2023 NDIA MICHIGAN CHAPTER
GROUND VEHICLE SYSTEMS ENGINEERING
AND TECHNOLOGY SYMPOSIUM
AUTONOMY, ARTIFICIAL INTELLIGENCE & ROBOTICS TECHNICAL SESSION
AUGUST 15-17, 2023 - NOVI, MICHIGAN

SECURE UPDATE PROCESS FOR ROBOTIC AND AUTONOMOUS SYSTEMS

Sabrina Pereira[1], Cameron Mott[1], Dariusz Mikulski[2]

[1]Intelligent Systems Division, Southwest Research Institute®, San Antonio, TX
[2]US Army DEVCOM Ground Vehicle Systems Center, Warren, MI

ABSTRACT

The importance of hardening robotic and autonomous systems (RAS) considered for field deployment against cyber threats has been recognized by organizations across the Department of Defense (DoD). Among these needs is the ability to securely provide these modern military vehicles with software updates containing critical new functionality and security improvements. A secure update process and system for military RAS has been implemented building on a framework designed for the automotive industry. Demonstrations of the capabilities and mitigations against possible attacks on the update process will be performed on a RAS MRZR in a mock field environment.

Citation: S. Pereira, C. Mott, D. Mikulski, "Secure Update Process For Robotic And Autonomous Systems," In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 15-17, 2023.

1. INTRODUCTION

Organizations across the Department of Defense (DoD) recognize that hardened cybersecurity in robotic and autonomous systems (RAS) considered for field deployment is essential. The Cybersecurity for Robotic & Autonomous Systems Hardening (CRASH) Joint Capabilities Technology Demonstration (JCTD) was created to develop a comprehensive cybersecurity software solution to improve the cyber resiliency of RAS against digital adversaries at various touch points. Among these needs is the ability to securely provide these modern military vehicles with software updates containing critical new functionality and security improvements.

As part of this effort, Southwest Research Institute (SwRI) is working with Ground Vehicle Systems Center (GVSC) to develop a secure software update capability that is secure by design and based on the Uptane standard. Over the past seven years, security professionals from the embedded and automotive domains have dedicated their expertise to develop Uptane as an open and peer-reviewed cybersecurity framework.

The secure update mechanism will be demonstrated on a Modular Autonomy and Robotic Software (MARS)/Robotics Technology Kernel (RTK) system, MRZR, in a mock field environment. SwRI has worked closely with the developers that are currently involved in providing updates to a RAS MRZR and has worked to tailor the software update implementation to their needs.

2. BACKGROUND

In the current development process, software updates for a RAS MRZR are directly performed by developers that are physically near the vehicle and target 2-4 powerful onboard computers. The developers log into the computers as a system administrator with the capability to modify anything on the system. An update may include new code from other developers, updated supporting libraries, or system configuration changes. In some scenarios, the entire system is rebuilt and deployed. Updates are synchronized from a developer's laptop to the computers running on the RAS using rsync [1], encrypted via Secure Sockets Layer (SSL) using a shared username and password.

This process is not at the maturity necessary for deployment. Adversaries can use the update process as a way to attack and steal sensitive information, infiltrate vehicles, and even install malware. There are security risks associated with a manual update process like this, such as relying on a single person with administrator privileges to authenticate, using only a password for authentication, and a lack of visibility into what changes were made by whom.

The developed secure software update system is based on Uptane, a peer-reviewed update framework designed for the automotive industry. This framework is designed to withstand attacks from malicious actors who aim to compromise in-vehicle computers, servers, and networks used in delivering updates. The system will require approvals from multiple parties, each limited in their privileges, and clear indications about the update status of a vehicle. Additionally, it will provide mechanisms to detect any manipulations during the update process and provide situational awareness to a command-and-control center.

3. SYSTEM DESIGN OVERVIEW

The update system architecture at a high level is pictured in Figure 1. The design includes two servers: an Image Server and a Director Server, as well as a secure source of time. Each server operates within a public key infrastructure and is responsible for storing, generating, and sending metadata about the updates. This metadata is used by the vehicles to validate the updates and detect potential security threats.

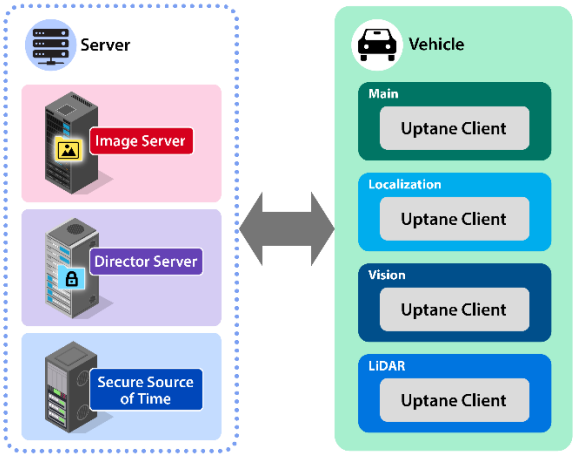

Figure 1. Server and Vehicle Architecture Depicting the Separation of Roles

The Image Server contains the developed update files and related metadata, while the Director Server provides encrypted updates and customized metadata on a vehicle-by-vehicle basis. When an update is performed, the main electronic control unit (ECU) in the vehicle downloads, decrypts, and verifies the metadata from the servers. If all verification checks pass, the update files are downloaded, where they undergo another round of verification before being installed. If any verification check fails, a security attack is logged and reported back to the Director Server.

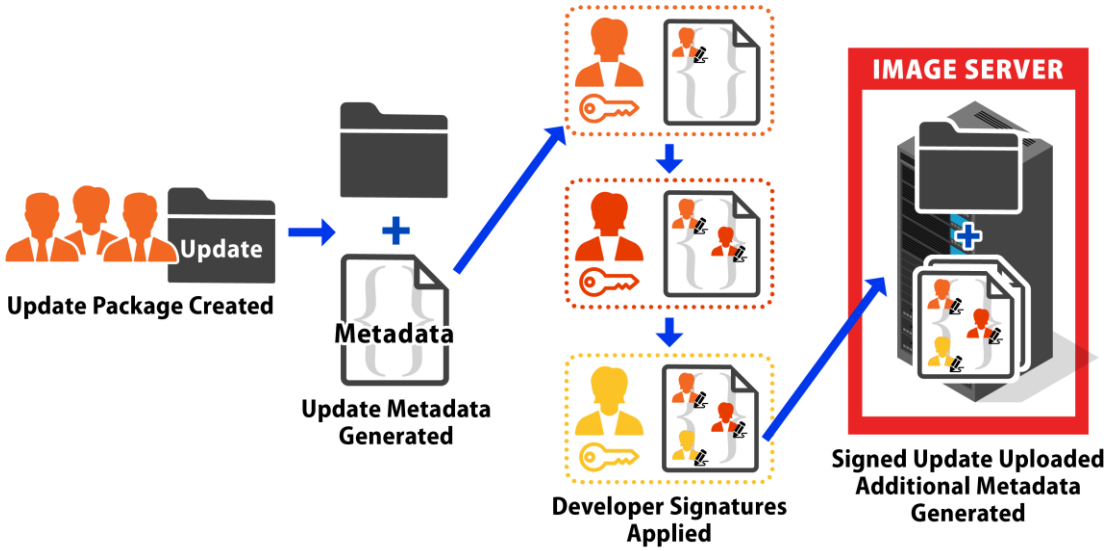

Figure 2. Developer Update Upload Process

3.1. Developer Update Upload

Once the developer has an update ready for the vehicle, they create an update package to upload to the Image Server. Through a set of provided tools and an interface with the Image Server, the developer signs the image through a metadata file that details information about the files in the update package such as file sizes and hashes.

The new update package, however, will not be eligible for selection until a quorum of developers have signed off on the update package. For example, with a signature threshold of three, two additional developers must review the update package and add their signature to the metadata file. Once the number of required signatures are performed, the update package is published and a set of additional metadata files for verification are generated on the server. This process is illustrated in Figure 2. The package is now eligible for vehicle installation.

3.2. Deployer Update Directing

Once the images become available on the Image Server, a person in a role of a Deployer can now log into the Director Server and select the vehicles to be updated. A deployer is responsible for assigning approved updates to specific vehicle ECUs; this ensures transparency and that updates are only applied to the vehicles that need them. The update deployment process is illustrated in Figure 3.

When the vehicle requests an update, the Director Server pulls the approved packages from the Image Server, encrypts them with a key specific to the vehicle, and generates a seperate set of metadata files for validation on the vehicle.

The vehicle downloads the encrypted update files from the Director Server along with both sets of metadata from the Director and Image Servers. These files are protected from adversarial attacks while in transit- the vehicle uses the metadata to perform cryptographic verification of the files.

3.3. Public Key Infrastructure

The vehicle's main method for image verification is through a series of checks on signatures contained in the metadata. The keys used to perform these signatures are created within a public key infrastructure (PKI). This PKI includes

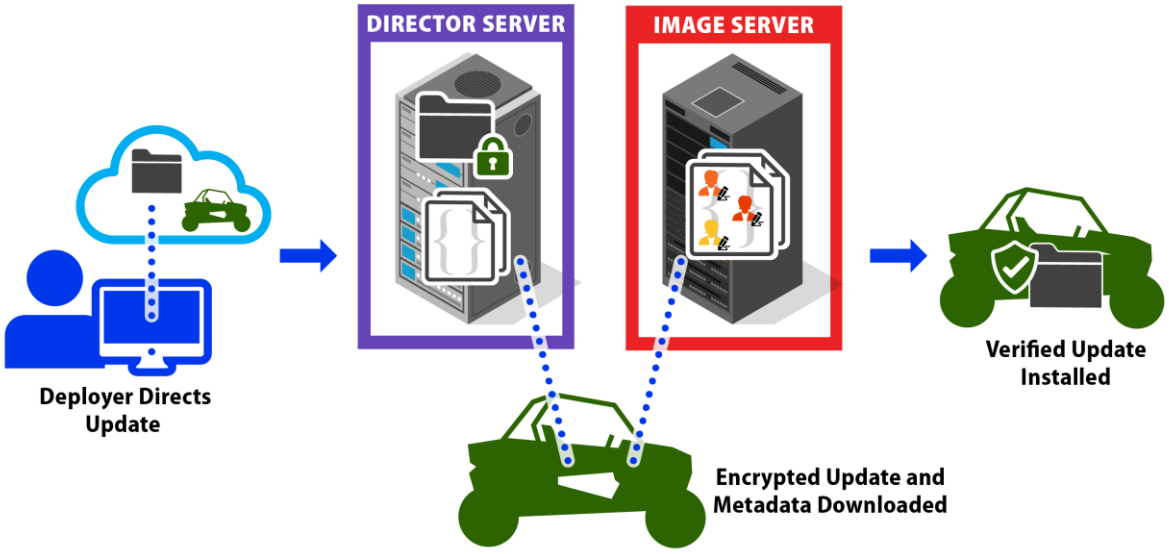

Figure 3. Update Deployment Process

establishing a hierarchy of keys, with a group of server administrators acting as the certificate authority for the system. The server administrator keys validate the keys used by the developers, the Image Server, and the Director Server (see Figure 4).

When setting up the PKI and update system, a group of personnel will be designated as the server administrators. They will create and distribute keys for themselves, the developers, and each of the servers. A quorum of the server administrators must sign two separate metadata files with the public identifier of the created keys, one to be uploaded to the Image Server and one to the Director Server. For compromise resilience, there is no overlap of keys between the system of each server.

These metadata files make up the root of trust for the vehicle. When a new vehicle is added under this system, it is provisioned with a key along with the metadata files. The public key identifier of the vehicle is then registered by a server administrator with the Director Server to authorize the vehicle to request updates.

If there is ever a compromise involving the keys (rogue developer, hijacked server, etc.), the server administrators will generate and sign files with updated public key identifiers and replace these on the servers. The vehicle update process will always pull the latest key information to perform update verification.

Additionally, access to the servers will require authentication, such that each user (developer,

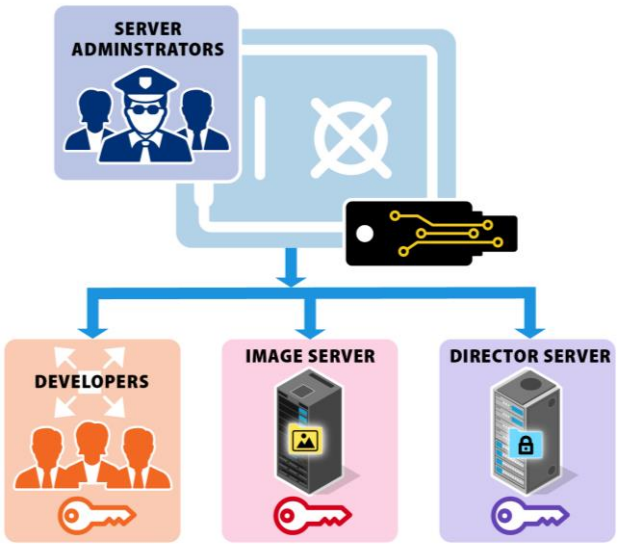

Figure 4. Key Hierarchy with Server Administrators Issuing and Validating Developer and Server Keys

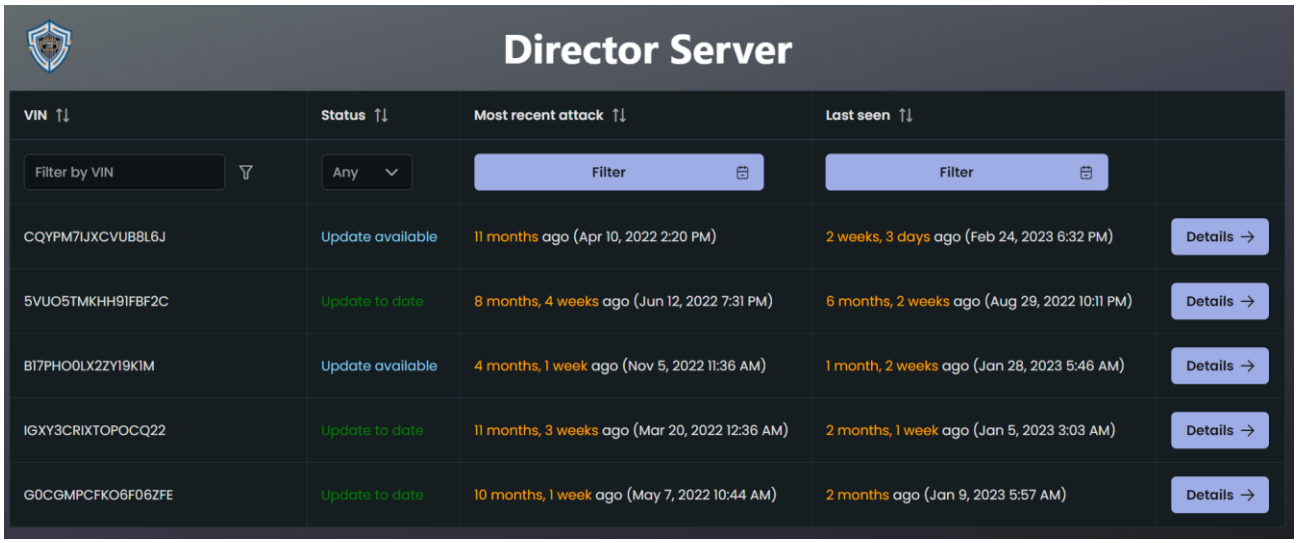

Figure 5. Current Iteration of Update Web Interface

deployer, server administrator) is only allowed to access information relevant to them.

4. UPDATE SYSTEM DEVELOPMENT

The Uptane update system has been tested by current RTK developers on a test RAS MRZR. In the test development environment, the developer acts as both the developer team and the deployer in creating the update, signing off on it and directing its installation on the vehicle computers. Currently, all the interactions are performed with command line tools on a benchtop environment with personnel familiar with Linux, current RTK update processes, and the Uptane system currently being developed.

In the future, those in developer and deployer roles would be authenticated and perform update processes through a web service, streamlining the process and widening usability. Development for this interface is currently underway with initial testing expected to occur this year. An image from the current iteration of the interface is shown in Figure 5. Additionally, the necessary steps to acquire an Interim Authority to Test (IATT) are currently being performed under the CRASH JTCD efforts to enable testing of the system in an operationally relevant environment with DoD servers.

The secure update mechanism will be demonstrated on an MRZR in a mock field environment with active adversaries represented by a red team of penetration testers. The demonstration will include a mission objective where the RAS will perform a crucial role by gathering data that would be dangerous for a soldier to gather. During the demonstration, the red team will be attempting to delay, destroy, or even take over the RAS. A deployer will direct an update under these conditions. Multiple attacks directed at the update process have already been demonstrated to be protected against in a benchtop environment.

5. RISK MANAGEMENT FRAMEWORK

The Army's risk management framework (RMF) [2] was applied to this effort. To be able to run servers in an operational environment where they would be deployed, the team followed DoD information technology guidelines and began application processes for either an IATT or an authority to operate (ATO). Establishing one of these authorities would indicate that the appropriate amount of due diligence had been followed in order to reduce risk of operating within the DoD-secured

servers. This authorization would also prove valuable to any future programs that leverage these efforts.

The RMF includes gathering artifacts that allow for security protections to be customized for the Director Server and Image Server. This effort is similar to a network security audit, helping to establish approved paths for data connectivity.

For CRASH, the team provided system diagrams, library dependencies, application programming interfaces (APIs), and network diagrams, with additional artifacts expected to be produced during the course of this project. Using these artifacts, allow-lists can be customized to permit the Uptane services that enable both server-to-vehicle and authorized agent-to-server interactions on DoD-approved servers.

6. CONTINUING WORK

During the first year of the program, a selection of attacks against vehicle update functionality and corresponding mitigation responses were demonstrated in a test bench environment. In the current program year, additional attacks and mitigations will be demonstrated with a focus on the update upload process and the safeguards against insider attacks. The target demonstration focus is on the deployment of the update system in the integrated environment, performing validation and verification of the update process between the MRZR and the system on DoD servers, or suitable representatives.

Advancements and further applications of the update capability outside current project expectations include expansion to fleets of vehicles of varying characteristics and requirements, incorporating support for post-quantum cryptographic algorithms, and deployment over heavily contested networks.

7. CONCLUSION

As part of a joint effort to improve and harden the cybersecurity of robotic and autonomous systems, a secure update system is being developed and tested. If not properly protected, an update can become an attack vector for an adversary to get access to the vehicle. This secure update system addresses this concern by providing a way for modern military vehicles to securely receive critical software updates in a manner resilient to cyber adversaries, including nation state attackers. Additional effort will be needed to field the system on DoD servers and on operational vehicles; current testing in benchtop environments show great promise.

8. REFERENCES

[1] W. Davison, "rsync," 20 October 2022. [Online]. Available: https://rsync.samba.org/.

[2] Office of the DoD Chief Information Officer, "Risk Management Framework (RMF) for DoD Systems, DoD Instruction 8510.01," 19 July 2022. [Online]. Available: https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/851001p.pdf.

[3] Uptane Community, "Uptane Standard for Design and Implementation 2.0.0," 2022. [Online]. Available: https://uptane.github.io/papers/uptane-standard.2.0.0.html.

[4] T. K. Kuppusamy, A. Brown, S. Awwad, D. McCoy, R. Bielawski, C. Mott, S. Lauzon, A. Weimerskirch and J. Cappos, "Uptane: Securing Software Updates for Automobiles," escar Europe, p. 11, 2016.

[5] C. Mott, D. Mikulski and S. Pereira, "Secure Software Updates for Robotic and Autonomous Systems," NDIA MICHIGAN CHAPTER GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM, p. 6, 2022.