Uptane: Securing delivery of software updates for ground vehicles

The first in a series of Uptane Whitepapers

Compromise–Not If, But When

In 1953, author Isaac Asimov wrote a short story called *Sally* [1]. Set in 2057, it was built on the premise that cars by that time would not only be self-driving, but also sentient. And, typical of mid 20th century science fiction, the story hinted that helpful and appealing technology carried with it a certain amount of risk.

In the 68 years since Asimov published this story, cars may not have become sentient, but they do now contain significantly more electronic components. The increasing number of smart programmable units on today's cars, called electronic control units, or ECUs, are based on large and complex code bases. The ECUs in current cars typically contain 100 million lines of code [2], significantly more than any other modern operating system. This code is very likely to contain multiple programming errors that may not be discovered for years. Coupled with the fact that today's cars support a multitude of wireless communication interfaces, the threat of a remote exploit has become more feasible. Furthermore, the distributed manner in which software is developed in the automotive industry, which involves numerous suppliers in complex supply chains, offers numerous opportunities for hackers to insert malware or to plan other types of attacks.

The consequences of such exploits have already been demonstrated in both simulated white hat hacks [3] and through accidental mishaps. [4] Now, the increasing involvement of nation-state actors and large criminal enterprises in attacking vehicles, bringing with them vast resources unavailable to the hackers of yesteryear, could mean malicious exploits will become more common, and potentially more dangerous. The 2020 Global Automotive Cybersecurity Report [5], released by UpStream Security in December 2020, shows cyber attacks on the automotive ecosystem increased 99% from 2018 to 2019, and by 700% since 2016. The results of a compromised ECU could include such worst-case scenarios as tampering with the brakes of an entire fleet of police cars, ransomware attacks on a vehicle fleet, increased vehicle theft, and other serious consequences.

It is now time to acknowledge that the science fiction nightmares of the 20th century have become the real-world challenges of the 21st. The development and implementation of reliable, adaptable, and resilient defenses for passenger vehicles and light trucks has never been more important.

This whitepaper describes Uptane, the first software update security system for the automotive industry that was intentionally developed to resist even attacks by nation-state level actors. Uptane uses layered defense mechanisms so the security of automotive software updates does not degrade all at once, but is supported by a hierarchy in which different levels of access to vehicle's or the

automaker's infrastructure must be gained. By building these multiple levels into the security system, even if attackers compromise servers, bribe operators, or gain access to vehicular networks, these incursions will be limited in how much damage they can cause.

Addressing the double-sided nature of software updates and why automobiles are so hard to secure

Though not directed against automobiles, the recent SolarWinds hack in which companies, government agencies, and academic institutions suffered significant data breaches after malware was slipped into a system management software update, is a sobering reminder that although it is necessary to update software, such updates are also always fraught with risk. The full impact [6] of the SolarWinds attack, which is known to have affected computer systems within the U.S. Departments of Defense, State, Homeland Security, Treasury, Commerce, and Energy, is still unknown.

Yet, ignoring software updates is not a viable option either. As Uptane steering committee member Dr. Justin Cappos, an associate professor of computer science and engineering at NYU Tandon School of Engineering, explained in a December 20, 2020 article in Yahoo Finance [7], nation-state actors are gravitating to such attack targets because updating software is something system maintainers are "supposed to be doing." In a sense, OEMs and suppliers find themselves between the proverbial rock and a hard place. But, even though software updates are risky, Dr. Cappos cautioned, "If you don't apply software updates, you're absolutely, definitely vulnerable because old software is vulnerable software."

For automotive OEMs, securing software updates is much more complicated than for a conventional server-based enterprise system. The ECUs are constrained by limited execution memory, sometimes minimal storage, and the lack of direct connections to the Internet. In addition, software updates need to be applied across a distributed system of automotive devices, which are designed and serviced by different suppliers, and the probability of bricking the vehicle (i.e., rendering it inoperable) must be reduced to negligible levels.

What are the threats and why isn't secure transport & code signing enough?

Increasingly, software updates for automotive ECUs are done using "software-over-the-air" (SOTA) strategies [8], in which revised versions of software programs are sent to vehicles over the Internet. This strategy is growing in popularity because it is a much quicker and more cost-effective delivery mechanism than traditional methods, such as distributing flash drives or requiring customers to bring in their vehicles for servicing. Furthermore, SOTA promises much higher penetration rates for essential updates. Established OEMs are recognizing the potential of this strategy. In May 2019, GM announced that the 2020 Cadillac CT5 sedan would feature an over-the-air software update mechanism, and that

such systems will be standard on all GM vehicles in the next four years [9]. In the same year, Ford's website announced that, beginning in 2020, it would begin equipping most redesigned vehicles in the U.S. with advanced over-the-air software update mechanisms, starting with the all-electric Mustang Mach-E [10]. Moreover, in 2021, Toyota and Nissan also begin rolling out OTA software update features, Toyota in the new Teammate driver assistance system for the Lexus LS, which will allow for hands-off driving on the highway. and Nissan in additions and adjustments to driving modes that improve the responsiveness of the steering wheel or reduce power consumption in the 2021 Ariya [11].

Unfortunately, connecting ECUs directly to the Internet exposes them to a wide range of attacks, some of which, like SolarWinds, introduce malware masquerading as legitimate software updates. The consequences of such attacks could be costly indeed, not only in terms of recalls or lost sales, but also, potentially, in loss of life.

The classes of attacks that an automotive software security system needs to defend against fall into four categories, presented here in order of increasing severity.

Read updates: The goal here is intellectual property theft, so these attackers aim to read the contents of software updates. This is generally achieved using an eavesdropping attack, where attackers read unencrypted software updates sent to the vehicles from a repository–or a server containing relevant metadata about software images, and sometimes the software images themselves–maintained by the OEM or a supplier.

Deny updates: In this class of attacks, the goal is to deny access to software updates so vehicles cannot fix software defects, including newly discovered vulnerabilities. These attacks include:

- *Drop-request attack:* blocks network traffic outside or inside the vehicle to prevent an ECU from receiving any software updates.
- *Slow retrieval attack:* slows delivery time of software updates to ECUs so that a known security vulnerability can be exploited before a corrective patch is received.
- *Freeze attack:* continues to send the last known software update to an ECU, even if a newer update exists.
- *Partial bundle installation attack:* allows only part of a software update to install by dropping traffic to selected ECUs.

Deny functionality: This class of attacks ups the threat ante a bit further by causing vehicles to fail to function in one of the following ways:

- *Rollback attack:* tricks an ECU into installing outdated software with known vulnerabilities.
- *Endless data attack:* causes an ECU to crash by sending it an infinite amount of data until it runs out of storage.

- *Mixed-bundles attack:* shuts down an ECU by causing it to install incompatible versions of software updates that must not be installed at the same time. Attackers can accomplish this by showing different software bundles to different ECUs at the same time.
- *Mix-and-match attack:* Like the mixed-bundles attack described above, this attack also causes ECUsto use arbitrary combinations of new versions of software updates. However, it is a more serious threat, as it proves that the attackers have abused repository keys to sign this software bundle. Thus, all the software updates provided can be completely arbitrary.

Control: The last and most severe method of attack is to force an ECU to install software of the attacker's choosing, thus entirely ceding control of that ECU. This means an attacker can arbitrarily modify the behavior of a vehicle by overwriting the existing software on an ECU with a malicious software program.

The most common defense against these attacks is to utilize a secure transport communication protocol (e.g., HyperText Transport Protocol Security - [HTTPS]) and basic code signing (i.e., using a single key to sign the code). However, a secure over-the-air software update system must do more. A single cryptographic signature does provide some protection against an arbitrary software attack, but is not enough on its own to defend against the full range of attacks described above. In addition, the single signing key itself is a single point of failure for the system. If an attacker gets control of this common signing key, then they have full control of all updatable ECUs.

Reliable over-the-air software updates require a solution that addresses all of the above attacks and is also compromise resilient. In a compromise resilient system, the security of the entire system does not disintegrate if a hacker obtains control of a repository or a signing key. In addition, compromise resilient systems like Uptane have built-in mechanisms to make a quicker recovery from such an attack.

What does Uptane do differently?

Uptane never conflicts with best practices, but rather expands and/or improves on best practices for highly resilient software. Uptane is designed to work *with* existing software management systems rather than to *replace* them, and does so by adding a more realistic approach to existing software update strategies. As stated above, Uptane acknowledges that compromise is not a matter of *if,* but of *when.* Attacks **will** occur and the best defense is a strategy that can isolate damage and limit exposure. The building blocks for this approach rest on four design principles.

- *Separation of trust:* distribute responsibility for the signing of metadata so if one signing key is compromised, then it will not affect other parts of the system.
- *Threshold signatures:* require that at least a minimum number of signatures must be gathered to attest to the authenticity of a file before the software

update can be downloaded.

- *Explicit and implicit revocation of keys:* provide a mechanism for replacing compromised keys so that malevolent parties cannot continue signing metadata to authenticate malware, and ensure that keys have reasonable validity periods (i.e., the same keys are not used forever).
- *Keeping the most vulnerable keys offline:* mandate that certain signing keys must always be unavailable to online services, thus making those signing keys harder to steal or compromise.

These principles are extracted from an established standard called The Update Framework (TUF) [12], a flexible framework and specification that has proven successful for securing software update systems and software repositories. Yet, researchers realized that some changes would be needed to adapt these principles for automotive ECUs.

The first change is to add a second repository to divide labor and responsibility for different aspects of the software update verification process. The Image repository holds an accurate inventory of all the software images currently on all ECUs on all vehicles maintained by an OEM, and the corresponding metadata. This Image repository uses offline keys to sign its own metadata, making it much harder for attackers to substitute compromised software images. The Director repository, which instructs vehicles what software updates should be installed next, uses online keys to sign its own metadata, allowing for easier and faster software update compaigns. By combining these two repositories, an OEM can provide both customization and strong security for the ECUs on their vehicles.

The second change made to the basic TUF design has to do with the way Uptane verifies software updates. In the verification step, the ECU determines if a file is safe to download by checking its accompanying metadata. A given ECU can be designed to use one of two different verification strategies–*full* or *partial*–depending on its processing power and other resources. Full verification requires checking that the hashes and sizes of software updates in the signed metadata match the hashes and sizes stored on the Image repository. Partial verification only requires a check of the signature on a subset of metadata received from the Director repository.

Basic Uptane Design

The diagram above illustrates how the checks and balances of the Uptane system works. The connected components on the right hand side of the diagram are on the vehicle, while the components on the left hand-side represent the repositories. The Image repository can be thought of as an authoritative source of information about software images, and the keeper of every image currently deployed by the OEM or supplier, along with the metadata files that prove their authenticity. The Director repository determines what software updates should be distributed to each ECU.

In the first step in the software update process, the vehicle sends its vehicle

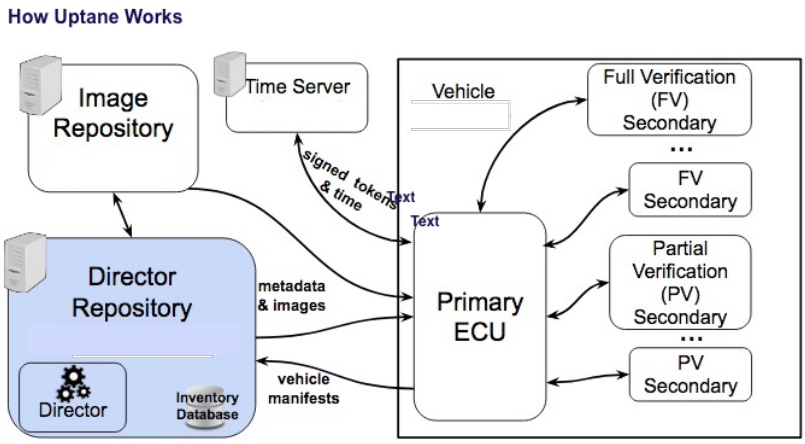

Figure 1: Uptane Process

version manifest for all installed ECUs to the Director repository. This vehicle version manifest contains signed information about existing, installed software images. Using this input, the Director chooses which software images should be installed next. The metadata and software images are distributed to the vehicle, which then runs a verification process. The diagram above shows a Primary ECU that is connected to a number of Secondary ECUs. The Primary ECU downloads software images and metadata from the Director and Image repositories, and then shares them with one or more Secondary ECUs on the same vehicle. ECUs are classified in terms of their access to storage space, memory, a power supply, and (optionally) a direct Internet connection. The chosen form of verification—*full* or *partial*—is also based on the available resources of the ECU, as well as the criticality of the software update (for safety and/or security). If the verification process finds no issues, then the software image can be flashed to the ECU, and the vehicle version manifest is updated.

Full verification provides better protection for those ECUs that have the memory and storage resources to conduct the more complex procedure. Yet, even the least capable ECUs can achieve basic protection by the less resource-intensive partial verification method. Thus, the security of the system as a whole is improved.

How Uptane continues to evolve to meet a changing marketplace

Uptane is very much a living and evolving technology. Over the past five years,the Uptane community has standardized this technology. A multinational

team of academic, government, and automotive industry collaborators have also identified key principles to guide the continuing development of the Uptane specification. Here are the key principles the Uptane community has embraced that will enable the technology to evolve in the years ahead.

Agility: Agility in this sense refers to staying ahead of the curve on emerging trends in the automotive software industry and being able to respond accordingly. The Uptane project benefits here from the wide variety of subject matter experts that continue to contribute to both the Standard [13] governing the technology and the Deployment Best Practices, which compile real-world variations from OEMs and first-tier suppliers [14]. Also, since Uptane is an open source project, anyone can review the Standard or the Deployment Best Practices and propose changes. Thus, the Uptane technology benefits from ongoing feedback derived from real-world implementations.

Ease of adoption: As mentioned earlier, Uptane never conflicts with deployment best practices and one component of that commitment is that adopters should not need to reconfigure their software update systems just to integrate the Uptane framework. For this reason, the Uptane community made a decision early on not to specify data binding formats or other protocols, operations, usage, and formats. Instead, the Uptane Standards team developed a new approach for specifying Uptane that separates interoperability aspects, such as backwards compatibility, localization, and deployment, from those aspects essential to reliability, security, and functionality. The latter group of aspects, which constitute the actual Uptane Standard, make up the baseline layer for instructions, while all the elements required for interoperability are specified in a second layer, known as an Uptane Protocols, Operations, Usage, and Formats (POUF) document. By giving implementers the option to create a POUF [15], Uptane technology can be adapted to the constraints of existing implementations, without requiring extensive modifications. The POUF concept also makes it easier to add suppliers to the software update ecosystem as needed, without having to share supplier proprietary designs.

Awareness of evolving regulations and standards: As government and industry begin to address the need for improved automotive security, the Uptane project is seeking to stay in alignment with emerging regulations and international standards governing over-the-air software updates and other aspects of cybersecurity important to the automotive space. This is achieved by leveraging insights from industry experts through ongoing revisions to the Uptane Standard, and by continuing to encourage all stakeholders in the automotive industry to provide feedback through an open-source Uptane forum.

Starting conversations about the challenges ahead

As new vehicles edge ever closer to the designs envisioned in novels and movies in the mid-20th century, there will be new cybersecurity questions to address. Currently, the Uptane project is opening conversation on three such issues. The

industry's response to these issues is still evolving so the Uptane project may not need to provide solutions in the immediate future. However, through Uptane whitepapers, we can frame the key questions that need to be resolved as our work moves forward. Look for new whitepapers in the coming months and years to address the following issues.

Allowing access to ECUs for emergency updates from federal/state/local governments

As SOTA delivery of automotive software updates becomes more sophisticated, government agencies and regulatory bodies, such as the U.S. Department of Transportation or its state or local equivalents, the Department of Homeland Security, or the Federal Emergency Management Agency, may require automakers to grant them access to vehicles in emergency situatiions. Such updates might cover things like changed rules of the road (across a state or country border), emergency routing (e.g., from US FEMA or US DOT), normal traffic updates (which would come from US DOT or their counterparts in other nations), and reliable maps (not from Google, et al). Though there have been no specific calls for such access as of yet, there has been discussion of these types of scenarios in other standards groups. Furthermore, all of these updates happen today on cellphones, and on infotainment units of vehicles with cellular connectivity. Accommodating this government access to vehicles will require some further discussion about how an Uptane implementation is configured, particularly about how to prioritize delegations, and perhaps also support a dual Director repository set-up.

Security issues related to the use of aftermarket materials

Aftermarket automotive supply companies create new parts for additional automotive functionality, and also refurbish and reuse parts following end-of-life support from OEMs. Thus, aftermarket suppliers are introducing ECUs to vehicles over which the OEM has no control. In addition, because aftermarket suppliers typically do not have access to original OEM designs, they often must reverse engineer the parts to figure out how they work. Such an approach perhaps keeps these aftermarket suppliers from being able to discover all relevant design information about the ECUs.

Reflecting the many issues this topic encompasses, a future whitepaper will try to effectively frame a number of questions, and describe whatever answers may be currently available. These questions will include:

- How can we deal with aftermarket ECUs that do not have their own Primary ECU?
- Can these aftermarket ECUs leverage an OEM's Director and Image repositories?
- If an aftermarket ECU does have its own Primary ECU, then is it capable of controlling a mutually exclusive set of ECUs?
- Could ownership of the Director be delegated to a third party or owner?

Alignment with government and industry regulations

The Uptane project continues to monitor evolving regulations and standards. While our primary concern is ensuring these requirements and recommendations are reflected in our Uptane Standard and Deployment Best Practices, a future whitepaper will address how the design and implementation of Uptane "dove-tails" with relevant standards, such as:

- Autosar Update Configuration Module (UCM)
- UNECE WP29 R155 & R156 cybersecurity and software update regulations
- ISO/SAE 21434 Road Vehicles: Cybersecurity Engineering
- ISO 24089 Road Vehicles: Software Updates
- SAE J3101 Hardware Protected Security Environment
- TCG (Trusted Computing Group) hardware security and remote attestation specs
- IETF SUIT (Software Updates for IoT Devices) specs

Learn More

The best place to learn more about Uptane is to go to our website [16]. Here you can read more about the specification, review the current version of the Uptane Standard for Design and Implementation and the Deployment Best Practices documents, as well as previous and future conference presentations, testing information, and other data. We welcome questions, feedback, and suggestions on these materials, the website, or any other aspect of the Uptane project. Feel free to raise issues on GitHub or email feedback to jcappos@nyu.edu.

Anyone in the automotive industry, open source community, or security community is welcome to join the Uptane Forum. This is a fairly low volume mailing list and is used to disseminate large news items, or to plan in-person Uptane workshops. The Uptane standardization initiative is discussed on the Uptane Standard mailing list, to coordinate the Uptane standardization effort. To be added to either of these mailing lists, send an email to lad278@nyu.edu.

References cited

1. "Sally (Short story)," *Wikipedia.* https://en.wikipedia.org/wiki/Sally_(short_story). 22 February 2021. Accessed 14 June 2021.
2. "How Much Code? Cars," *The Code Institute.* https://codeinstitute.net/blog/much-code-cars. Accessed 14 June 2021.
3. Greenberg, Andy. "Tesla Responds to Chinese Hack With a Major Security Upgrade," *Wired.* https://www.wired.com/2016/09/tesla-responds-chinese-hack-major-security-upgrade/. 27 September 2016. Accessed 14 June 2021.
4. O'Kane, Sean. "Chrysler's over-the-air update fiasco is limited to the Northeast, but customers are still waiting for a fix," *The Verge.*

https://www.theverge.com/2018/2/14/17013016/fiat-chrysler-ota-update-problem-jeep. 14 February 2018. Accessed 14 June 2021.
5. Upstream Security. *2020 Global Automotive Cybersecurity Report.* https://upstream.auto/upstream-security-global-automotive-cybersecurity-report-2020/. Accessed 14 June 2021.
6. "2020 United States federal government data breach," *Wikipedia.* https://en.wikipedia.org/wiki/2020_United_States_federal_government_data_breach. 12 June 2021. Accessed 14 June 2021.
7. Howley, Daniel. "Why Russia's massive cyberattack is especially insidious," *Yahoo Finance.* https://au.finance.yahoo.com/news/why-russias-massive-cyberattack-is-especially-insidious-222912267.html?___s=p54njaazgqic1gqfruk3. 19 December 2020. Accessed 14 June 2021.
8. Juliussen, Egil. "Remote software update: Future growth business," *IHS Markit.* https://ihsmarkit.com/research-analysis/remote-software-update-future-growth-business.html. 13 January 2015. Accessed 14 June 2021.
9. Hawkins, Andrew J. "GM's new 'digital nerve system' will enable over-the-air software updates on all vehicles," *The Verge.* https://www.theverge.com/2019/5/21/18633000/gm-ota-software-updates-digital-platform-reuss. 21 May 2019. Accessed 14 June 2021.
10. "No more FOMO: New Ford over-the-air updates help Mustang Mach-E get even better with time–Without leaving home," Ford Media Center. https://media.ford.com/content/fordmedia/fna/us/en/news/2020/05/12/new-ford-over-the-air-updates-mustang-mach-e.html. 12 May 2020. Accessed 14 June 2021.
11. Oshikiri, Tomoyoshi. "Toyota and Nissan to upgrade driving functions remotely," *Nikkei Asia.* https://asia.nikkei.com/Business/Automobiles/Toyota-and-Nissan-to-upgrade-driving-functions-remotely. 9 February 2021. Accessed 14 June 2021.
12. *The Update Framework Website.* https://theupdateframework.io/. Accessed 14 June 2021.
13. *Uptane Standard for Design and Implementation.* https://uptane.github.io/papers/uptane-standard.1.1.0.html. 8 January 2021. Accessed 14 June 2021.
14. *Uptane Deployment Best Practices.* https://uptane.github.io/papers/uptane-deployment-best-practices-1.1.0.html. 8 January 2021. Accessed 14 June 2021.
15. Moore, Marina, McDonald, Ira, Weimerskirch, André, Awwad, Sebastien, DeLong, Lois Anne, and Cappos, Justin. "Using a Dual-Layer Specification to Offer Selective Interoperability for Uptane." *ESCAR USA 2020 Special Issue, SAE Int. J. Transp. Cyber. & Privacy* 2(2):113-129, 2019. https://ssl.engineering.nyu.edu/papers/moore_pouf_2020.pdf.
16. *Uptane Website.* https://uptane.github.io/. Accessed 14 June 2021.