Intelligent Ground Vehicle Competition 2023

Indian Institute of Technology Madras
Team Abhiyaan
Vikram

IGVC Cyber Challenge Report

I hereby certify that the development of the vehicle, Vikram, as described in this report, is equivalent to the work involved in a senior design course. This report has been prepared by the students of Team Abhiyaan under my guidance.

Dr. Sathyan Subbiah
Faculty Advisor, Team Abhiyaan
Professor, Department of Mechanical Engineering
IIT Madras

SATHYAN SUBBIAH, Ph.D.
Professor
Department of Mechanical Engineering
Indian Institute of Technology Madras
Chennai - 600 036, India

Team Members

Electronics	Mechanical	Software
Adil Mohammed K	Advait Abhijeet Kadam	Aahana Hegde
Arun Krishna AMS	Ananya Dhanvantri Guruprasad	Aayush Agrawal
Dhaksin Prabu K	Aneesh Bhandari	Amar Nath Singh
Kesava Aruna Prakash R L	Arvind P	Keerthi Vasan M
M.R.Kaushik	Hiran Neelakantan	Lalit Jayanti
Niranjan A. Kartha	Jayesh soni	Saiharan Rajakumar
Rahul	Krishma Mehta	Sukriti Shukla
Soumya Ranjan Behera	Mugdha Meda	Suneet Swamy
Vamsi Krishna Chilakamarri	Reitesh KV Raman	Suraj Rathi
	Saumya Mathur	U.K.Arvindan
	Shobhith Vadlamudi	
	Sudharsan Raja	

*Names are hyperlinks for emails.

Contents

1. THE NIST RMF PROCESS

1.1. *Overview*

Security issues often come from oversight or negligence on the developers' side. In order to prevent this to a large extent, one could follow a comprehensive process that tries to make sure that all the boxes are checked with regard to security, before deploying a product out into the world. The NIST RMF is such a process, in which an organization decides the level of security risk it can tolerate, and applies protections accordingly. It prescribes the following steps:

1.1.1. *Prepare*

The "Prepare" step, which was newly added to the RMF in 2018, involves properly organizing the team to deal with information security, and also clearly establishing the level of risk tolerance required. We identify the different kinds of information processed, and decide what level of security each type of information entails, based on the priorities of the organization.

1.1.2. *Categorize*

We examine of the information we identified in the Prepare step. We determine the worst-case impact that a malicious actor could have on the system if they managed to compromise the confidentiality, integrity, or availability of all these kinds of information.

1.1.3. *Select*

Once we are done with categorization, we can select appropriate controls to protect the information, and tailor them according to our needs. We plan out how we will implement and follow-up on them. We also create a strategy for continuously monitoring the system after the controls have been deployed.

1.1.4. *Implement*

After selecting appropriate controls, we implement them in our system and document it.

1.1.5. *Assess*

A team, ideally independent of the one that implemented the controls, is selected to determine whether the controls are functional and meet the privacy requirements of the system. Any extra privacy concerns are also documented, and plans are made to remedy them over time.

1.1.6. *Authorize*

An authorizing official considers the assessments and plans made, and determine whether they are acceptable to the organization.

1.1.7. *Monitor*

We continuously monitor the deployed controls based on the strategy decided in the Select step, in order to make sure that our controls stay functional, and no new risk factors have come up.

1.2. *Identified Threat Concept*

The hypothetical use-case for our bot is as a delivery robot within our institute. People could request the bot's services using an app, place a package inside, and ask for it to be delivered to a location within the campus. Once that location is reached, the intended recipient could command the bot to open up, and retrieve the package.

1.3. *How the team applies the RMF*

As we are team of 30 college students as opposed to a large organization, we have focused more on categorizing threats and implementations of controls, than documenting organizational risk tolerance standards.

We first came up with an architecture to implement our sevice, with information security in mind. We use two different information systems — one is the NUC, and the other is the server which co-ordinates delivery requests. The NUC only takes commands from the server and gives location telemetry, and has no knowledge of the user data or delivery requests.

Next, we identified the kinds of information that we need to process.

2. APPLYING THE RMF

2.1. *Prepare*

This is the information that we need to process in order for our service to work properly:

1. Data that users provide us to register for the service (stored on the server)
   - Name
   - Roll number
   - E-mail address
   - Password used for Registration
2. Data that users provide us while using the service (stored on the server)
   - Pick-up location
   - Delivery location
   - Recipient details
3. Telemetry data sent from the bot to the server
   - Location
4. Commands sent from the server to the bot
5. Data stored on the bot required for it to function (stored on the bot)
   - Source code
   - GPG keys
6. Debugging information and commands (between the bot and developers in the team)
   - Sensor data
   - Tele-operation commands
   - Remote e-stop
7. Delivery tokens for opening the bot to access delivery contents (sent to the intended recipient when the bot reaches its destination)

2.2. *Categorize*

Kind of Information	**Confidentiality**	**Integrity**	**Availability**	**Description of threat**
User profile data	High	High	Low	Malicious actors might try to harvest user information that they may later use for spamming or identity theft.
Usage data	High	Low	Low	We do not want to leak user location or delivery details as these are highly sensitive.
Bot-location-telemetry	None	Moderate	Moderate	We want users to know of the bot location only when it is currently performing a delivery for them, or it is close to their location.
Bot commands	Low	High	Moderate	We do not want malicious actors to be able to send rogue commands to the bot, as they may be able to take control over it.
Bot credentials and code	High	High	High	Access to passphrases used to control the bot would result in a complete takeover of the service.
Debugging information	Low	High	Moderate	Debug access to the bot can also let anyone take full control over it by tele-operating it.
Delivery tokens	High	Moderate	Moderate	We do not want unauthorized people to access others' packages

2.3. *Select, Implement*

Based on the impact that a particular kind of information has on our system, the NIST RMF recommends that we select a particular set of controls. We then tailor each control to our specific needs.

We have selected controls based on our judgement of which ones would be the most appropriate, while also considering their demonstrability.

2.3.1. *Controls that we have implemented*

AC-1 (Access Control Policy and Procedures)

Implementation	**Demonstration Strategy**	**Information Protected**	**Threat mitigated**
Only current team members are allowed to access the code on our system. Once someone leaves the team, their public keys are removed from the git repositories and computers used.	Observe that `~/.ssh/authorized_keys` file on the NUC only have team members' public keys. Also observe that the GitHub repositories have only team members in them.	Bot credentials and code, debug information	Malicious actors with access to ex-members' accounts
Only people from within the institute are authorized to request deliveries.	Observe that the server requires you to specify an institute e-mail ID when you register.	User details, bot commands	Spammers from outside the institute trying to DoS the service by registering rogue accounts

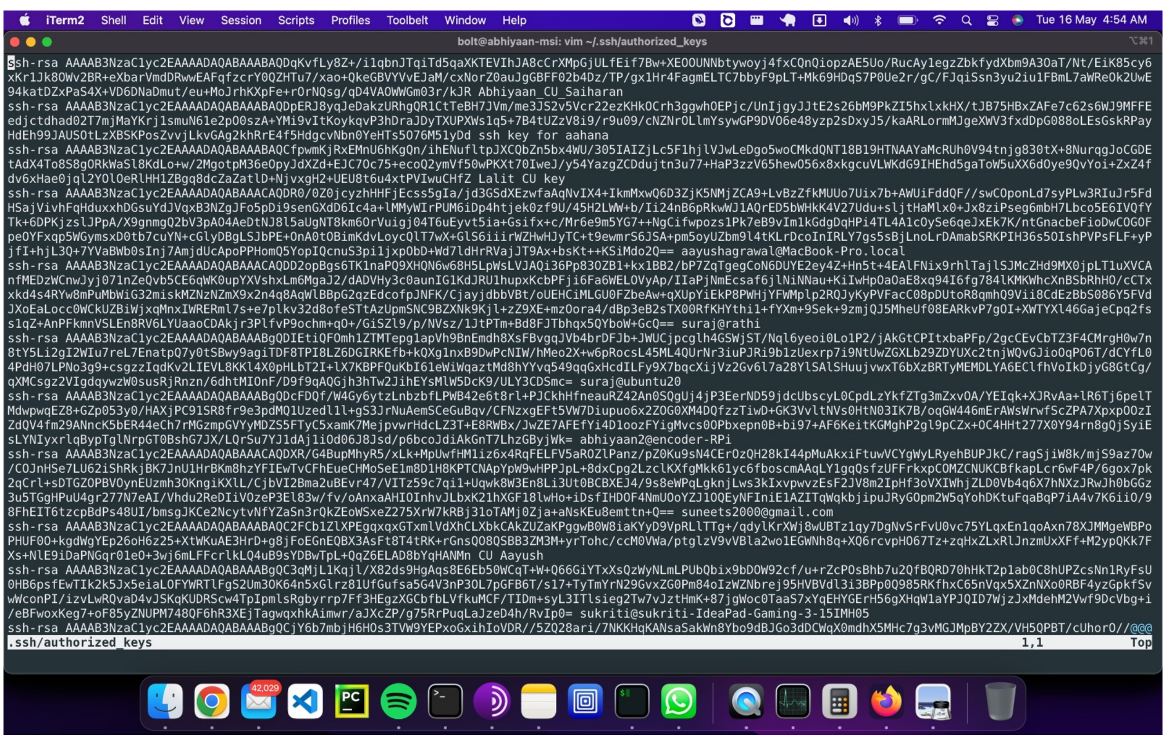

Figure 1. Public keys of team members in /.ssh/authorizedkeys

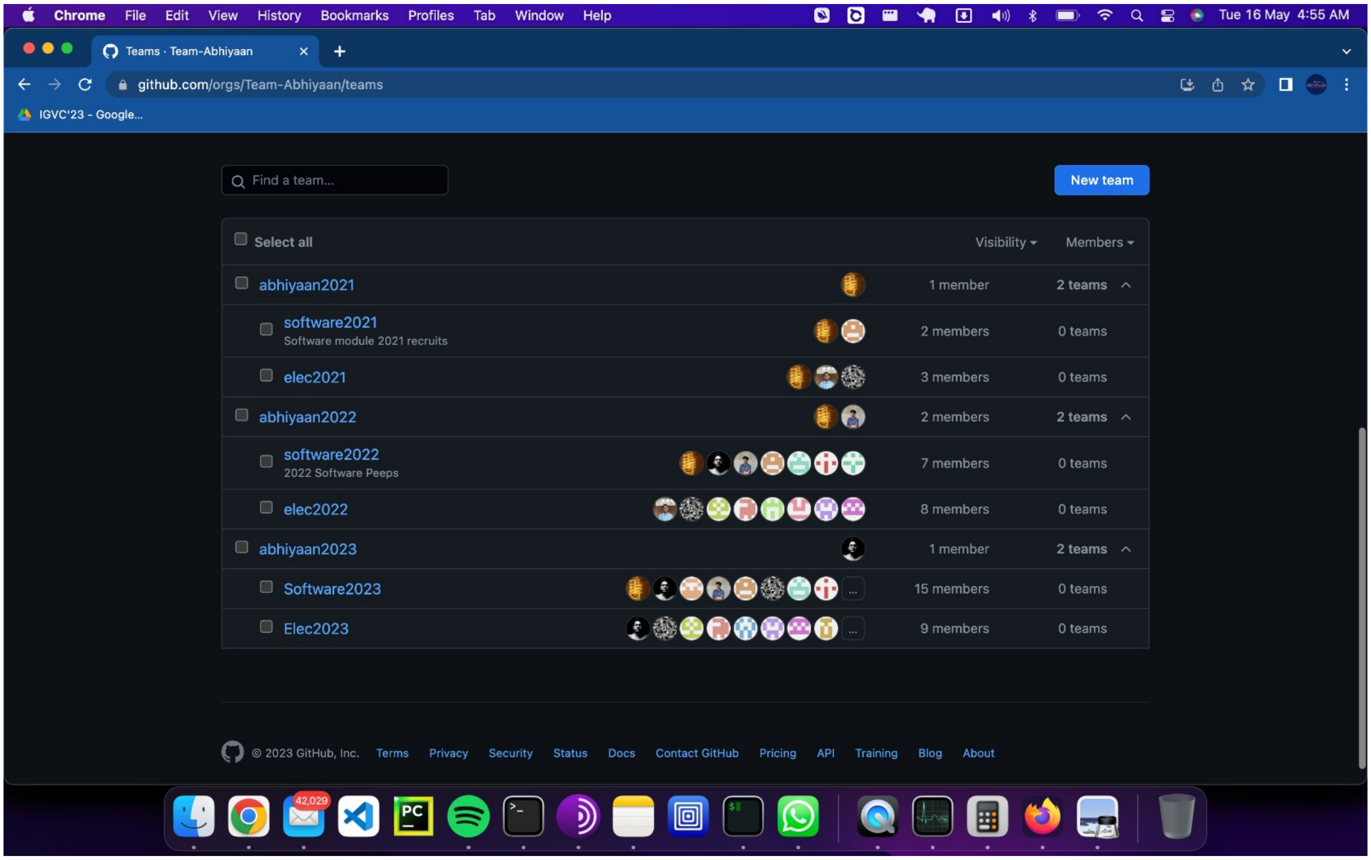

Figure 2. Only current team members have access to Github repositories

AC-2 (Account Management)

Implementation	**Demonstration Strategy**	**Information Protected**	**Threat mitigated**
Only team leads' accounts have ‘root‘ access on the NUC.	Observe that only the team leads are in the ‘sudo‘ group.	Bot credentials and code	Malicious actors with access to team members' accounts
A separate `www-data` user is used for running deployed code. This user does not have read access to the source code, and has heavily curtailed permissions. (SC-2)	Log in as `www-data` and observe that this user cannot open extra ports, read files, etc.	Bot credentials and code, debugging information	Someone with code execution access on the NUC now requires privilege escalation to do serious damage.

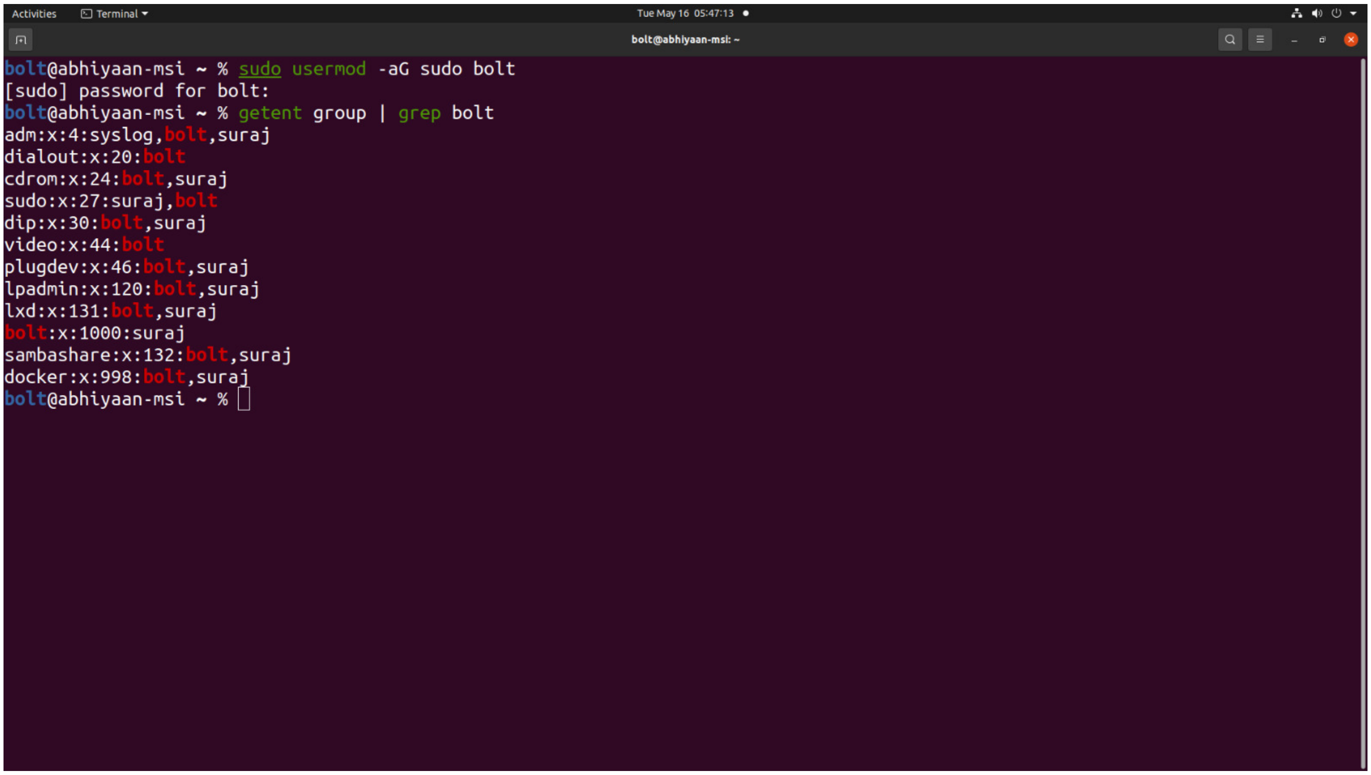

Figure 3. `bolt` & `suraj`: user accounts of team leads having `sudo` access

AC-4 (Information Flow Enforcement)

Implementation	**Demonstration Strategy**	**Information Protected**	**Threat mitigated**
The bot refuses to connect to the server unless the connection is served over HTTPS.	We could temporarily switch the server to HTTP, and observe that the bot goes offline. Starting it as HTTPS makes it work again.	Bot commands, telemetry	Man-in-the-middle attacks
All unused networking ports are blocked on the NUC.	Try to run netcat as `www-data` on a blocked port, and observe that we are not allowed to do so.	Debug information, bot credentials and code	Hackers cannot run a shell.
The `www-data` account on the NUC is only allowed to access a pre-determined set of IP addresses. Any packets going elsewhere are filtered out and logged.	Try pinging an unauthorized IP address as `www-data` and observe that it fails.	Debug information, bot credentials and code	Hackers cannot send payloads to the NUC.

AC-7 (Unsuccessful Logon Attempts)

Implementation	**Demonstration Strategy**	**Information Protected**	**Threat mitigated**
Users are timed out of logins on the server after five failed attempts.	Perform five failed attempts and observe that we are locked out temporarily.	User data, usage	Hackers cannot easily brute-force the user account passwords.

AC-12 (Session Termination)

Implementation	**Demonstration Strategy**	**Information Protected**	**Threat mitigated**
All logins to the NUC time out after five minutes of inactivity.	Open `~/.bashrc` and observe that the environment variable `TMOUT` is set to **300**. Change this to **5** to demonstrate how it works for a timeout of 5 seconds.	Bot credentials and code, debug information	Minimizes the risk of someone catching one of the team members' laptops unlocked and accessing the NUC from there.

AC-17 (Remote Access)

Implementation	**Demonstration Strategy**	**Information Protected**	**Threat mitigated**
We only allow access to SSH using public keys of sufficient length.	Observe that trying to SSH into any account on the NUC using a password always fails. Also observe that the `authorized_keys` on the NUC have sufficiently large keys.	Bot credentials and code, debug information	Hackers cannot access the NUC remotely by guessing passwords.

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDD2opBgs6TK1naPQ9XHQN6w68H5LpWsLVJAQi36Pp83OZB1+kx1BB2/bP7ZqTgegCoN6DUYE2ey4Z+Hn5t+4EAlFNix9rhlTajlSJMcZHd9MX0jpLT1uXVCA
nfMEDzWCnwJyj071nZeQvb5CE6qWK0upYXVshxLm6MgaJ2/dADVHy3c0aunIG1KdJRU1hupxKcbPFji6Fa6WELOVyAp/IIaPjNmEcsaf6jlNiNNau+KiIwHpOaOaE8xq94I6fg784lKMKWhcXnBSbRhHO/cCTx
xkd4s4RYw8mPuMbWiG32miskMZNzNZmX9x2n4q8AqWlBBpG2qzEdcofpJNFK/CjayjdbbVBt/oUEHCiMLGU0FZbeAw+qXUpYiEkP8PWHjYFWMplp2RQJyKyPVFacC08pDUtoR8qmhQ9Vii8CdEzBbS086Y5FVd
JXoEaLocc0WCkUZBiWjxqMnxIWRERml7s+e7plkv32d8ofeSTtAzUpmSNC9BZXNk9Kjl+zZ9XE+mzOora4/dBp3eB2sTX00RfKHYthi1+fYXm+9Sek+9zmjQJ5MheUf08EARkvP7gOI+XWTYXl46GajeCpq2fs
s1qZ+AnPFkmnVSLEn8RV6LYUaaoCDAkjr3PlfvP9ochm+qO+/GiSZl9/p/NVsz/1JtPTm+Bd8FJTbhqx5QYboW+GcQ== suraj@rathi

Figure 4. `RSA 2048` bit asymmetric public encryption key

AC-18 (Wireless Access)

Implementation	Demonstration Strategy	Information Protected	Threat mitigated
The on-board router employs WPA2 PSK encryption, and uses a strong password, and the SSID is not broadcast.	Observe that encryption is enabled for the WiFi network, and that we need to manually enter the SSID to connect to it.	Bot credentials and code, telemetry data, debug information	Hackers cannot bruteforce the WiFi password, or perform evil twin attacks.
The administrator page of the router uses a different passphrase than the network. (AC-18(4))	Observe that the WiFi password (copy/pasted from a password manager) does not work on the router administration page.	Bot credentials and code, telemetry data, debug information	Hackers with access to the WiFi network cannot change any settings.

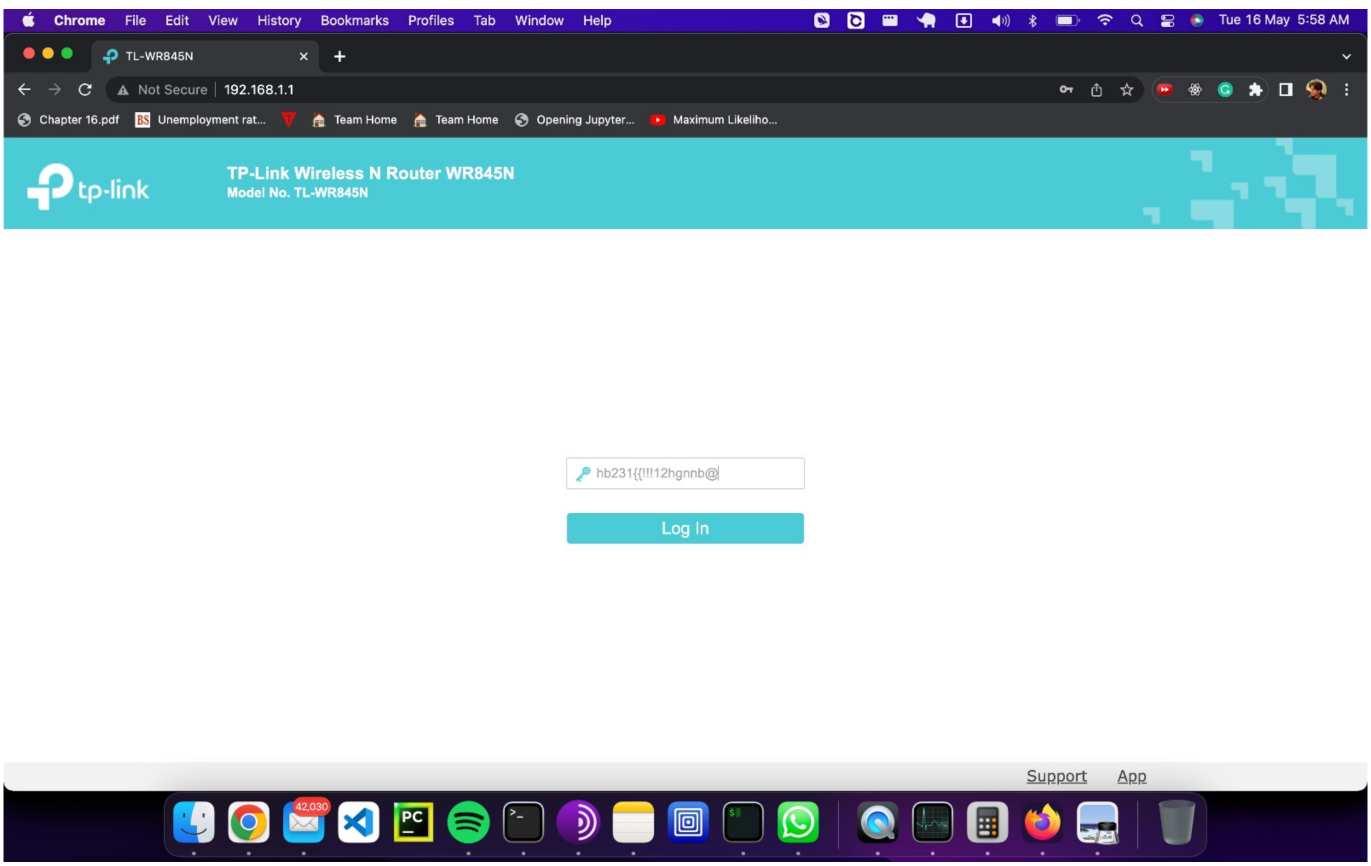

Figure 5. Router Administration page password(*For Illustrative purposes only*)

IA-3 (Device Identification and Authentication)

Implementation	Demonstration Strategy	Information Protected	Threat mitigated
Only machines with authorized MAC addresses are allowed to connect to the network. (AC-18(1))	Observe that a phone with the wireless credentials cannot log in to the network, while a laptop with an authorized MAC address can.	Bot credentials and code, telemetry data, debug information	Hackers cannot access the network unless they know team members' MAC addresses as well.
Authenticated communication between Tiva and NUC is established.	On connecting a rogue TIVA, the NUC throws an error and does not interact further with the TIVA.	Debug information	Hackers with physical access cannot easily replace the TIVA.

IA-5 (Authenticator Management)

Implementation	Demonstration Strategy	Information Protected	Threat mitigated
All users are required to set strong passwords while signing up on the server.	The server refuses to create an account if the password is too weak.	User data	Hackers cannot bruteforce their way into user accounts.
The team lead saves passwords and encryption keys in a password manager encrypted using a strong password.	Unlock the password manager and show the list of passwords saved.	User data, bot credentials and code	By not storing encryption keys in plaintext, we can prevent people with physical acceess to machines from decrypting data.

IA-9 (Service Idenfication and Authentication)

Implementation	Demonstration Strategy	Information Protected	Threat mitigated
TLS is used to verify the authenticity of the server, and the NUC refuses to connect unless the server is served over HTTPS. (AC-4(4))	Notice that the server uses HTTPS, and redirects HTTP connections to HTTPS. Also notice that if we turn off HTTPS on the server, the NUC throws an error.	Bot telemetry, user data, bot commands	Man-in-the-middle attacks

SC-8 (Transmission Confidentiality and Integrity)

Implementation	Demonstration Strategy	Information Protected	Threat mitigated
User passwords are salted and hashed before they are stored.	Create a new dummy account and observe that the database entry is salted and hashed.	User data	Someone with access to the database cannot know user passwords.

SC-13 (Cryptographic Protection)

Implementation	Demonstration Strategy	Information Protected	Threat mitigated
User data on the server is encrypted.	Observe the database controls to verify that encryption is enabled.	User data, usage details	A rogue actor on the server cannot view the user database unless they have encryption keys.

SC-41 (Port and I/O Device Access)

Implementation	Demonstration Strategy	Information Protected	Threat mitigated
Unused I/O ports on the NUC are disabled.	Plug in a microcontroller into a disabled port, and observe that it does not get detected.	Bot credentials and code, debug information	Rogue live USBs cannot be booted into.

SI-5 (Security Alerts, Advisories, and Directives)

Implementation	Demonstration Strategy	Information Protected	Threat mitigated
If the bot is idling and it detects that its GPS location has shifted far enough from its starting position, the team lead is alerted.	Push the bot away from its location for a sufficient distance, and notice that the team lead gets notified.	Miscellaneous	Someone cannot physically steal the vehicle without alerting the team lead.

SI-14 (Non-Persistence)

Implementation	Demonstration Strategy	Information Protected	Threat mitigated
The NUC uses non-persistent storage for every partition accessible by `www-data`.	Observe that a file created in one boot does not appear once you reboot.	Bot commands, debug information	Someone with RCE on the NUC cannot make persistent changes to it.

SI-16 (Memory Protection)

Implementation	Demonstration Strategy	Information Protected	Threat mitigated
The NX bit is enabled in the NUC.	Run `dmesg | grep 'Executable Disable'` and observe that protection is active. This means that NX is enabled.	Bot credentials and code, debug information, bot commands	Arbitrary shellcode cannot be executed on the NUC.

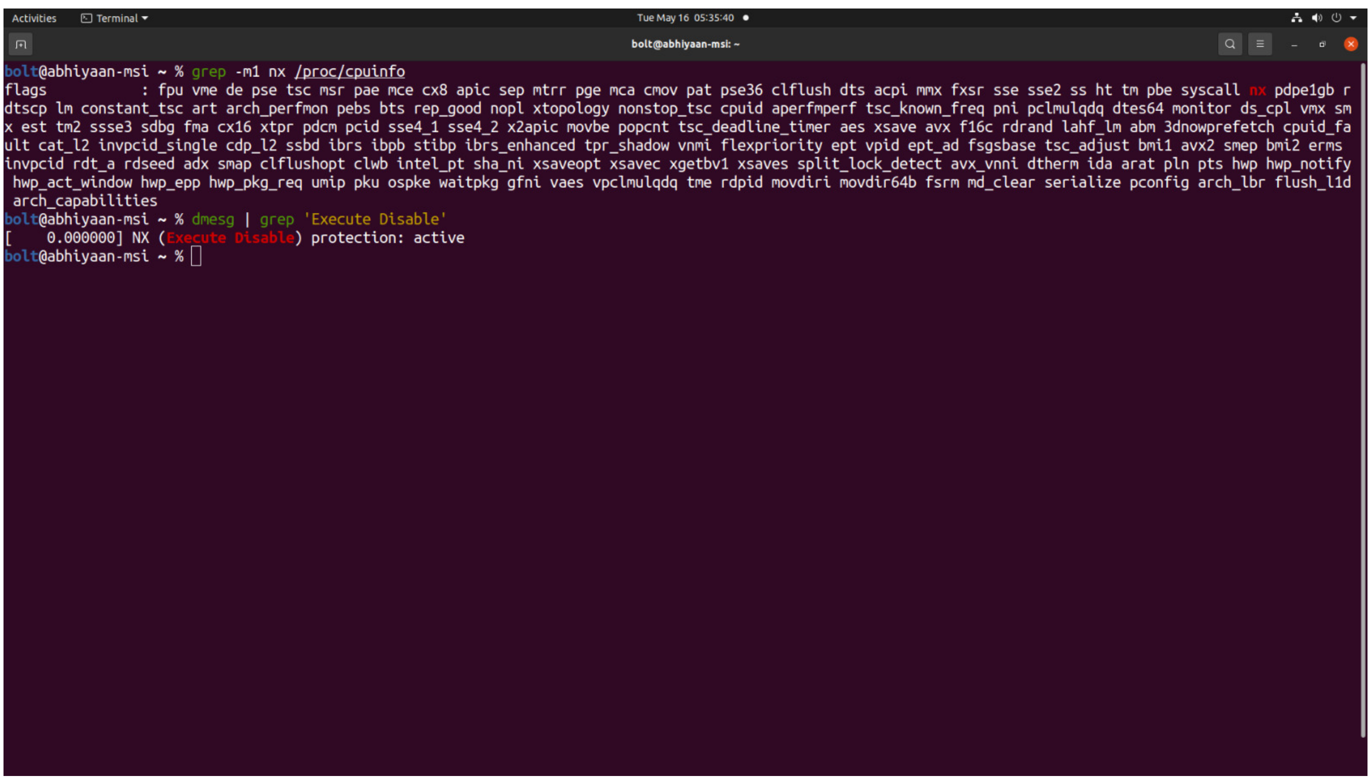

Figure 6. `dmesg | grep 'Execution Disable'` indicating `protection: active`

2.3.2. *Controls that are not implemented but would be appropriate*

SC-13 (Cryptographic Protection)

Implementation	Demonstration Strategy	Information Protected	Threat mitigated
The SSD on the NUC is encrypted to prevent someone with physical access to the system from harvesting keys.	Boot up the NUC and observe that it asks for a decryption key.	Bot credentials and code	Someone with physical access to the NUC cannot view code and data unless they have encryption keys.

SI-7 (Software, Firmware, and Information Integrity)

Implementation	Demonstration Strategy	Information Protected	Threat mitigated
The TIVA microcontroller has Secure Boot enabled, to prevent unauthorized code from being uploaded. (SI-7(15))	Observe that any code upload to the TIVA fails unless it is correctly signed.	Bot credentials and code, debug information	Someone with physical access to the microcontrollers cannot upload rogue code.

2.4. *Assess, Authorize*

The team lead reviews the controls selected and the protections implemented, and approves them.

2.5. *Monitor*

Kernel logs, WiFi logs, and server are continuously checked by team members, to make sure that no unauthorized access is happening.