++++++++++++ PerformanceProfileType +++++++++++++++++++ -->
<xsd:complexType name="PerformanceProfileType">
  <xsd:sequence>
    <xsd:element name="PARALLEL_SCALING" minOccurs="0"/>
    <xsd:element name="OVERALL_PERFORMANCE" type="PerformanceEnum"/>
    <xsd:element name="HOSTS_TO_SCAN" minOccurs="0">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="EXTERNAL_SCANNERS" type="xs:positiveInteger" minOccurs="0"/>
          <xsd:element name="SCANNER_APPLIANCES" type="xs:positiveInteger" minOccurs="0"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="PARALLEL_Targets" type="xs:positiveInteger" minOccurs="0"/>
    <xsd:element name="PROCESSES_TO_RUN"/>
  </xsd:sequence>
</xsd:complexType>
```
<xs:element name="TOTAL_PROCESSES" type="xs:positiveInteger" />
<xs:element name="HTTP_PROCESSES" type="xs:positiveInteger" />
  </xs:sequence>
</xs:complexType>

<xs:element name="PACKET_DELAY" type="PacketDelayEnum"/>

<xs:element name="PORT_SCANNING_AND_HOST_DISCOVERY" type="IntensityEnum"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="CustomPasswordEntryType">
<xs:sequence>
  <xs:element name="CUSTOM" minOccurs="1" maxOccurs="100">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ID" type="xs:positiveInteger" />
        <xs:element name="TITLE" type="xs:string" />
        <xs:element name="TYPE" type="xs:string" />
        <xs:element name="LOGIN_PASSWORD" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:sequence>
</xs:complexType>

<xs:complexType name="PasswordBruteforceType">
<xs:sequence>
  <xs:element name="SYSTEM">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="HAS_SYSTEM" type="xs:boolean" />
        <xs:element name="SYSTEM_LEVEL">
          <xs:simpleType>
            <xs:restriction base="xs:string">
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:sequence>
</xs:complexType>
Appendix A

<x:enumeration value="Minimal" />
<x:enumeration value="Limited" />
<x:enumeration value="Standard" />
<x:enumeration value="Exhaustive" />
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:complexType>
</xs:element>
<x:element name="CUSTOM_LIST" type="CustomPasswordEntryType"
minOccurs="0" />
</xs:sequence>
</xs:complexType>

<!- +++++++++++++++ VulnListType +++++++++++++++++++ ->
<x:complexType name="VulnListType">
</xs:sequence>
</xs:complexType>

<!- +++++++++++++++ ScanProfileType +++++++++++++++++++ ->
<x:complexType name="ScanProfileType">
</xs:choice maxOccurs="unbounded">
</xs:element name="PORTS" type="PortsProfileType"/>
</xs:element name="SCAN_DEAD_HOSTS" type="xs:boolean"/>
</xs:element name="PERFORMANCE" type="PerformanceProfileType"/>
</xs:element name="LOAD_BALANCER_DETECTION" type="xs:boolean"/>
</xs:element name="VULNERABILITY_LIST" type="VulnListType"
minOccurs="0" />
</xs:element name="PASSWORD_BRUTE_FORCING"
type="PasswordBruteforceType" minOccurs="0" />
</xs:element name="AUTHENTICATION" type="xs:string"
minOccurs="0"/>
</xs:element name="ADDL_CERT_DETECTION" type="xs:boolean"/>
</xs:element name="LITE_OS_SCAN" type="xs:boolean" minOccurs="0"/>
</xs:element name="CUSTOM_HTTP_HEADER" minOccurs="0">
<xs:complexType>
  <xs:sequence>
    <xs:element name="VALUE" type="xs:string" minOccurs="0" />  
    <xs:element name="DEFINITION_KEY" type="xs:string" minOccurs="0" />  
    <xs:element name="DEFINITION_VALUE" type="xs:string" minOccurs="0" />
  </xs:sequence>
</xs:complexType>

<xs:element name="HOST_ALIVE_TESTING" minOccurs="0" type="xs:boolean"/>
<xs:element name="DO_NOT_OVERWRITE_OS" minOccurs="0" type="xs:boolean"/>
<xs:element name="TEST_AUTHENTICATION" minOccurs="0" type="xs:boolean"/>
</xs:choice>
</xs:complexType>

<!-- +++++++++++++++ TCPPortsType +++++++++++++++++++ -->
<xs:complexType name="TCPPortsType">
  <xs:sequence>
    <xs:element name="TCP_PORTS_STANDARD_SCAN" type="xs:boolean"/>
    <xs:element name="TCP_PORTS_ADDITIONAL" type="AdditionalPortsType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<!-- +++++++++++++++ HostDiscoTCPPortsType +++++++++++++++++++ -->
<xs:complexType name="HostDiscoTCPPortsType">
  <xs:sequence>
    <xs:element name="STANDARD_SCAN" type="xs:boolean"/>
    <xs:element name="TCP_ADDITIONAL" type="AdditionalPortsType" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>

<!-- +++++++++++++++ TrafficDumpEnum +++++++++++++++++++ -->
<xs:simpleType name="TrafficDumpEnum">

97
Appendix A

<x:s:restriction base="xs:string">
  <xs:enumeration value="yes" />
  <xs:enumeration value="Yes" />
  <xs:enumeration value="YES" />
  <xs:enumeration value="on" />
  <xs:enumeration value="On" />
  <xs:enumeration value="ON" />
  <xs:enumeration value="all" />
  <xs:enumeration value="no" />
  <xs:enumeration value="No" />
  <xs:enumeration value="NO" />
  <xs:enumeration value="off" />
  <xs:enumeration value="Off" />
  <xs:enumeration value="OFF" />
</xs:restriction>
</xs:simpleType>

<!-- +++++++++++++++  MapProfileType+++++++++++++++++++ -->
<x:s:complexType name="MapProfileType">
  <xs:sequence>
    <xs:element name="BASIC_INFO_GATHERING_ON" type="BasicIGEnum"/>
    <xs:element name="TCP_PORTS">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="TCP_PORTS_STANDARD_SCAN" type="xs:boolean"/>
          <xs:element name="TCP_PORTS_ADDITIONAL" type="AdditionalPortsType" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="UDP_PORTS">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="UDP_PORTS_STANDARD_SCAN" type="xs:boolean" minOccurs="0"/>
          <xs:element name="UDP_PORTS_ADDITIONAL" type="AdditionalPortsType" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:element name="MAP_OPTIONS">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="PERFORM_LIVE_HOST_SWEEP" type="xs:boolean"/>
      <xs:element name="DISABLE_DNS_TRAFFIC" type="xs:boolean"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="MAP_PERFORMANCE">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="OVERALL_PERFORMANCE" type="PerformanceEnum"/>
      <xs:element name="MAP_PARALLEL">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="EXTERNAL_SCANNERS" type="xs:positiveInteger"/>
            <xs:element name="SCANNER_APPLIANCES" type="xs:positiveInteger"/>
            <xs:element name="NETBLOCK_SIZE" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="PACKET_DELAY" type="PacketDelayEnum"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="MAP_AUTHENTICATION">
  <xs:simpleType>  
</xs:element>
Appendix A

<xs:restriction base="xs:string">
    <xs:enumeration value="none" />
    <xs:enumeration value="vCenter" />
    <xs:enumeration value="VMware-ESXi" />
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>

<!-- +++++++++++++++ HostDiscoveryType +++++++++++++++++++ -->
<xs:complexType name="HostDiscoveryType">
    <xs:sequence>
        <xs:element name="TCP_PORTS" minOccurs="0" type="HostDiscoTCPPortsType"/>
        <xs:element name="UDP_PORTS" minOccurs="0" type="HostDiscoUDPPortsType"/>
        <xs:element name="ICMP" type="xs:boolean"/>
    </xs:sequence>
</xs:complexType>

<!-- +++++++++++++++ BlockResourcesType +++++++++++++++++++ -->
<xs:complexType name="BlockResourcesType">
    <xs:sequence>
        <xs:element name="WATCHGUARD_DEFAULT_BLOCKED_PORTS" type="xs:boolean" minOccurs="0"/>
        <xs:element name="CUSTOM_PORT_LIST" type="xs:string" minOccurs="0"/>
        <xs:element name="ALL_REGISTERED_IPS" type="xs:boolean" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<!-- +++++++++++++++ AdditionalProfileType +++++++++++++++++++ -->
<xs:complexType name="AdditionalProfileType">
    <xs:sequence>
        <xs:element name="HOST_DISCOVERY" type="HostDiscoveryType"/>
        <xs:element name="BLOCK_RESOURCES" minOccurs="0" type="BlockResourcesType"/>
        <xs:element name="PACKET_OPTIONS" type="PacketOptionsType"/>
    </xs:sequence>
</xs:complexType>
<!-- +++++++++++++++ PacketOptionsType +++++++++++++++++++ -->
<xs:complexType name="PacketOptionsType">
  <xs:sequence>
    <xs:element name="IGNORE_FIREWALL_GENERATED_TCP_RST" type="xs:boolean"/>
    <xs:element name="IGNORE_ALL_TCP_RST" type="xs:boolean"/>
    <xs:element name="IGNORE_FIREWALL_GENERATED_TCP_SYN_ACK" type="xs:boolean"/>
    <xs:element name="NOT_SEND_TCP_ACK_OR_SYN_ACK_DURING_HOST_DISCOVERY" type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>

<!-- +++++++++++++++ AuthOptionsType +++++++++++++++++++ -->
<xs:complexType name="AuthOptionsType">
  <xs:sequence>
    <xs:element name="MULTIPLE_AUTH_RECORDS" type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>

<!-- +++++++++++++++ OptionProfileType +++++++++++++++++++ -->
<xs:complexType name="OptionProfileType">
  <xs:sequence>
    <xs:element name="SCAN" type="ScanProfileType"/>
    <xs:element name="MAP" minOccurs="0" type="MapProfileType"/>
    <xs:element name="ADDITIONAL" minOccurs="0" type="AdditionalProfileType"/>
    <xs:element name="AUTHENTICATION_OPTIONS" minOccurs="0" type="AuthOptionsType"/>
  </xs:sequence>
</xs:complexType>

<!-- +++++++++++++++ ScanType +++++++++++++++++++ -->
<xs:complexType name="ScanType">
  <xs:sequence>
    <xs:element name="TARGETS"/>
  </xs:sequence>
</xs:complexType>
<xs:sequence>
  <xs:element name="TARGET" minOccurs="1">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="IP" minOccurs="0" maxOccurs="4096" type="xs:string"/>
        <xs:element name="IPV6" minOccurs="0" maxOccurs="4096" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:sequence>
    <xs:element name="SCAN_DURATION_LIMIT" minOccurs="0" maxOccurs="1" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="CONF">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="TYPE">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="vm"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="QID_EXCLUSIONS" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:choice maxOccurs="unbounded">
      <xs:element name="CANCELLED" type="QBOOLEAN"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
<xs:complexType name="MSSQLAuthRecordType">
<xs:sequence>
  <xs:element name="RECORD" minOccurs="1" maxOccurs="unbounded">
    <xs:complexType>
      <xs:choice maxOccurs="unbounded">
        <xs:element name="HIDE_PASSWORDS" type="QBOOLEAN" minOccurs="0" maxOccurs="1"/>
        <xs:element name="META" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice maxOccurs="unbounded">
              <xs:element name="TITLE" type="QBOOLEAN" minOccurs="0" maxOccurs="1"/>
              <xs:element name="SEVERITY" type="QBOOLEAN" minOccurs="0" maxOccurs="1"/>
              <xs:element name="CVSS" type="QBOOLEAN" minOccurs="0" maxOccurs="1"/>
              <xs:element name="CVSS3" type="QBOOLEAN" minOccurs="0" maxOccurs="1"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
        <xs:element name="DEBUG" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:choice maxOccurs="unbounded">
              <xs:element name="LOG" type="QBOOLEAN" minOccurs="0" maxOccurs="1"/>
              <xs:element name="TRAFFIC_DUMP" type="TrafficDumpEnum" minOccurs="0" maxOccurs="1"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:choice>
    </xs:complexType>
  </xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType>
  <xs:choice maxOccurs="unbounded">
    <xs:element name="AUTH_NAME" type="xs:base64Binary"/>
    <xs:element name="AUTH_ID" type="xs:base64Binary"/>
    <xs:element name="AUTH_TYPE" type="xs:positiveInteger"/>
    <xs:element name="DATABASE" type="xs:base64Binary"/>
    <xs:element name="DOMAIN" type="xs:base64Binary"/>
    <xs:element name="HOSTID" type="QBOOLEAN"/>
    <xs:element name="INSTANCE" type="xs:base64Binary"/>
    <xs:element name="IP_RANGE" type="xs:string"/>
    <xs:element name="PORT" type="xs:positiveInteger"/>
    <xs:element name="PASSWORD" type="xs:base64Binary"/>
    <xs:element name="USERNAME" type="xs:base64Binary"/>
  </xs:choice>
</xs:complexType>

<xs:complexType name="NTAuthRecordType">
  <xs:sequence>
    <xs:element name="RECORD" minOccurs="1" maxOccurs="unbounded">
      <xs:complexType>
        <xs:choice maxOccurs="unbounded">
          <xs:element name="AUTH_NAME" type="xs:base64Binary"/>
          <xs:element name="AUTH_ID" type="xs:base64Binary"/>
          <xs:element name="DOMAIN" type="xs:base64Binary"/>
          <xs:element name="HOSTID" type="QBOOLEAN"/>
          <xs:element name="PASSWORD" type="xs:base64Binary"/>
          <xs:element name="USERNAME" type="xs:base64Binary"/>
        </xs:choice>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="ADAuthRecordType">
  <xs:sequence>
    <xs:element name="RECORD" minOccurs="1" maxOccurs="unbounded">
      <xs:complexType>
        <xs:choice maxOccurs="unbounded">
          <xs:element name="AUTH_NAME" type="xs:base64Binary"/>
          <xs:element name="AUTH_ID" type="xs:base64Binary"/>
        </xs:choice>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
<xs:element name="AUTH_PROTOCOLS" type="xs:string"/>
<xs:element name="DOMAIN" type="xs:base64Binary"/>
<xs:element name="PASSWORD" type="xs:base64Binary"/>
<xs:element name="TRUST" type="xs:boolean"/>
<xs:element name="USERNAME" type="xs:base64Binary"/>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="SSHAuthRecordType">
<xs:sequence>
<xs:element name="RECORD" minOccurs="1" maxOccurs="unbounded">
<xs:complexType>
<xs:choice maxOccurs="unbounded">
<xs:element name="AUTH_NAME" type="xs:base64Binary"/>
<xs:element name="AUTH_ID" type="xs:base64Binary"/>
<xs:element name="ClearedText" type="xs:boolean"/>
<xs:element name="IP_RANGE" type="xs:string"/>
<xs:element name="DOMAIN" type="xs:base64Binary"/>
<xs:element name="HOSTID" type="QBOOLEAN"/>
<xs:element name="HOSTID_PATH" type="xs:base64Binary"/>
<xs:element name="PASSWORD" type="xs:base64Binary"/>
<xs:element name="USERNAME" type="xs:base64Binary"/>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="SSH2AuthRecordType">
<xs:sequence>
<xs:element name="RECORD" minOccurs="1" maxOccurs="unbounded">
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="SNMPAuthRecordType">
<xs:sequence>
<xs:element name="RECORD" minOccurs="1" maxOccurs="unbounded">
<xs:complexType>
<xs:choice maxOccurs="unbounded">
<xs:element name="AUTH_NAME" type="xs:base64Binary"/>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:element name="IP_RANGE" type="xs:string"/>
<xs:element name="COMMUNITIES" type="xs:base64Binary"/>
<xs:element name="VERSION" minOccurs="0" type="xs:positiveInteger"/>
<xs:element name="USER" minOccurs="0" type="xs:base64Binary"/>
<xs:element name="AUTH_PASS" minOccurs="0" type="xs:base64Binary"/>
<xs:element name="PRIV_PASS" minOccurs="0" type="xs:base64Binary"/>
<xs:element name="AUTH_ALG" minOccurs="0" type="xs:base64Binary"/>
<xs:element name="PRIV_ALG" minOccurs="0" type="xs:base64Binary"/>
<xs:element name="SEC_ENG" minOccurs="0" type="xs:base64Binary"/>
<xs:element name="CTX_ENG" minOccurs="0" type="xs:base64Binary"/>
<xs:element name="CONTEXT" minOccurs="0" type="xs:base64Binary"/>
<xs:element name="AUTH_ID" type="xs:base64Binary"/>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="UNIXAuthRecordType">
<xs:sequence>
<xs:element name="RECORD" minOccurs="1" maxOccurs="unbounded">
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="WEBLOGICAuthRecordType">
<xs:sequence>
<xs:element name="RECORD" minOccurs="1" maxOccurs="unbounded">
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="WEBAuthRecordType">
<xs:sequence>
<xs:element name="RECORD" minOccurs="1" maxOccurs="unbounded">
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType>
    <xs:sequence>
        <xs:element name="RECORD" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
</xs:complexType>
<xs:complexType name="DOCKERAuthRecordType">
    <xs:sequence>
        <xs:element name="RECORD" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
</xs:complexType>
<xs:complexType name="ORACLEAuthRecordType">
    <xs:sequence>
        <xs:element name="RECORD" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
</xs:complexType>
<xs:complexType name="ORACLELISTENERAuthRecordType">
    <xs:sequence>
        <xs:element name="RECORD" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
</xs:complexType>
<xs:complexType name="POSTGRESQLAuthRecordType">
    <xs:sequence>
        <xs:element name="RECORD" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
</xs:complexType>
<xs:complexType name="DB2AuthRecordType">
    <xs:sequence>
        <xs:element name="RECORD" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
</xs:complexType>
<xs:complexType name="MONGODBAuthRecordType">
    <xs:sequence>
        <xs:element name="RECORD" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
</xs:complexType>
<xs:complexType name="AWSAuthRecordType">
    <xs:sequence>
        <xs:element name="RECORD" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
Appendix A

<xs:complexType>

<!-- +++++++++++++++ AuthenticationRecordsType +++++++++++++++++++ -->
<xs:complexType name="AuthenticationRecordsType">
  <xs:choice maxOccurs="unbounded">
    <xs:element name="MSSQL" type="MSSQLAuthRecordType"/>
    <xs:element name="NT" type="NTAuthRecordType"/>
    <xs:element name="AD" type="ADAuthRecordType"/>
    <xs:element name="SSH" type="SSHAuthRecordType"/>
    <xs:element name="SSH2" type="SSH2AuthRecordType"/>
    <xs:element name="SNMP" type="SNMPAuthRecordType"/>
    <xs:element name="UNIX" type="UNIXAuthRecordType"/>
    <xs:element name="WEBLOGIC" type="WEBLOGICAuthRecordType"/>
    <xs:element name="WEB" type="WEBAuthRecordType"/>
    <xs:element name="DOCKER" type="DOCKERAuthRecordType"/>
    <xs:element name="ORACLE" type="ORACLEAuthRecordType"/>
    <xs:element name="ORACLE_LISTENER" type="ORACLELISTENERAuthRecordType"/>
    <xs:element name="POSTGRESQL" type="POSTGRESQLAuthRecordType"/>
    <xs:element name="DB2" type="DB2AuthRecordType"/>
    <xs:element name="MONGODB" type="MONGODBAuthRecordType"/>
    <xs:element name="AWS" type="AWSAuthRecordType"/>
  </xs:choice>
</xs:complexType>

<!-- +++++++++++++++ VirtualHostList +++++++++++++++++++ -->
<xs:complexType name="VirtualHostList">
  <xs:sequence>
    <xs:element name="VIRTUAL_HOST" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="IP" type="xs:string"/>
          <xs:element name="PORT" type="xs:string"/>
          <xs:element name="FQDN" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<!-- +++++++++++++++ HeadlessScannerJobType +++++++++++++++++++ -->

<xs:complexType name="HeadlessScannerJobType">
  <xs:sequence>
    <xs:element name="JOB_TYPE">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="scan" />
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="SCAN" type="ScanType" />
    <xs:element name="REPORT" type="ReportType" minOccurs="0" />  
    <!-- xs:element name="AUTHS" minOccurs="0" maxOccurs="1" type="AuthenticationRecordsType" -->
    <xs:element name="VIRTUAL_HOST_LIST" type="VirtualHostList" minOccurs="0" maxOccurs="1" />
    <xs:element name="AUTHS" minOccurs="0" maxOccurs="1" />
    <xs:element name="CUSTOM_SHELL_SCRIPTS" minOccurs="0" maxOccurs="1" />
  </xs:sequence>
</xs:complexType>

<xs:element name="HEADLESS_SCANNER_JOB" type="HeadlessScannerJobType"/>
</xs:schema>
Appendix A

Sample Scan Options

```xml
<OPTION_PROFILE>
  <SCAN>
    <PORTS>
      <TCP_PORTS>
        <TCP_PORTS_TYPE>custom</TCP_PORTS_TYPE>
        <TCP_PORTS_ADDITIONAL>
          <HAS_ADDITIONAL>1</HAS_ADDITIONAL>
          <ADDITIONAL_PORTS>138,139,445</ADDITIONAL_PORTS>
        </TCP_PORTS_ADDITIONAL>
        <THREE_WAY_HANDSHAKE>0</THREE_WAY_HANDSHAKE>
      </TCP_PORTS>
      <UDP_PORTS>
        <UDP_PORTS_TYPE>custom</UDP_PORTS_TYPE>
        <UDP_PORTS_ADDITIONAL>
          <HAS_ADDITIONAL>1</HAS_ADDITIONAL>
          <ADDITIONAL_PORTS>138,139,445</ADDITIONAL_PORTS>
        </UDP_PORTS_ADDITIONAL>
      </UDP_PORTS>
    </PORTS>
    <SCAN_DEAD_HOSTS>0</SCAN_DEAD_HOSTS>
    <PERFORMANCE>
      <PARALLEL_SCALING>1</PARALLEL_SCALING>
      <OVERALL_PERFORMANCE>Custom</OVERALL_PERFORMANCE>
      <HOSTS_TO_SCAN>
        <EXTERNAL_SCANNERS>15</EXTERNAL_SCANNERS>
        <SCANNER_APPLIANCES>30</SCANNER_APPLIANCES>
      </HOSTS_TO_SCAN>
      <PROCESSES_TO_RUN>
        <TOTAL_PROCESSES>10</TOTAL_PROCESSES>
        <HTTP_PROCESSES>10</HTTP_PROCESSES>
      </PROCESSES_TO_RUN>
      <PACKET_DELAY>Minimum</PACKET_DELAY>
      <PORT_SCANNING_AND_HOST_DISCOVERY>Normal</PORT_SCANNING_AND_HOST_DISCOVERY>
    </PERFORMANCE>
  </SCAN>
</OPTION_PROFILE>
```
<SYSTEM_LEVEL>Exhaustive</SYSTEM_LEVEL>
</SYSTEM>

<CUSTOM_LIST>
  <CUSTOM>
    <ID>1002</ID>
    <TITLE>FTP-weak</TITLE>
    <TYPE>FTP</TYPE>
    <LOGIN_PASSWORD>L:anonymous,P:none</LOGIN_PASSWORD>
  </CUSTOM>
  <CUSTOM>
    <ID>1001</ID>
    <TITLE>SSH-weak</TITLE>
    <TYPE>SSH</TYPE>
    <LOGIN_PASSWORD>L:root,P:toor</LOGIN_PASSWORD>
  </CUSTOM>
  <CUSTOM>
    <ID>1003</ID>
    <TITLE>Windows-weak</TITLE>
    <TYPE>Windows</TYPE>
    <LOGIN_PASSWORD>L:administrator,P:admin</LOGIN_PASSWORD>
  </CUSTOM>
</CUSTOM_LIST>
</PASSWORD_BRUTE_FORCING>

<ADDL_CERT_DETECTION>0</ADDL_CERT_DETECTION>
<LITE_OS_SCAN>1</LITE_OS_SCAN>

<CUSTOM_HTTP_HEADER>
  <VALUE>CUSTOM_HEADER</VALUE>
  <DEFINITION_KEY>CUSTOM</DEFINITION_KEY>
  <DEFINITION_VALUE>Value</DEFINITION_VALUE>
</CUSTOM_HTTP_HEADER>

<HOST_ALIVE_TESTING>1</HOST_ALIVE_TESTING>
<DO_NOT_OVERWRITE_OS>1</DO_NOT_OVERWRITE_OS>
</SCAN>

<ADDITIONAL>
  <HOST_DISCOVERY>
    <TCP_PORTS>
      <STANDARD_SCAN>1</STANDARD_SCAN>
      <TCP_ADDITIONAL>
        <HAS_ADDITIONAL>1</HAS_ADDITIONAL>
        <ADDITIONAL_PORTS>3128</ADDITIONAL_PORTS>
      </TCP_ADDITIONAL>
    </TCP_PORTS>
  </HOST_DISCOVERY>
</ADDITIONAL>
Appendix A

</TCP_PORTS>
<UDP_PORTS>
  <CUSTOM_PORT>53</CUSTOM_PORT>
</UDP_PORTS>
<ICMP>1</ICMP>
</HOST_DISCOVERY>
<BLOCK_RESOURCES>
  <CUSTOM_PORT_LIST>111</CUSTOM_PORT_LIST>
  <ALL_REGISTERED_IPS>1</ALL_REGISTERED_IPS>
</BLOCK_RESOURCES>
<PACKET_OPTIONS>
  <IGNORE_FIREWALL_GENERATED_TCP_RST>1</IGNORE_FIREWALL_GENERATED_TCP_RST>
  <IGNORE_ALL_TCP_RST>1</IGNORE_ALL_TCP_RST>
  <IGNORE_FIREWALL_GENERATED_TCP_SYN_ACK>1</IGNORE_FIREWALL_GENERATED_TCP_SYN_ACK>
  <NOT_SEND_TCP_ACK_OR_SYN_ACK_DURING_HOST_DISCOVERY>1</NOT_SEND_TCP_ACK_OR_SYN_ACK_DURING_HOST_DISCOVERY>
</PACKET_OPTIONS>
</ADDITIONAL>
</OPTION_PROFILE>
Appendix B

This is the XSD schema of the results XML file.

<?xml version="1.0"?>
<!--
Revision: 0.1.0
Date: 08/04/20
Author: tgevorgyan@qualys.com

TODO: Add restrictions enforced by QWEB.
-->
<x:schema xmlns:xss="http://www.w3.org/2001/XMLSchema">
<!-- +++++++++++++++ QBOOLEAN +++++++++++++++++++ -->
<x:simpleType name="ProtocolEnum">
  <xs:restriction base="xs:string">
    <xs:enumeration value="tcp"/>
    <xs:enumeration value="udp"/>
  </xs:restriction>
</x:simpleType>

<!-- +++++++++++++++ QBOOLEAN +++++++++++++++++++ -->
<x:simpleType name="QBOOLEAN">
  <xs:restriction base="xs:string">
    <xs:enumeration value="TRUE"/>
    <xs:enumeration value="True"/>
    <xs:enumeration value="true"/>
    <xs:enumeration value="1"/>
    <xs:enumeration value="FALSE"/>
    <xs:enumeration value="False"/>
    <xs:enumeration value="false"/>
    <xs:enumeration value="0"/>
  </xs:restriction>
</x:simpleType>

<!-- +++++++++++++++ TrackingType +++++++++++++++++++ -->
<x:simpleType name="TrackingType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="ip"/>
    <xs:enumeration value="ipv6"/>
  </xs:restriction>
</x:simpleType>
Appendix B

<x:simpleType>
  <xs:restriction base="xs:string">
    <xs:enumeration value="netbios"/>
    <xs:enumeration value="dns"/>
  </xs:restriction>
</xs:simpleType>

<!-- +++++++++++++++ IPAddressStringType +++++++++++++++++++ -->
<x:simpleType name="IPAddressStringType">
  <xs:restriction base="xs:string">
    <xs:pattern value="((1?[0-9]?[0-9]|2[0-4][0-9]|25[0-5])\.)\.(1?[0-9]|[0-9]|2[0-4][0-9]|25[0-5])"/>
  </xs:restriction>
</xs:simpleType>

<!-- +++++++++++++++ CertificateType +++++++++++++++++++ -->
<x:complexType name="CertificateType">
  <xs:sequence>
    <xs:element name="CERTIFICATE" maxOccurs="unbounded">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:attribute name="format" type="xs:string" fixed="PEM"/>
            <xs:attribute name="fingerprint" type="xs:string"/>
            <xs:attribute name="parent" type="xs:string"/>
            <xs:attribute name="source" type="xs:string"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<!-- +++++++++++++++ ServiceType +++++++++++++++++++ -->
<x:complexType name="ServiceType">
  <xs:sequence>
    <xs:element name="NAME" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="TRANSPORT" minOccurs="0" type="xs:string"/>
    <xs:element name="UUID" minOccurs="0" type="xs:string"/>
    <xs:element name="VERSION" minOccurs="0" type="xs:string"/>
    <xs:element name="PIPE" minOccurs="0" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
<!-- +++++++++++++++ PortType +++++++++++++++++++ -->
<x:s:complexType name="PortType">
  <x:s:sequence>
    <x:s:element name="PORT" minOccurs="0" maxOccurs="unbounded">
      <x:s:complexType>
        <x:s:choice maxOccurs="unbounded">
          <x:s:element name="NUMBER" type="xs:positiveInteger" />
          <x:s:element name="DEFAULT_SERVICE" type="ServiceType" />
          <x:s:element name="SERVICE" type="ServiceType" />
        </x:s:choice>
      </x:s:complexType>
    </x:s:element>
  </x:s:sequence>
</x:s:complexType>

<!-- +++++++++++++++ PortsType +++++++++++++++++++ -->
<x:s:complexType name="PortsType">
  <x:s:choice maxOccurs="unbounded">
    <x:s:element name="TCP" type="PortType" minOccurs="0" />
    <x:s:element name="UDP" type="PortType" minOccurs="0" />
  </x:s:choice>
</x:s:complexType>

<!-- +++++++++++++++ InstanceType +++++++++++++++++++ -->
<x:s:complexType name="InstanceType">
  <x:s:sequence>
    <x:s:element name="INSTANCE" minOccurs="0" maxOccurs="unbounded">
      <x:s:complexType>
        <x:s:choice maxOccurs="unbounded">
          <x:s:element name="ID" type="xs:positiveInteger" />
          <x:s:element name="TECHNOLOGY" type="xs:string" />
          <x:s:element name="PORT" type="xs:positiveInteger" />
          <x:s:element name="INSTANCE_NAME" type="xs:string" />
          <x:s:element name="DATABASE" type="xs:string" minOccurs="0" />
          <x:s:element name="ERROR_MSG" type="xs:string" />
        </x:s:choice>
      </x:s:complexType>
    </x:s:element>
  </x:s:sequence>
</x:s:complexType>
<!-- +++++++++++++++ TrustedScanningType ++++++++++++++++ -->
<xs:complexType name="TrustedScanningType">
  <xs:choice maxOccurs="unbounded">
    <xs:element name="TYPE" type="xs:string"/>
    <xs:element name="STATUS" type="xs:integer"/>
    <xs:element name="AUTH_ID" type="xs:positiveInteger"/>
    <xs:element name="AUTH_NAME" type="xs:string"/>
    <xs:element name="INSTANCE" type="xs:string" minOccurs="0"/>
    <xs:element name="PORT" type="xs:positiveInteger"/>
    <xs:element name="ERROR_MSG" type="xs:string"/>
  </xs:choice>
</xs:complexType>

<!-- +++++++++++++++ VulnType ++++++++++++++++ -->
<xs:complexType name="VulnType">
  <xs:choice maxOccurs="unbounded">
    <xs:element name="QID" type="xs:positiveInteger"/>
    <xs:element name="I" type="xs:positiveInteger" minOccurs="0" maxOccurs="1"/>
    <xs:element name="PORT" type="xs:positiveInteger" minOccurs="0" maxOccurs="1"/>
    <xs:element name="PROTOCOL" type="ProtocolEnum" minOccurs="0" maxOccurs="1"/>
    <xs:element name="VIRTUAL_HOST" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="SSL" type="QBOOLEAN" minOccurs="0" maxOccurs="1"/>
    <xs:element name="RESULT" type="xs:string" maxOccurs="1"/>
    <xs:element name="FLAGS" type="xs:string"/>
    <xs:element name="INFO" minOccurs="0" maxOccurs="unbounded"/>
  </xs:choice>
</xs:complexType>

<!-- +++++++++++++++ InfoType ++++++++++++++++ -->
<xs:complexType name="InfoType">
  <xs:choice maxOccurs="unbounded">
    <xs:element name="DURATION" type="xs:positiveInteger"/>
    <xs:element name="STARTTIME" type="xs:positiveInteger"/>
    <xs:element name="STOPTIME" type="xs:positiveInteger"/>
  </xs:choice>
</xs:complexType>
<xs:complexType name="VulnsNotFoundType">
  <xs:sequence>
    <xs:element name="Q" type="xs:positiveInteger" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType> 

<!-- +++++++++++++++ HostType +++++++++++++++++++ -->
<xs:complexType name="HostType">
  <xs:choice maxOccurs="unbounded">
    <xs:element name="REPORT_REF" type="xs:string" />
    <xs:element name="REPORT_VERSION" type="xs:positiveInteger" />
    <xs:element name="TRACKING_METHOD" type="TrackingType" />
    <xs:element name="IP_ADDRESS" type="IPAddressStringType" />
    <xs:element name="DNS_HOSTNAME" type="xs:string" />
    <xs:element name="NETBIOS_HOSTNAME" type="xs:string" />
    <xs:element name="CPE" type="xs:string" />
    <xs:element name="OS" type="xs:string" />
    <xs:element name="DEVICETYPE" type="xs:string" />
    <xs:element name="CERTIFICATES" type="CertificateType" />
    <xs:element name="PORTS" type="PortsType" />
    <xs:element name="INSTANCES" type="InstanceType" />
    <xs:element name="BAD_VULNS" type="xs:string" />
    <xs:element name="TRUSTED_SCANNING" type="TrustedScanningType" maxOccurs="unbounded" minOccurs="0" />
    <xs:element name="VULNS NOT FOUND" type="VulnsNotFoundType" maxOccurs="unbounded" />
    <xs:element name="VULN" type="VulnType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:choice>
</xs:complexType>

<xs:element name="HOST" type="HostType"/>
Customizing Headless Scanner Input and Output

The Headless Scanner input file name, location and scanner output directory can be customized using environment variables. Two environment variables you can use for the customization:

- `SCAN_INPUT_FILE` environment variable if present is expected to point to the scan specification file.

- `SCAN_OUTPUT_DIR` environment variable, if present, it instructs the headless scanner where to generate the output/result files.

Here is an example:

```bash
docker run \
    -it \
    --rm \
    --ulimit core=-1 \
    --security-opt seccomp=unconfined \
    --name vm_scanner \
    --net=host \
    -v $(pwd)/:/var/qualys \
    -v /tmp/:/usr/share/qualys/ \
    --env SCAN_OUTPUT_DIR="custom" \ 
    --env SCAN_INPUT_FILE="/var/qualys/headless_vm_spec.xml" \ 
    art-hq.intranet.qualys.com:5001/qualys/headless/hsengine:0.6.429b3-2-beta
```

In the above example, the input file `headless_vm_spec.xml` can be read from the current directory, and the output can generated in `/tmp/custom` directory.