Robot basic architecture based on SSES Blueprint

Test document

V1.4 12/12/2022

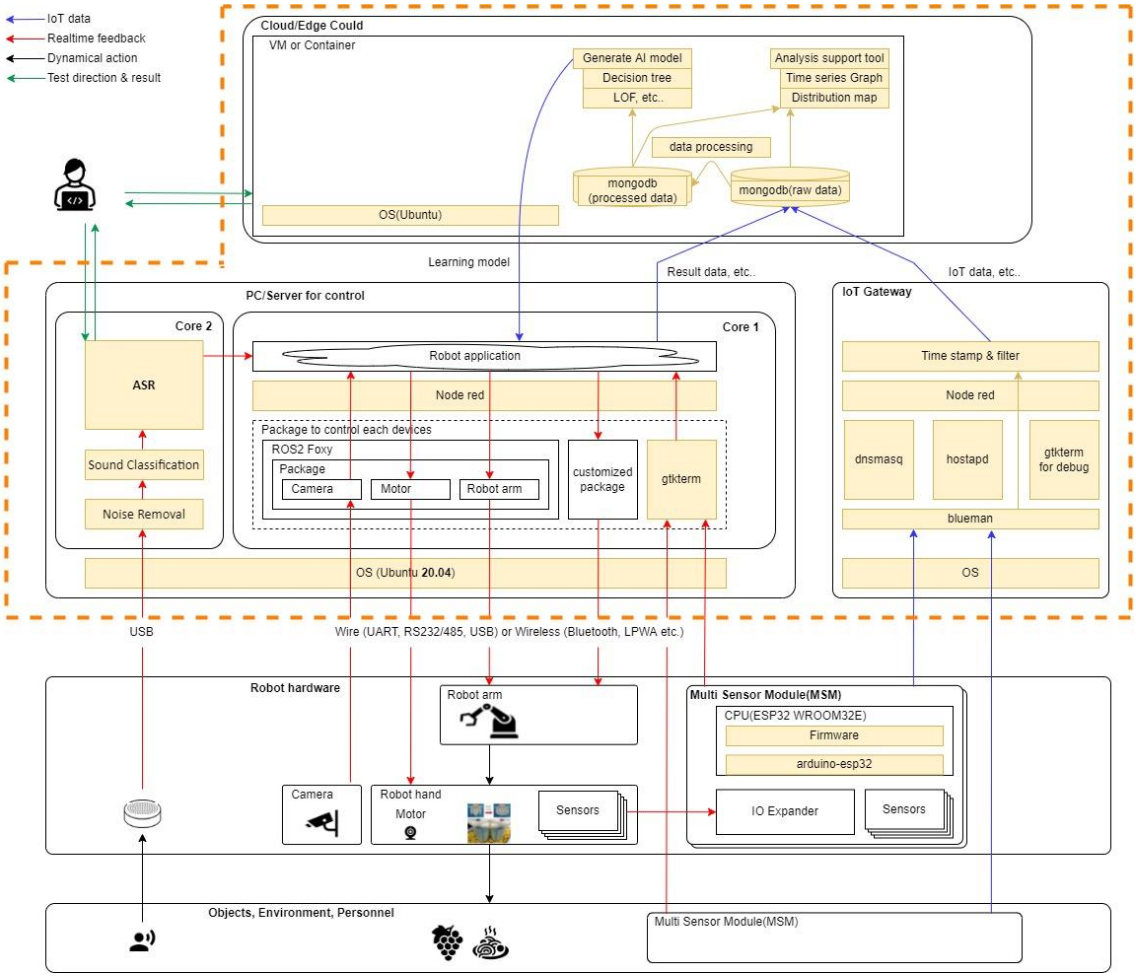
Table of contents

1 Introduction

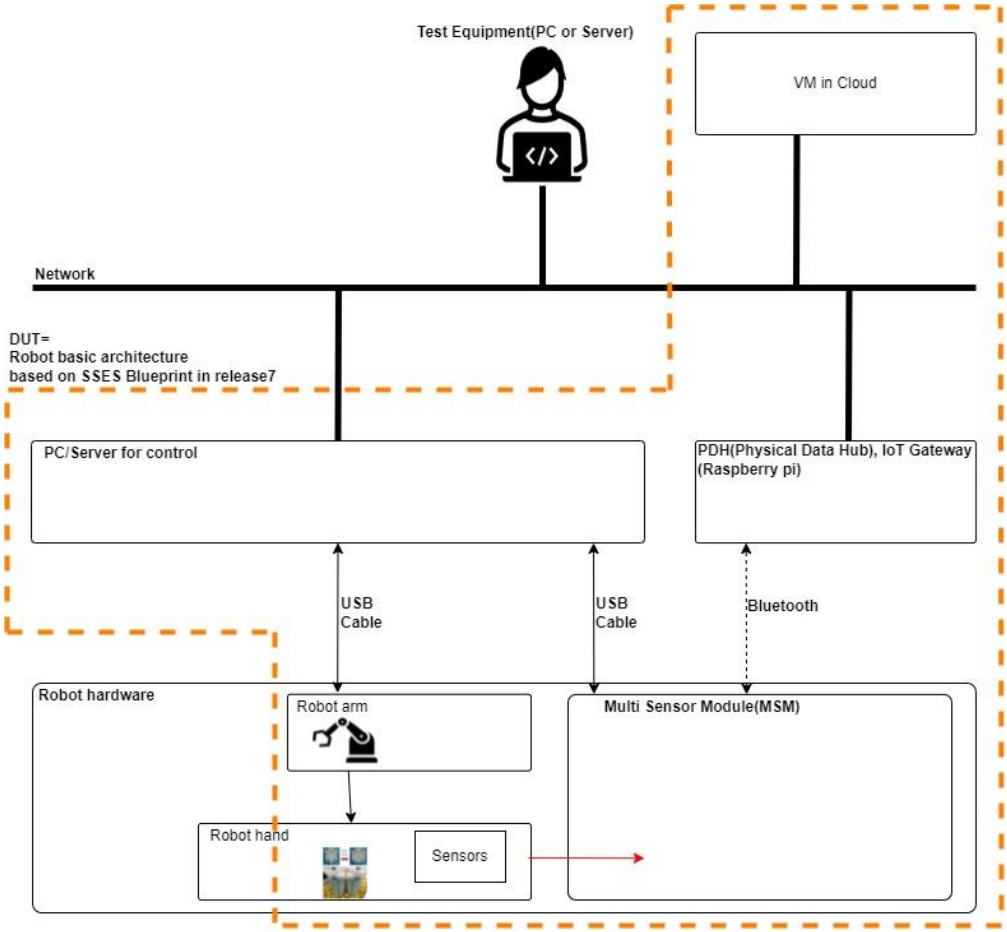
This document covers Test Deployment Environment and Test Case for Robot basic architecture based on SSES Blueprint. The scopes of test are installation SW to HW for robot application and connectivity between each SW and HW.

2 Overall Test Architecture

The following figure indicates overall test architecture, DUT(Device under test), and TE(Test Equipment). We will build these test bed in Ritsumeikan university.

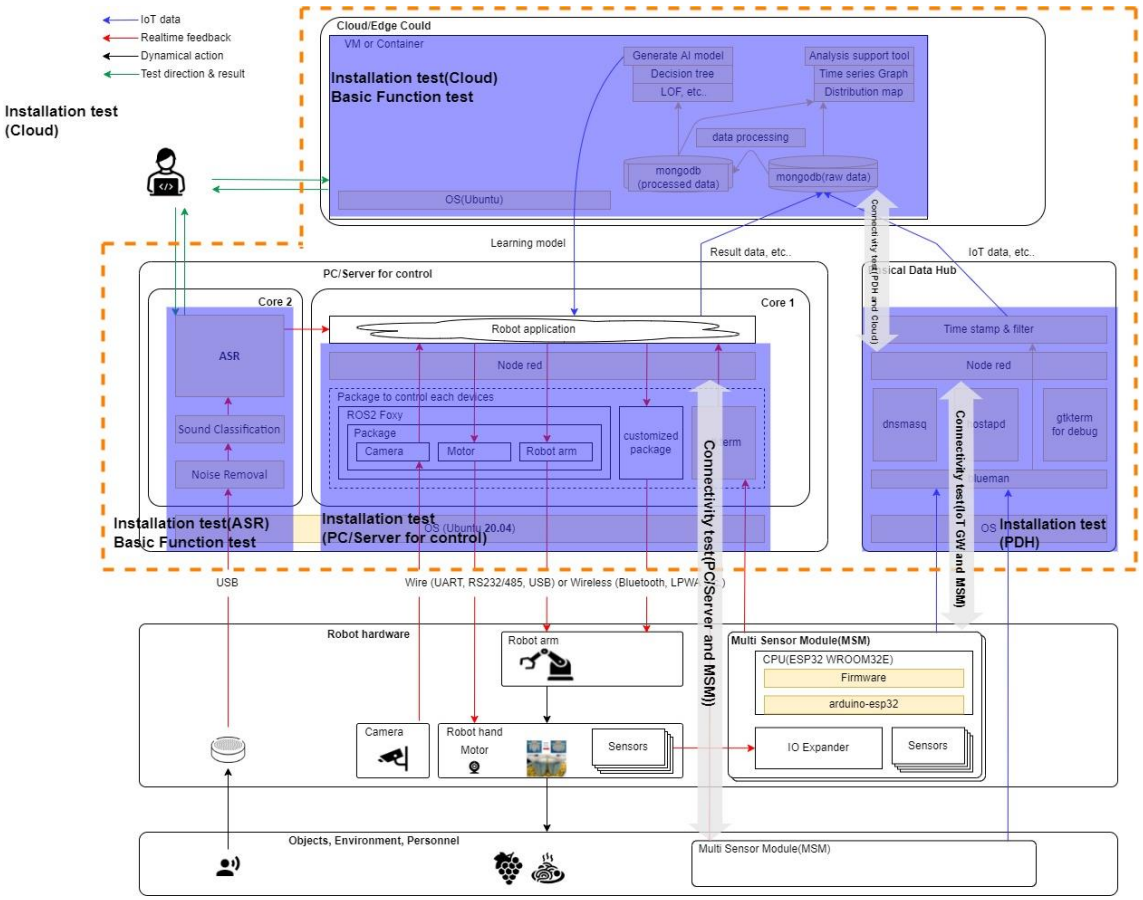

IoT data
Realtime feedback
Dynamical action
Test direction & result
Cloud/Edge Could
VM or Container
Generate AI model
Decision tree
LOF, etc..
Analysis support tool
Time series Graph
Distribution map
data processing
mongodb (processed data)
mongodb(raw data)
OS(Ubuntu)
Learning model
Result data, etc..
IoT data, etc..
PC/Server for control
Core 2
Core 1
ASR
Sound Classification
Noise Removal
Robot application
Node red
Package to control each devices
ROS2 Foxy
Package
Camera
Motor
Robot arm
customized package
gtkterm
OS (Ubuntu 20.04)
IoT Gateway
Time stamp & filter
Node red
dnsmasq
hostapd
gtkterm for debug
blueman
OS
USB
Wire (UART, RS232/485, USB) or Wireless (Bluetooth, LPWA etc.)
Robot hardware
Robot arm
Camera
Robot hand
Motor
Sensors
Multi Sensor Module(MSM)
CPU(ESP32 WROOM32E)
Firmware
arduino-esp32
IO Expander
Sensors
Objects, Environment, Personnel
Multi Sensor Module(MSM)

The following figure indicates HW and its connection. All machines except the cloud are on the same local area network.



3 Test API description

The following figure coverage of this test.



Bare Metal Deployment

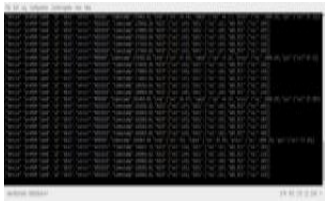
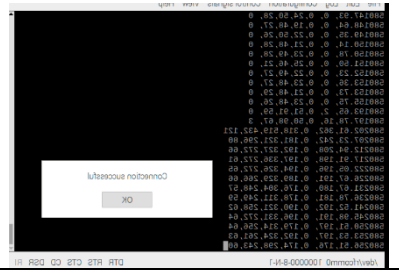
No	Test Case	Test input	Test Procedure	Expected output	Test result
1	PDH,IoT gateway	-	-	-	-
1-1	system configuration	-	For more details, refer to installation guide. 1. Log in to IoT gateway raspberry pi 2. Display setting vi /boot/config.txt hdmi_force_hotplug=1 3. Set to run Node-RED when PowerON sudo systemctl enable nodered.service 4. Allow VNC and SSH Connect Test Equipment to IoT gateway via USB. Run terminal. 5. Set time server		OK
1-2	Install GTKTerm	-	sudo apt install gtkterm $ which gtkterm	/usr/bin/gtkterm	OK

1-3	Install hostapd	-	sudo apt install hostapd hostapd -v	hostapd v2.X User space daemon for IEEE 802.11 AP management, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator Copyright I 2002-2019, Jouni Malinen <j@w1.fi> and contributors	OK
1-4	Install dnsmasq	-	sudo apt install dnsmasq dnsmasq -v	Dnsmasq version 2.85 Copyright I 2000-2021 Simon Kelley Compile time options: Ipv6 GNU-getopt Dbus no-Ubus i18n IDN2 DHCP DHCPv6 no-Lua TFTP conntrack ipset auth cryptohash DNSSEC loop-detect inotify dumpfile This software comes with ABSOLUTELY NO WARRANTY. Dnsmasq is free software, and you are welcome to redistribute it under the terms of the GNU General Public License, version 2 or 3.	OK
1-5	Install blueman	-	sudo apt install blueman $ sudo find / -name blueman	/var/lib/blueman	$ sudo find / -name blueman /var/lib/blueman
2	PC/Server for control	-	-		
2-1	Install GTKterm	-	Refer to installation guide.		OK
2-2	Install python	-	Refer to installation guide.		OK
2-3	Install Node-RED	-	Refer to installation guide.		OK
2-4	Install ASR		Refer to installation guide.		OK
3	Cloud/Edge Cloud				

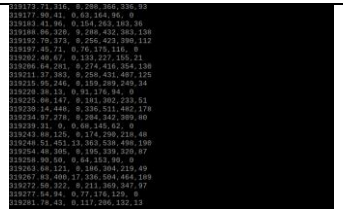
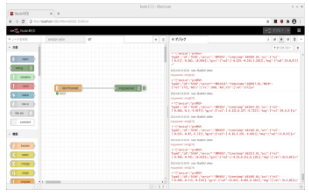
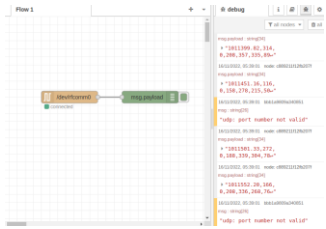
3-1	Install MongoDB		Refer to installation guide.		OK

Connectivity test

No	Test Case	Test input	Test Procedure	Expected output	Test result

1	MSM to PDH,IoT Gateway	-	For more details, refer to installation guide.
			The following commands are executed from IoT gateway terminal.
			1. bluetoothctl
			2. power on
			3. scan on
			You can detect MSM and its address.
			4. scan off
			5. exit
			6. sudo rfcomm bind <serial port#> <address>
			7. rfcomm show 0
			8. ls -l /dev/rfcom*
			9. gtkterm -p </dev/rfcomm#> -s 1000000
				The gtkterm shows the following message.
				 
					After step 6 is described.
					$ sudo rfcomm bind 0 44:17:93:60:4C:9E
					
					$ rfcomm show 0
					rfcomm0: 44:17:93:60:4C:9E channel 1 clean
					
					$ ls -l /dev/rfcom*
					crw-rw---- 1 root dialout 216, 0 Nov 15 01:06 /dev/rfcomm0
					
					$ gtkterm -p /dev/rfcomm0 -s 1000000
					 

2	MSM to Cloud via PDH,IoT Gateway (Node-RED)	-	1.Set Node-RED flow to PDH. Refer to installation guide. 2.Install 3.Confirm whether mongodb receives data from MSM or not by the following	Return documents from mongodb.	> use AGV > db.motion.find() { _id: ObjectId(“637c55be29c01301cdaf3786”),

			command from Test Equipment. Mongosh <URL> --username <username> use <database name> db.AGV.find()		device: 'R-MSM', id: '4C9E', sensor: 'BMX160', timestamp: 730903.76, acc: { val: [4.107, 8.038, 5.298], units: 'm/s2' }, gyro: { val: [-0.03, -0.152, 0.122], units: 'deg/s' }, mag: { val: [48, 209, -14], units: 'nT' }, datetime: '2022-11-22T13:53:18.467+09:00' } …
3	MSM to PC/Server for control	-	Execute the following commands in PC/Server for control. Gtkterm -p <port name> -s 1000000 *The port name is port which is connected to MSM via USB cable.		
4	MSM to PC/Server for control (Node-RED)		Create Node-RED flow in PC/Server for control. 1. Run Node-RED in PC/Server for control 2. Add "serial in" with baud rate=1Mbps and port which connected to MSM. 3. Add "debug". 4. Connect the "serial in" and the "debug". 5. Deploy	The Node-RED shows the following message. 	

Basic function test

No	Test Case	Test input	Test Procedure	Expected output	Test result

1	Analysis support tool	-	1.Connect to VM which “analysis support tool” is installed.
			2.Change directory to folder which “analysis support tool” is installed.
			2.Execute python3.8
			3.Execute the following command
			import msm_data_process
			datetimes,timestamps,acc_x,acc_y,acc_z,gyro_x,gyro_y,gyro_z,mag_x,mag_y,mag_z = msm_data_process.read_raw_db_motion(<dev_id>,<start time>,<end time>)
			import common
			common.plt_fig(datetimes,gyro_z,”test”)
			
			<dev_id> is your MSM id.
			E.g.
			<start time> = “2022-11-07T12:00:00+09:00”
			<end time> = “2022-11-07T13:00:00+09:00”
				Time series data graph of gyro_z from MSM will be generated.	**2.Change directory to folder which “analysis support tool” is installed.**
					$ sudo apt install -y python3-pip
					$ pip3 install matplotlib pandas seaborn pymongo
					
					3.Execute the following command
					>>> import msm_data_process
					>>>
					datetimes,timestamps,acc_x,acc_y,acc_z,gyro_x,gyro_y,gyro_z,mag_x,mag_y,mag_
					plt.figure(figsize=(60,15))
					plt.plot(x,y)
					plt.xticks(rotation=45)
					plt.minorticks_on()
					plt.grid(which=”major”, color=”gray”, linestyle=”solid”)
					plt.grid(which=”minor”, color=”lightgray”, linestyle=”dotted”)
					plt.gca().xaxis.set_major_locator(mdates.MinuteLocator(byminute=None,
					intervz = msm_data_process.read_raw_db_motion(“B3DA”,”2022-11-07T12:00:00+09:00”,”2022-11-07T13:00:00+09:00”)
					>>> import common
					>>> common.plt_fig(datetimes,gyro_z,”test”)
					
					$ ls
					test.png

2	ASR		$ ./mediaMin -M0 -cx86 -I ../pcaps/asr_test1.pcap -L -d0x10000c19 -r20 … 00:00:09.922.692 INFO: DSDeleteSession() removed term1 stream 0 from group “asr_test1”, session = 0 A KING ROLLED THE STAKE IN THE EARLY DAYS WE FOUND WHEN EVENTS TAKE A BAD TURN … # Confirm that the words “A KING ROLLED THE STAKE IN THE EARLY DAYS WE FOUND WHEN EVENTS TAKE A BAD TURN”.		$ pwd /home/ubuntu/Signalogic_2020v8/DirectCore/apps/SigC641x_C667x/mediaTest/mediaMin $ ./mediaMin -M0 -cx86 -I ../pcaps/asr_test1.pcap -L -d0x10000c19 -r20 … 00:00:09.922.692 INFO: DSDeleteSession() removed term1 stream 0 from group “asr_test1”, session = 0 A KING ROLLED THE STAKE IN THE EARLY DAYS WE FOUND

					WHEN EVENTS TAKE A BAD TURN …

Blueval test

No	Test Case	Test input	Test Procedure	Expected output	Test result
1	PDH,IoT gateway Layer:OS	-	1.Create directory mkdir ~/vuls cd ~/vuls mkdir go-cve-dictionary-log goval-dictionary-log gost-log 2.Fetch NVD docker run –rm -it ¥ -v $PWD:/go-cve-dictionary ¥ -v $PWD/go-cve-dictionary-log:/var/log/go-cve-dictionary ¥ vuls/go-cve-dictionary fetch nvd 3.Fetch OVAL docker run –rm -it ¥ -v $PWD:/goval-dictionary ¥ -v $PWD/goval-dictionary-log:/var/log/goval-dictionary ¥ vuls/goval-dictionary fetch □eypat 11 4.Fetch gost docker run –rm -I ¥ -v $PWD:/gost ¥ -v $PWD/gost-log:/var/log/gost ¥ vuls/gost fetch □eypat 6. Create config.toml [servers] [servers.master]	-	OK

			host = “<IP Address>” port = “22” user = “<user name>” eypath = “/root/.ssh/id_rsa” # path to ssh private key in docker 7. 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 ¥ -v /etc/timezone:/etc/timezone:ro ¥ vuls/vuls scan ¥ -config=./config.toml 8. 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 ¥ vuls/vuls report ¥ -format-list ¥ -config=./config.toml ■lynis git clone https://github.com/CISOfy/lynis cd lynis; ./lynis audit system		
2	PC/Server for control Layer:OS	-	■vuls 1.Create directory mkdir ~/vuls	-	OK

			cd ~/vuls mkdir go-cve-dictionary-log goval-dictionary-log gost-log 2.Fetch NVD docker run –rm -it ¥ -v $PWD:/go-cve-dictionary ¥ -v $PWD/go-cve-dictionary-log:/var/log/go-cve-dictionary ¥ vuls/go-cve-dictionary fetch nvd 3.Fetch OVAL docker run –rm -it ¥ -v $PWD:/goval-dictionary ¥ -v $PWD/goval-dictionary-log:/var/log/goval-dictionary ¥ vuls/goval-dictionary fetch ubuntu 18 19 20 21 22 4.Fetch gost docker run –rm -I ¥ -v $PWD:/gost ¥ -v $PWD/gost-log:/var/log/gost ¥ vuls/gost fetch ubuntu 9. Create config.toml [servers] [servers.master] host = “<IP Address>” port = “22” user = “<user name>” eypath = “/root/.ssh/id_rsa” # path to ssh private key in docker 10. 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 ¥ -v /etc/timezone:/etc/timezone:ro ¥ vuls/vuls scan ¥ -config=./config.toml 11. 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 ¥ vuls/vuls report ¥ -format-list ¥ -config=./config.toml ■lynis git clone https://github.com/CISOfy/lynis cd lynis; ./lynis audit system		
3	Cloud/Edge Cloud Layer:OS		■vuls 1.Create directory mkdir ~/vuls cd ~/vuls mkdir go-cve-dictionary-log goval-dictionary-log gost-log 2.Fetch NVD docker run –rm -it ¥ -v $PWD:/go-cve-dictionary ¥		OK

			-v $PWD/go-cve-dictionary-log:/var/log/go-cve-dictionary ¥ vuls/go-cve-dictionary fetch nvd 3.Fetch OVAL docker run –rm -it ¥ -v $PWD:/goval-dictionary ¥ -v $PWD/goval-dictionary-log:/var/log/goval-dictionary ¥ vuls/goval-dictionary fetch ubuntu 18 19 20 21 22 4.Fetch gost docker run –rm -I ¥ -v $PWD:/gost ¥ -v $PWD/gost-log:/var/log/gost ¥ vuls/gost fetch ubuntu 12. Create config.toml [servers] [servers.master] host = “<IP Address>” port = “22” user = “<user name>” eypath = “/root/.ssh/id_rsa” # path to ssh private key in docker 13. 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 ¥ -v /etc/timezone:/etc/timezone:ro ¥		

			vuls/vuls scan ¥ -config=./config.toml 14. 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 ¥ vuls/vuls report ¥ -format-list ¥ -config=./config.toml ■lynis git clone https://github.com/CISOfy/lynis cd lynis; ./lynis audit system		

4 Revision history

Version	Date	Editor	Contents
0.1	02/07/2022	Fukano	Draft version
1.0	02/10/2022	Fukano	Review completed and published as first edition
1.1	03/04/2022	Inoue	Minor modifications to procedures
1.2	03/23/2022	Inoue	Write test result
1.3	11/07/2022	Fukano	Updated for Release7 ・Add cloud ・Add ASR
1.4	12/12/2022	Fukano	Added header and footer.