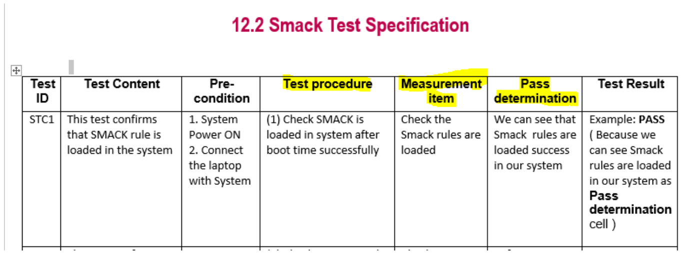
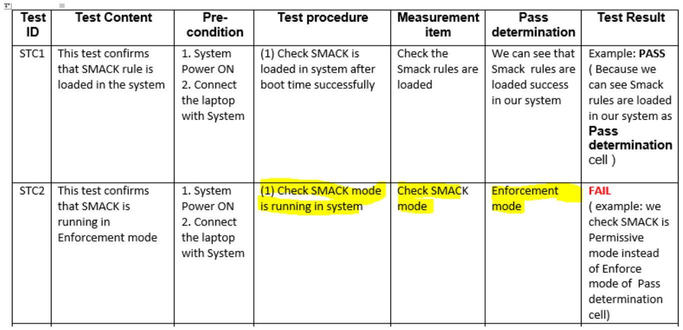
SCRUM WAY OF WORK

Backlogs

Scope	Feature	Tasks	Description	DoD	Estimation (Days)
Environment setup	Environment setup	Environment setup	Hardware setup WebOS installation		3
Development	Firewall (27days)	F1: Self Technical Knowledge Research	Research about Netfilter framework and relevant libraries Define network topology, details will be provided after checking the platform.	Complete network topology Complete to integration of relevant firewall libs Complete to integration of necessary kernel configs	2
		F2: Set up GitLab repo, and build environment	Created GitLab repo. Define location for firewall-manager service Added the firewall-manager files.		2
		F3: Neifilter kernel configuration	Check necessary kernel configs Integrate these configs into source code Perform build and fix issues Optimate boot time regarding these configs		1
		F4: Setup system environment which prepares firewall implementation	Prepare firewall libs and make sure that commands are supported. Build a webOS image to flash onto Raspberry Pi 4 with a firewall.		3
		F5: Implement basic firewall feature so that it can block traffic from IPs and Ports.	Gather whitelist from connections Create firewall whitelist rules	Complete collect firewall rules	3

		F6: Design Firewall default/custom chain	Design Firewall chain structure Make an automation tool to generate rules from gathered whitelist	Load the success configuration file at startup	3
		F7: Make and run Firewall service	Make Firewall config file and load it at bootup		2
		F8: Implement advanced firewall feature so that it can prevent attack-type	Firewall rules configuration to prevent DDoS attacks, SYNC/ICMP floods, etc.	Complete firewall rule for the type of attacks Complete firewall log and accounting Complete firewall test	2
		F9: Implement advanced firewall - Log, Protect attack	Implement firewall log and save drop logs		2
		F10: Firewall accounting integrate	Implement Accounting log to counter valid/invalid packets		1
		F11: Firewall - Integration with other features	Turn on enforcing mode and check for connection from other features		2
		F12: Integration test	Perform test one-by-one firewall configs to make sure it's working properly		1
	Smack (24days)	S1: Self Technical Knowledge Research	Research about SMACK feature and how it work.	Have Technical Knowledge regarding SMACK SMACK Overview Operation, components Necessary files and service Hardware and software preparation	3
		S2: - Set up GitLab repo, and build environment - Porting SMACK feature	Porting SMACK to webos platform Append smack feature to webos image and build it		1
		S3: Flashing to device and changing permission mode	Flashing the built image to the Pi 4 board Enable permissive mode in smack feature in the webos image		1

		S4: Analyze system logs in audit log	Check system logs in audit log and make smack rule table	Source code download and build Flash firmware Check SMACK integrate	1
		S5: Porting smack, systemd mount	Mount systemd to webos image containing SMACK feature		2
		S6: Investigate resources all modules in the project	Initial labeling for all system files	Complete master sheet for SMACK rules Make defined rules file for all file systems Make defined rules file for all features/services Apply these rules to target devices Coordinate with other modules to apply missing rules	4
		S7: Labeling and getting all rules	Make labeling config files Apply these config files Request other modules to test all missing rules.		4
		S8: Enable enforce mode, request other modules to test to make sure all system run as well.	Turn on enforcing mode and check for permissions from other features/Systems	Resolve issues when turn on enforcing mode Turn on enforcing mode Analyze audit log and resolve issues occur Support other features regarding permission	4
		S9: Integration with other features	Integration with other features (Firewall, EV, Music Player) Making sure that all components operate as required.		4
Verification/Validation	Security Features Test	**SFT_Task1**: Make detailed Test case for Test Scenario: STC1 in 12.2 Smack Test Specification.docx file.	Read carefully information in Test Scenario **STC1** in 12.2 Smack Test Specification.docx file. Based on information of these cells below: • Test procedure • Measurement item • Pass determination	Have a detailed Test case for Test Scenario **STC1** with all mandatory fields: Test Purpose Precondition Test steps	0.2

<table>
<tr>
<td></td>
<td></td>
<td></td>
<td>

<b>12.2 Smack Test Specification</b>
<table>
<tr><th>Test ID</th><th>Test Content</th><th>Pre-condition</th><th>Test procedure</th><th>Measurement item</th><th>Pass determination</th><th>Test Result</th></tr>
<tr><td>STC1</td><td>This test confirms that SMACK rule is loaded in the system</td><td>1. System Power ON
2. Connect the laptop with System</td><td>(1) Check SMACK is loaded in system after boot time successfully</td><td>Check the Smack rules are loaded</td><td>We can see that Smack rules are loaded success in our system</td><td>Example: <b>PASS</b> (Because we can see Smack rules are loaded in our system as <b>Pass determination</b> cell)</td></tr>
</table>
<p>You need to know:</p>
<p>Command to check SMACK rules as <b>Test procedure</b> cell
Know about How to check Smack rule is loaded success</p>
<p><b>Note</b>: You can read <u>12.1 Task and Test case Example.docx</u> file to understand about <b>Description, Define of Done</b> of task and <b>Detailed Test case.</b></p>
</td>
<td>Expected Result (Have specific expected result for each Test step)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><b>SFT_Task2</b>: Make detailed Test case for Test Scenario: STC2 in 12.2 Smack Test Specification.docx</td>
<td>
<p>Read carefully information in Test Scenario <b>STC2</b> in 12.2 Smack Test Specification.docx . Based on information of these cells below in 12.2 Smack Test Specification.docx:</p>
<ul>
<li>Test procedure</li>
<li>Measurement item</li>
<li>Pass determination</li>
</ul>
<p>You need to know command(s)/ write Script to run as <b>Test procedure</b> cell</p>
<p>Understand expected result of each command or script</p>

<table>
<tr><th>Test ID</th><th>Test Content</th><th>Pre-condition</th><th>Test procedure</th><th>Measurement item</th><th>Pass determination</th><th>Test Result</th></tr>
<tr><td>STC1</td><td>This test confirms that SMACK rule is loaded in the system</td><td>1. System Power ON
2. Connect the laptop with System</td><td>(1) Check SMACK is loaded in system after boot time successfully</td><td>Check the Smack rules are loaded</td><td>We can see that Smack rules are loaded success in our system</td><td>Example: <b>PASS</b> (Because we can see Smack rules are loaded in our system as <b>Pass determination</b> cell)</td></tr>
<tr><td>STC2</td><td>This test confirms that SMACK is running in Enforcement mode</td><td>1. System Power ON
2. Connect the laptop with System</td><td>(1) Check SMACK mode is running in system</td><td>Check SMACK mode</td><td>Enforcement mode</td><td>FAIL
(example: we check SMACK is Permissive mode instead of Enforce mode of Pass determination cell)</td></tr>
</table>
</td>
<td>
<p>Have a detailed Test case for Test Scenario <b>STC2</b> with all mandatory fields:</p>
<p>Test Purpose
Precondition
Test steps
Expected Result (Have specific expected result for each Test step)</p>
</td>
<td>0.2</td>
</tr>
</table>

		SFT_Task3: Make detailed Test case for Test Scenario: STC3 in 12.2 Smack Test Specification.docx	Read carefully information in Test Scenario **STC3** in 12.2 Smack Test Specification.docx . Based on information of these cells below in 12.2 Smack Test Specification.docx: Test procedure Measurement item Pass determination You need to know command(s)/ write Script to run Understand expected result of each command or script	Have a detailed Test case for Test Scenario **STC3** with all mandatory fields: Test Purpose Precondition Test steps Expected Result (Have specific expected result for each Test step)	0.6
		SFT_Task4: Make detailed Test case for Test Scenario: STC4 in 12.2 Smack Test Specification.docx	Read carefully information in Test Scenario **STC4** in 12.2 Smack Test Specification.docx . Based on information of these cells below in 12.2 Smack Test Specification.docx: Test procedure Measurement item Pass determination You need to know command(s)/ write Script to run Understand expected result of each command or script	Have a detailed Test case for Test Scenario **STC4** with all mandatory fields: Test Purpose Precondition Test steps Expected Result (Have specific expected result for each Test step)	1
		SFT_Task5: Make detailed Test case for Test Scenario: STC5 in 12.2 Smack Test Specification.docx	Read carefully information in Test Scenario **STC5** in 12.2 Smack Test Specification.docx. Based on information of these cells below in 12.2 Smack Test Specification.docx: Test procedure Measurement item Pass determination You need to know command(s)/ write Script to run Understand expected result of each command or script	Have a detailed Test case for Test Scenario **STC5** with all mandatory fields: Test Purpose Precondition Test steps Expected Result (Have specific expected result for each Test step)	1

		SFT_Task6: Make detailed Test case for Test Scenario: STC6 in 12.2 Smack Test Specification.docx	Read carefully information in Test Scenario **STC6 in 12.2 Smack Test Specification.docx** . Based on information of these cells below in 12.2 Smack Test Specification.docx: Test procedure Measurement item Pass determination You need to know command(s)/ write Script to run Understand expected result of each command or script	Have a detailed Test case for Test Scenario **STC6** with all mandatory fields: Test Purpose Precondition Test steps Expected Result (Have specific expected result for each Test step)	1
		SFT_Task7: Make detailed Test case for Test Scenario: STC7 in 12.2 Smack Test Specification.docx	Read carefully information in Test Scenario **STC7** in 12.2 Smack Test Specification.docx . Based on information of these cells below in 12.2 Smack Test Specification.docx: Test procedure Measurement item Pass determination You need to know command(s)/ write Script to run Understand expected result of each command or script	Have a detailed Test case for Test Scenario **STC7** with all mandatory fields: Test Purpose Precondition Test steps Expected Result (Have specific expected result for each Test step)	1
		SFT_Task8: Make more Test cases for Smack out of list 7 Test Scenarios in 12.2 Smack Test Specification.docx	Based on your understood about Smack , Smack rule, Smack label, You design Smack Test case by your self.	Have a detailed Test case with all mandatory fields: Test Purpose Precondition Test steps Expected Result (Have specific expected result for each Test step)	2
		SFT_Task9: Research about Scapy and create Ethernet packet by Scapy	Understand about Scapy Install success Scapy in Lapop	Run Scapy in your laptop Have document with specific scapy command for	2

			Create Ethernet package by Scapy which can set Ethernet type, IP, port, protocol by Scapy	Ethernet package with Ethernet type, IP, port, protocol	
		SFT_Task10: Setup Test environment for firewall	You need to: Config to connect Ethernet package from Laptop to Raspberry Pi (ping success) Send Ethernet packet success from laptop to Raspberry Pi and we can capture that packet in Raspberry Pi and can see it on Wireshark tool	Ping from Laptop to Raspberry Pi success Send Ethernet packet by Scapy from your laptop success We can see your packet in Raspberry Pi on Wire Shark tool after capture packet in Raspberry Pi system	2
		SFT_Task11: Make detailed Test case Check default policy is Drop	Run command to check Firewall rules Find default Policy of Firewall	Have a detailed Test case with all mandatory fields: Test Purpose Precondition Test steps Expected Result (Have specific expected result for each Test step)	0.5
		SFT_Task12: Make detailed Test case Check valid packet will be accepted by firewall	You need to check and know : Choose a firewall rule Create and send Ethernet packet with data following information accept in that firewall rule Verify Ethernet packet in Raspberry Pi system	Have a detailed Test case with all mandatory fields: Test Purpose Precondition Test steps Expected Result (Have specific expected result for each Test step)	1

		SFT_Task13: Make detailed Test case Check invalid packet will be dropped by firewall	You need to check and know : Choose other firewall rule Create and send Ethernet packet with data NOT following information accept in that firewall rule Verify Ethernet packet in Raspberry Pi system	Have a detailed Test case with all mandatory fields: Test Purpose Precondition Test steps Expected Result (Have specific expected result for each Test step)	1
		SFT_Task14: Make detailed Test case verify the system accept packet with Establish state	You need to: Understand about packet with Establish state Create and send Ethernet packet Establish state Verify that packet in Raspberry Pi system	Have a detailed Test case with all mandatory fields: Test Purpose Precondition Test steps Expected Result (Have specific expected result for each Test step)	1
		SFT_Task15: Make detailed Test case verify the system Drop packet with non-NEW state and has flag SYN	You need to: Understand about packet with non-NEW state and has flag SYN Create and send packet with non-NEW state and has flag SYN Verify that packet in Raspberry Pi system	Have a detailed Test case with all mandatory fields: Test Purpose Precondition Test steps Expected Result (Have specific expected result for each Test step)	1
		SFT_Task16: Make detailed Test case verify firewall mitigates SYN flood attack	You need to: Understand about SYN flood attack Perform SYN flood attack from your laptop Verify SYN flood attack in Raspberry Pi system	Have a detailed Test case with all mandatory fields: Test Purpose Precondition Test steps	2

				Expected Result (Have specific expected result for each Test step)	
		SFT_Task17: Make detailed Test case verify firewall mitigates UDP flood attack	You need to: Understand about UDP flood attack Perform UDP flood attack from your laptop Verify UDP flood attack in Raspberry Pi system	Have a detailed Test case with all mandatory fields: Test Purpose Precondition Test steps Expected Result (Have specific expected result for each Test step)	1
		SFT_Task18: Make detailed Test case verify firewall support NAT rules/ DNS tables	You need to: Understand about NAT rules/ DNS tables Verify firewall support NAT rules/ DNS tables	Have a detailed Test case with all mandatory fields: Test Purpose Precondition Test steps Expected Result (Have specific expected result for each Test step)	1.5
	Open Source Software Vulnerability Scan (OSSVS)	**OSSVS-1** Research supply chain attack and supply chain management Research OSS scanner (dependency-check, Snyk)	Research real-world supply chain attack (e.g. Solarwinds hack) Research supply chain management Research OSS scanner such as dependency-check and work with a small sample project	Understand supply chain attack and supply chain management Have a scan report using dependency-check for a sample Java project	3

		OSSVS-2 Research SBOM management and tool Research fosslight-yocto scanner	Research SBOM management and tool (sbom-manager, meta-wr-sbom) Research fosslight-yocto and run fosslight-yocto to scan WebOS build directory	Have fosslight-yocto scan result containing OSS version in WebOS	1
		OSSVS-3 Create a list of CVEs existed in specific OSS CVE triage	Choose 2-3 OSS to create a list of CVEs (openssl, libxml2, etc.) Search for CVEs associated with the OSS version and create a report (self-defined Excel template) for vulnerability management CVE triage (only for CVEs meeting certain criteria, e.g. CVSS score >= 7.0): check the source code if the OSS is vulnerable (not patched) check the mitigation plan (upgrade or patch)	Have an intermediate OSSVS report that contains OSS version and CVEs for 2-3 OSS.	3
		OSSVS-4 Upgrade/patch OSS Make final report	Upgrade or apply patch to OSS to mitigate CVEs Re-build to check if OSS is upgraded/patched Finalize report	Have a final OSSVS report with all CVEs (meeting certain criteria) are mitigated.	2
	Exploit Memory (EM)	**EM-1** Research basic binary exploitation techniques Research memory protection mechanism Research compile options for enabling memory protection mechanism in gcc	Research basic binary exploitation techniques such as buffer overflow, shellcode, etc. to understand the need of memory protection mechanisms Research memory protection mechanisms such as RELRO, Canary, NX, Fortify Source, etc. to understand how they can protect an executable Research compile options in gcc to enable memory protection mechanisms (e.g. -fstack-protector-all, etc.)	Understand basic binary exploitation concept Understand how each memory protection mechanism can protect an executable Understand related gcc compile options	3

		EM-2 Analyze checksec.sh script to understand how it checks for memory protection mechanisms	Read checksec.sh script to understand how it checks if a mechanism is enabled or not.	Understand how to check if memory protection mechanism is enabled: RELRO PIE Canary NX	2
		EM-3 Execute checksec.sh (in WebOS build directory) Process the result from checksec.sh script (exclude OSS binaries, etc.) to make a final test report	Execute checksec.sh script to scan all binaries in the build directory Based on the .csv report from checksec.sh, make a final report (self-defined Excel template)	Have scan result of checksec.sh execution Have a final report for EM	0.5
		EM-4 Modify compile options to fix EM issues if any Re-build and regression test to verify	For each EM issues, add necessary compile options to the bitbake recipe to enable memory protection mechanism Re-build the image and execute checksec.sh again to verify if the issue is resolved	All EM issues resolved	1
	Operational Security Hardening (OSH)	**OSH-1** Research OSH requirements	Based on the RHEL OSH checklist, create a new checklist for Raspberry Pi by excluding non-applicable requirements (e.g. RHEL-specific requirements) Research OSH requirements to have a brief understanding of the requirement	Have a OSH requirement list for Raspberry Pi Understand all OSH requirements Have technical solution for all OSH requirements	3
		OSH-2 Implement OSH scripts for OSH requirements	Define logging convention for the output of the OSH script Write a Bash script for each OSH requirement	Have OSH script for all OSH requirements	3

		OSH-3 Execute OSH scripts on Raspberry Pi board Make a test report	Execute OSH scripts on Raspberry Pi and make a test report (self-defined Excel template).	Have a final report for OSH	0.5
		OSH-4 Define mitigation plan for OSH issues Implement patch to fix OSH issues Regression test to verify	For each OSH issue, define the solution and apply it to fix the issue. Re-build the image and re-run OSH to check if the issue is resolved	All OSH issues are resolved.	1.5
					95 days

Team Size: 2 Students

- 1 Scrum Master + Developer
- 1 Developer

Sprint Planning:

- **1 Sprint:** 2 weeks
- **Number of Sprints**: 5 sprints
- **Assignment**:
  - Student1:
    - Firewall development: F1 -> F12 Task
    - Firewall test: SFT_Task9 -> SFT_Task18

        - OSH: OSH_1 -> OSH_4
    - Student2:
        - Smack development: S1 -> S9 Task
        - Smack test: SFT_Task1 -> SFT_Task8
        - OSSVS: OSSVS_1 -> OSSVS_4
        - EM: EM_1 -> EM_4