Steps To Implement Security Scan Requirements

Who should perform security scan

If you are working on driver code, Vuls and Lynis are needed.

If you are developing container based application code, Vuls, Lynis and Kuber-Hunter are needed.

If you are developing VM based application code, Vuls (testing setup inside each VM) and Lynis (testing setup inside each VM) are needed.

Vuls scan usually takes around 10 to 20 mins.

Kuber-Hunter usually takes about 10 mins.

Lynis scan usually takes about 2 mins.

Incubation Inclusion in Release Review: Security Requirements Criteria

- Releases typically occur every 6 months.
- A release will use the last TSC approved security requirements that were approved at least 6 month prior to the release.
- Exceptions must be submitted a minimum of 21 days prior to release
- Note: Critical vulnerabilities/security items, as categorized by the Akraino Security Sub-Committee, must be fixed even if found inside lock out window.

Release 4 (Target Date November 30, 2020) Incubation Requirements:

- Vuls PASS/FAIL Criteria, v1.0
- Lynis PASS/FAIL Criteria, v1.0
- Kube-Hunter PASS/FAIL Criteria, v1.0

Month	6/2020	7/2020	8/2020	9/2020	10/2020	11/2020	12/2020	1/2021
Release						Rel. 4		
Security Requirement Update	v. 1.0							
Minimum Security Requirement						v. 1.0		
Maximum Security Requirement						v. 1.0		

		Release 4 Minimum Security Requirement Lock Out Window				

Example:

Month	11	12	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6
Release								Rel #1						Rel #2						Rel #3
Security Requirement Update	Sec Ver. 1				Sec Ver. 2					Sec Ver. 3		Sec Ver. 4					Sec Ver. 5			
Minimum Security Requirement								Sec Ver. 1						Sec Ver. 2						Sec Ver. 4
Maximum Security Requirement								Sec Ver. 2						Sec Ver. 4						Sec Ver. 5
		Release #1 Security Requirement Lock Out Window						Release #2 Security Requirement Lock Out Window						Release #3 Security Requirement Lock Out Window						

Incubation Security Requirement (Release Based)

Maturity Review: Security Requirements Criteria

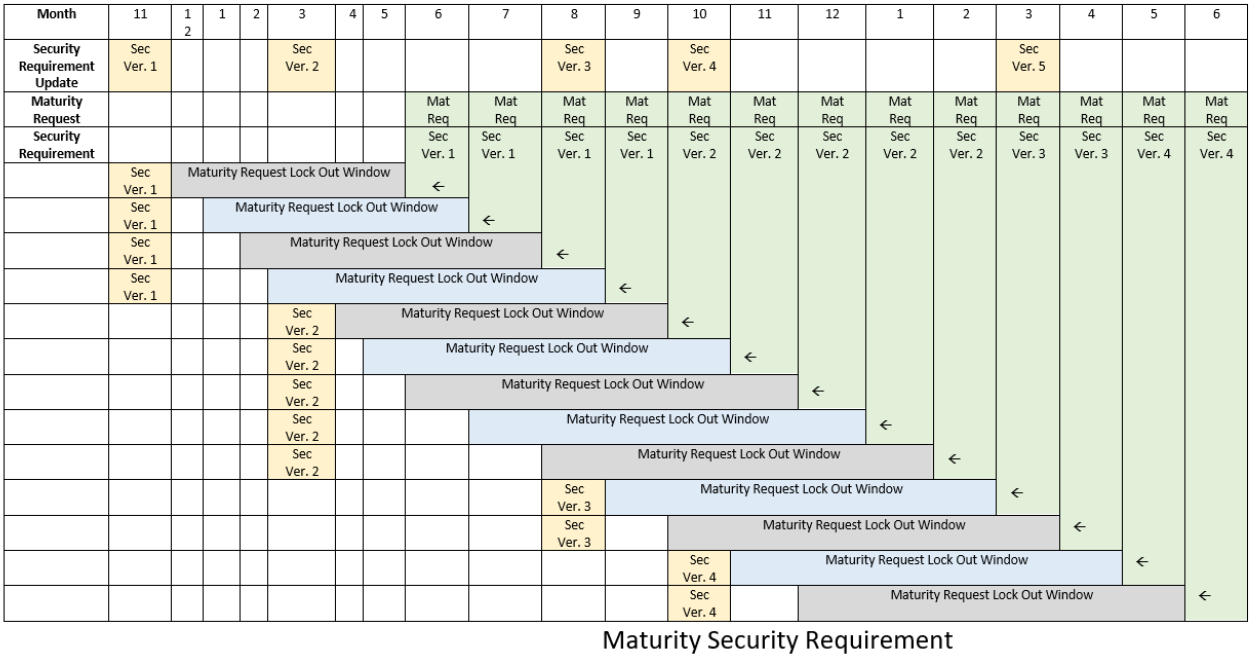
- Exception granted in cases of non-applicability.
- Exception granted in cases where another security mechanism specified in the blueprint and implemented mitigates the risk.
- Exceptions requested for cases above must be approved by the security sub-committee.
- Exceptions require a maximum of 21 days to review.
- The formal email date received, requesting a maturity review would be the Maturity Request date and this would define the set of security requirements that apply.
- Note: Critical vulnerabilities/security items, as categorized by the Akraino Security Sub-Committee, must be fixed even if found inside lock out window.

Current Maturity Requirements:

- Vuls PASS/FAIL Criteria, v1.0
- Lynis PASS/FAIL Criteria, v1.0
- Kube-Hunter PASS/FAIL Criteria, v1.0

Month	6/2020	7/2020	8/2020	9/2020	10/2020	11/2020	12/2020	1/2021
Maturity Request								
Security Requirement Update	v. 1.0							
Minimum Security Requirement		v. 1.0	v. 1.0	v. 1.0	v. 1.0	v. 1.0	v. 1.0	
Maximum Security Requirement		v. 1.0	v. 1.0	v. 1.0	v. 1.0	v. 1.0	v. 1.0	
		Release 4 Minimum Security Requirement Lock Out Window						

Example:



Vuls

Vuls will be integrated with Blueprint Validation Framework (Bluval User Guide)

Below are the list of tasks for integration.

Installation

Install Vuls containers (https://vuls.io/docs/en/install-with-docker.html). Vuls containers can be found at: https://hub.docker.com/u/vuls/

- Install go-cve-dictionary, run "docker pull vuls/go-cve-dictionary"
- Install goval-dictionary, run "docker pull vuls/goval-dictionary"
- Install gost, run "docker pull vuls/gost"
- Install vuls, run "docker pull vuls/vuls"

Set up and run

Detailed instruction can be found at https://vuls.io/docs/en/tutorial-docker.html

- Prepare log dir

  $ cd /path/to/working/dir

  $ mkdir go-cve-dictionary-log goval-dictionary-log gost-log

- Fetch NVD

  $ for i in `seq 2002 $(date +"%Y")`; do \ docker run --rm -it \ -v $PWD:/vuls \ -v $PWD/go-cve-dictionary-log:/var/log/vuls \ vuls/go-cve-dictionary fetchnvd -years $i; \ done

- Fetch OVAL
  - if you are using redhat/fedora

  $ docker run --rm -it \ -v $PWD:/vuls \ -v $PWD/goval-dictionary-log:/var/log/vuls \ vuls/goval-dictionary fetch-redhat 5 6 7 8

  if you are using ubuntu/debian

  $ docker run --rm -it \ -v $PWD:/vuls \ -v $PWD/goval-dictionary-log:/var/log/vuls \ vuls/goval-dictionary fetch-ubuntu 16 17 18 19

- Fetch gost

  $ docker run --rm -i \ -v $PWD:/vuls \ -v $PWD/goval-log:/var/log/gost \ vuls/gost fetch redhat

  Or

  $ docker run --rm -i \ -v $PWD:/vuls \ -v $PWD/goval-log:/var/log/gost \ vuls/gost fetch ubuntu

- Config the SUT, configuration will be stored in config.toml
  - SSH key generation & distribution: As Vuls connects to target server through SSH, and Vuls has to use SSH key-based authentication. There needs to be a way to generate SSH key pair, save the private key for Vuls container and dispatch the public key to target server. We probably don't want to store the private key with the container image if the container image is public accessible.

    [servers]

    [servers.c74]

    host = "54.249.93.16"

    port = "22"

    user = "vuls-user"

    keyPath = "/root/.ssh/id_rsa" # path to ssh private key in docker

- Start Vuls container to run tests
  - $ docker run --rm -it \ -v ~/.ssh:/root/.ssh:ro \ -v $PWD:/vuls \ -v $PWD/vuls-log:/var/log/vuls \ -v /etc/localtime:/etc/localtime:ro \ -e "TZ=Asia/Tokyo" \ vuls/vuls scan \ -config=./config.toml
- To get the report:
  - $ docker run --rm -it -v ~/.ssh:/root/.ssh:ro -v $PWD:/vuls -v $PWD/vuls-log:/var/log/vuls -v /etc/localtime:/etc/localtime:ro -e "TZ=Asia /Tokyo" vuls/vuls report -config=./config.toml
- Write Bluval configuration file for security tests
- Push test results to LF Nexus
  - Todo: How to tell test success or fail
  - Todo: Sample Test result
- Show test results in Bluval UI

Vuls Incubation and Maturity: PASS/FAIL Criteria, v1.0

All Critical vulnerabilities detected by Vuls must be patched/fixed. Critical vulnerabilities are defined as a CVSS score of 9.0-10.0. After patches/fixes are applied, Vuls must be run again to verify that the vulnerability is no longer detected.

The vuls.log output file and exception requests for any vulnerabilities that cannot be fixed must be sent to the security sub-committee.

Lynis

Lynis requires to run on SUT (System Under Test). The overall test framework will the similar to that of Vuls. As to the Lynis installation, there are two options:

1. Lynis is pre-installed on SUT by project team.
2. Lynis is to be installed as part of test flow from Validation Framework.

Considering the complexity of installing application on target system, it is recommended that option 1 is to be used.

For more information about Lynis, please check the link below:

https://cisofy.com/documentation/lynis/get-started/

You can download Lynis from cisofy.com. It is just over 1,000 lines of shell code.

You can use this version to check the configuration of a single server, either local or remote, as well as the configuration of a single docker file.

Just do ./lynis audit system

The output of the scan will be save in /var/log with the file name lynis-report.dat.

Then please upload these files in above folder and the report in txt or log format for the report.

Install and Execute

If you are using CentOS:

$ yum install lynis; lynis audit system

If you are using Ubuntu:

```
$ git clone https://github.com/CISOfy/lynis

$ cd lynis; ./lynis audit system
```

Report

After running, detailed test logs are stored in /var/log/lynis.log, information for each test includes:

- Time of an action/event
- Reason(s) why a test failed or was skipped
- Output of (internal) tests
- Suggestions about configuration options or how to fix/improve things
- Threat/impact score

In addition to log file, Lynis also creates a report and stores it in /var/log/lynis-report.dat. The report file contains the following information:

- Remarks = #<remark>
- Section = [<section name>]
- Option/value = <option name>=<value of option>

Lynis Incubation: PASS/FAIL Criteria, v1.0

1. The Lynis Program Update test MUST pass with no errors.
2. The following list of tests MUST complete as passing as described below.

   In the lynis.log outputfile each test suite has one or more individual tests. The beginning and ending of a test suite is marked with "====". For example, the 'ID BOOT-5122' test suite should display:

   ```
   020-04-08 15:36:28 ====
   2020-04-08 15:36:28 Performing test ID BOOT-5122 (Check for GRUB boot password)
   ...

   2020-04-08 15:36:29 Hardening: assigned maximum number of hardening points for this item (3).
   2020-04-08 15:36:29 ===
   ```

   If any tests in the test suit failed, there would be the following:

   ```
   2020-04-08 15:36:29 Suggestion: <Description of failed test>
   ```

   Also, the 'Hardening' line show above would not say 'assigned maximum number of hardening points', instead it would say 'assigned partial number of hardening points'.

1	Test: Checking PASS_MAX_DAYS option in /etc/login.defs
2	Performing test ID AUTH-9328 (Default umask values)
3	Performing test ID SSH-7440 (Check OpenSSH option: AllowUsers and AllowGroups)
4	Test: checking for file /etc/network/if-up.d/ntpdate
5	Performing test ID KRNL-6000 (Check sysctl key pairs in scan profile) : Following sub-tests required
5a	sysctl key fs.suid_dumpable contains equal expected and current value (0)
5b	sysctl key kernel.dmesg_restrict contains equal expected and current value (1)
5c	sysctl key net.ipv4.conf.default.accept_source_route contains equal expected and current value (0)
6	Test: Check if one or more compilers can be found on the system

The lynis.log output file and exception requests for any of the items listed above that cannot be fixed must be sent to the security sub-committee.

Lynis Maturity: PASS/FAIL Criteria, v1.0

1. The Lynis Program Update test MUST pass with no errors.
2. The following list of tests MUST complete as passing as described below.

In the lynis.log outputfile each test suite has one or more individual tests. The beginning and ending of a test suite is marked with "====". For example, the 'ID BOOT-5122' test suite should display:

```
020-04-08 15:36:28 ====
2020-04-08 15:36:28 Performing test ID BOOT-5122 (Check for GRUB boot password)
...

2020-04-08 15:36:29 Hardening: assigned maximum number of hardening points for this item (3).
2020-04-08 15:36:29 ===
```

If any tests in the test suit failed, there would be the following:

```
2020-04-08 15:36:29 Suggestion: <Description of failed test>
```

Also, the 'Hardening' line show above would not say 'assigned maximum number of hardening points', instead it would say 'assigned partial number of hardening points'.

1	Performing test ID BOOT-5122 (Check for GRUB boot password)
2	Performing test ID BOOT-5184 (Check permissions for boot files/scripts)
3	Test: Checking presence /var/run/reboot-required.pkgs
4	Performing test ID AUTH-9228 (Check password file consistency with pwck)
5	Performing test ID AUTH-9229 (Check password hashing methods)
6	Test: Checking SHA_CRYPT_MIN_ROUNDS option in /etc/login.defs
7	Test: Checking PASS_MAX_DAYS option in /etc/login.defs
8	Test: collecting accounts which have an expired password (last day changed + maximum change time)
9	Performing test ID AUTH-9328 (Default umask values)
10	Performing test ID FILE-6368 (Checking ACL support on root file system)
11	Performing test ID USB-2000 (Check USB authorizations)
12	Performing test ID USB-3000 (Check for presence of USBGuard)
13	Performing test ID PKGS-7370 (Checking for debsums utility)
14	Performing test ID PKGS-7388 (Check security repository in apt sources.list file)
15	Performing test ID SSH-7408 (Check SSH specific defined options)
16	Test: Checking AllowTcpForwarding in /tmp/lynis.ZotHQ7RQAj
17	Test: Checking ClientAliveCountMax in /tmp/lynis.ZotHQ7RQAj
18	Test: Checking ClientAliveInterval in /tmp/lynis.ZotHQ7RQAj
19	Test: Checking FingerprintHash in /tmp/lynis.ZotHQ7RQAj
20	Test: Checking IgnoreRhosts in /tmp/lynis.ZotHQ7RQAj
21	Test: Checking MaxAuthTries in /tmp/lynis.ZotHQ7RQAj
22	Test: Checking MaxSessions in /tmp/lynis.ZotHQ7RQAj
23	Test: Checking Port in /tmp/lynis.ZotHQ7RQAj
24	Test: Checking StrictModes in /tmp/lynis.ZotHQ7RQAj
25	Test: Checking TCPKeepAlive in /tmp/lynis.ZotHQ7RQAj
26	Performing test ID SSH-7440 (Check OpenSSH option: AllowUsers and AllowGroups)
27	Test: checking for file /etc/network/if-up.d/ntpdate
28	Performing test ID KRNL-6000 (Check sysctl key pairs in scan profile)
29	Test: Check if one or more compilers can be found on the system

The lynis.log output file and exception requests for any of the items listed above that cannot be fixed must be sent to the security sub-committee.

Kuber-Hunter

Kube-Hunter Incubation and Maturity: PASS/FAIL Criteria, v1.0

The kube-hunter vulnerabilities listed as 'Yes' in the 'Critical' column MUST be resolved.

	Kube-Hunter Vulnerability	CVE	Critical	Remediation
1	KHV002 - Kubernetes version disclosure		Yes	Disable --enable-debugging-handlers kubelet flag.
2	KHV003 - Azure Metadata Exposure		No - Azure Only	
3	KHV004 - Azure SPN Exposure		No - Azure Only	
4	KHV005 - Access to Kubernetes API		Yes	
5	KHV006 - Insecure (HTTP) access to Kubernetes API		Yes	Ensure your setup is exposing kube-api only on an HTTPS port.
				Do not enable kube-api's --insecure-port flag in production.
6	KHV007 - Specific Access to Kubernetes API		Yes	Review the RBAC permissions to Kubernetes API server for the anonymous and default service account.
				Depending on the Kubernetes cluster setup and preferences this may not be a problem.
7	KHV020 - Possible Arp Spoof		Yes	Consider dropping the NET_RAW capability from your pods using Pod.spec.securityContext.capabilities
8	KHV021 - Certificate Includes Email Address		Yes	Do not include email address in the Kubernetes API server certificate. (You should continue to use certificates to secure the API Server!)
9	KHV022 - Critical Privilege Escalation CVE	CVE-2018-1002105	Yes	Kubernetes v1.0.x-1.9.x – no fix available
				Kubernetes v1.10.0-1.10.10 – fixed inv1.10.11
				Kubernetes v1.11.0-1.11.4 – fixed inv1.11.5
				Kubernetes v1.12.0-1.12.2 – fixed inv1.12.3
10	KHV023 - Denial of Service to Kubernetes API Server	CVE-2019-1002100	Yes	Upgrade your kube-apiserver to newer versions, namely v1.11.8, v1.12.6, or v1.13.4
				If you cannot upgrade, or until you do, the best course of action is to remove patch permissions from untrusted users, or generally from admins who don't really use it.
12	KHV024 - Possible Ping Flood Attack	CVE-2019-9512	Yes	Disable HTTP/2 support OR obtain a software patch if available
13		CVE-2019-9514	Yes	Disable HTTP/2 support OR obtain a software patch if available
14	KHV026 - Arbitrary Access To Cluster Scoped Resources	CVE-2019-11247	Yes	Vulnerable versions: Fixed versions:
				Kubernetes 1.7.x-1.12.x Fixed in v1.13.9 by #80852
				Kubernetes 1.13.0-1.13.8 Fixed in v1.14.5 by #80851
				Kubernetes 1.14.0-1.14.4 Fixed in v1.15.2 by #80850
				Kubernetes 1.15.0-1.15.1 Fixed in master by #80750
15	KHV027 - Kubectl Vulnerable To CVE-2019-11246	CVE-2019-11246	Yes	Update your kubectl client to one of the following versions: 1.12.9, 1.13.6, 1.14.2

16	KHV028 - Kubectl Vulnerable To CVE-2019-1002101	CVE-2019-1002101	Yes	The issue is resolved in kubectl v1.11.9, v1.12.7, v1.13.5, and v1.14.0.
17	KHV029 - Dashboard Exposed		Yes	Do not leave the Dashboard insecured.
18	KHV030 - Possible DNS Spoof		Yes	Consider using DNS over TLS. CoreDNS (the common DNS server for Kubernetes) supports this out of the box, but your client applications might not.
19	KHV031 - Etcd Remote Write Access Event		Yes	Ensure your etcd is accepting connections only from the Kubernetes API, using the --trusted-ca-file etcd flag. This is usually done by the installer, or cloud platform.
20	KHV032 - Etcd Remote Read Access Event		Yes	Ensure your etcd is accepting connections only from the Kubernetes API, using the --trusted-ca-file etcd flag. This is usually done by the installer, or cloud platform.
21	KHV033 - Etcd Remote version disclosure		Yes	
22	KHV034 - Etcd is accessible using insecure connection (HTTP)		Yes	Ensure your setup is exposing etcd only on an HTTPS port by using the etcd flags --key-file and --cert-file.
23	KHV036 - Anonymous Authentication		Yes	Ensure kubelet is protected using --anonymous-auth=false kubelet flag. Allow only legitimate users using --client-ca-file or --authentication-token-webhook kubelet flags. This is usually done by the installer or cloud provider.
24	KHV037 - Exposed Container Logs		Yes	Disable --enable-debugging-handlers kubelet flag.
25	KHV038 - Exposed Running Pods		Yes	Disable --enable-debugging-handlers kubelet flag.
26	KHV039 - Exposed Exec On Container		Yes	Disable --enable-debugging-handlers kubelet flag.
27	KHV040 - Exposed Run Inside Container		Yes	Disable --enable-debugging-handlers kubelet flag.
28	KHV041 - Exposed Port Forward		Yes	Disable --enable-debugging-handlers kubelet flag.
29	KHV042 - Exposed Attaching To Container		Yes	Disable --enable-debugging-handlers kubelet flag.
30	KHV043 - Cluster Health Disclosure		Yes	Disable --enable-debugging-handlers kubelet flag.
31	KHV044 - Privileged Container		Yes	Minimize the use of privileged containers. Use Pod Security Policies to enforce using privileged: false policy.
32	KHV045 - Exposed System Logs		Yes	Disable --enable-debugging-handlers kubelet flag.
33	KHV046 - Exposed Kubelet Cmdline		Yes	Disable --enable-debugging-handlers kubelet flag.
34	KHV047 - Pod With Mount To /var/log		Yes	Consider disallowing running as root: Using Kubernetes Pod Security Policies with MustRunAsNonRoot policy.
				Consider disallowing writable host mounts to /var/log: Using Kubernetes Pod Security Policies with AllowedHostPaths policy.
35	KHV049 - kubectl proxy Exposed		Ye	Expose your applications in a permanent, legitimate way, such as via Ingress.
				Close open proxies immediately after use.
36	KHV050 - Read access to Pod service account token		Yes	It is recommended to explicitly specify a Service Account for all of your workloads (serviceAccountName in Pod. Spec), and manage their permissions according to the least privilege principle.
				Consider opting out automatic mounting of SA token using automountServiceAccountToken: false on ServiceAccount resource or Pod.spec.
37	Access to pod's secrets		Yes	https://blog.aquasec.com/managing-kubernetes-secrets
				Securing etcd—secret data is stored in etcd. By default, etcd data is not encrypted and neither are your secrets. You should enable encryption at rest, limit access to etcd to admin users only, and safely dispose of disks where etcd data was formerly stored
				Use SSL/TLS—when running etcd in a cluster, you must use secure peer-to-peer communication.
38	CAP_NET_RAW Enabled		Yes - If applicable	CAP_NET_RAW is used to open a raw socket and is used by ping. If this is not required CAP_NET_RAW MUST be removed.
				https://www.suse.com/c/demystifying-containers-part-iv-container-security/

The cluster.log and pod.log output files and exception requests for any of the items listed above that cannot be fixed must be sent to the security sub-committee.

Security Scan Additional Information and Tips

How To Create Security Logs

Copy their log directory to $WORKSPACE/archives in their local server, and then use this command to upload files.

*lftools deploy archives -p '**/*.log' $NEXUS_URL $NEXUS_PATH $WORKSPACE*

Below are commands to zip the results into a folder and push this results.zip file using lftools:

zip -r results.zip <results_folder_path>

lftools deploy nexus-zip $NEXUS_URL logs $NEXUS_PATH results.zip

Jira tickets tracking integration with Bluval:

https://jira.akraino.org/secure/RapidBoard.jspa?rapidView=5&projectKey=VAL&view=detail&selectedIssue=VAL-79

https://jira.akraino.org/secure/RapidBoard.jspa?rapidView=5&projectKey=VAL&view=detail&selectedIssue=VAL-80

Sample results sheet comprising of combined results and vulnerabilities of all security tools (EALT-EDGE Blueprint):