UPTANE

Security and Customizability of Software Updates for Vehicles

Trishank Karthik Kuppusamy,
Lois Anne DeLong,
and Justin Cappos

A widely accepted premise is that complex software frequently contains bugs that can be remotely exploited by attackers. When this software is on an electronic control unit (ECU) in a vehicle, exploitation of these bugs can have life or death consequences. Since software for vehicles is likely to proliferate and grow more complex in time, the number of exploitable vulnerabilities will increase. As a result, manufacturers are keenly aware of the need to quickly and efficiently deploy updates so that software vulnerabilities can be remedied as soon as possible.

However, existing software-update security systems are not compromise resilient; if an attacker breaks into

Digital Object Identifier 10.1109/MVT.2017.2778751
Date of publication: 1 February 2018

1556-6072/18©2018IEEE

PHOTO: ©ISTOCKPHOTO.COM/SJO; GEAR: ©ISTOCKPHOTO.COM/RIBKHAN

any portion of an automobile's infrastructure, they could compromise numerous vehicles. The industry needs to dynamically choose updates for vehicles based on fresh information, forcing manufacturers to choose existing systems that sign updates using a key stored on the server. Attackers who compromise the repository can abuse this online key and cause malicious software to be installed on vehicles.

In this article we discuss Uptane, the first, to our knowledge, compromise-resilient software update security system designed specifically for vehicles. It is designed to make obtaining all the pieces required to control a vehicle extremely difficult for attackers.

Risks and Benefits of Connected Cars

Vehicles are being connected to the Internet on a more frequent basis, providing owners with many benefits, e.g., facilitating infotainment systems, updating navigation maps, and enabling emergency response systems. However, the downside of connecting vehicles to the Internet is that software vulnerabilities are exposed. If these vulnerabilities are remotely exploited, it could jeopardize vehicles and prove fatal to passengers and drivers. It is therefore critical for automobile manufacturers to deploy software updates over-the-air (SOTA) as soon as possible. Using SOTA, manufacturers can add to and improve on existing features, and, most importantly, fix software bugs on ECUs without incurring the high costs traditionally associated with a manual recall.

The need for secure SOTA systems was demonstrated in 2015, when security researchers remotely exploited a Jeep's software system by commandeering its dashboard functions, steering, transmission, and brakes [1]. This incident illustrates several security issues. Chrysler, the manufacturer of Jeep, was forced to recall 1.4 million automobiles because presumably there was no SOTA facility even though the automobiles were connected to the Internet using a Sprint cellular network. Chrysler sent a Universal Serial Bus (USB) drive with a software update to every affected owner and advised them to install it through the vehicle's dashboard. Chrysler's researchers considered several important security issues.

First, it proves that ECUs are just as vulnerable to attack by malicious parties as any other smart device. To combat these security shortcomings, auto manufacturers must recognize that they are not immune

Since software for vehicles is likely to grow larger and more complex over time, the number of exploitable vulnerabilities will likely increase.

to attacks; they must adapt their policies, including those governing how they address software updates, accordingly, or face devastating consequences. In addition to the damage done to their reputations, both Chrysler and Sprint saw their stock values drop by roughly 2 and 8%, respectively, after this widely publicized simulated hack [2].

Second, it demonstrates that caution must be exercised when designing a software update system. Major repositories or servers used to host and distribute software updates by companies, including Adobe, Apache, Debian, Fedora, FreeBSD, Gentoo, GitHub, GNU Savannah, Linux, Microsoft, npm, Opera, PHP, RedHat, RubyGems, SourceForge, and WordPress, have been compromised, some on multiple occasions [3]–[26]. These lapses in security occurred because of the use of off-the-shelf security solutions such as secure sockets layer (SSL)/transport layer security (TLS), which are known to be inadequate against practical threats in this domain [27], [28].

When attackers compromise a repository, they also make the signing key for updates vulnerable. If this happens, the impact can be significant because attackers can then distribute and install malware on unsuspecting vehicles. In 2013, an attack exploited flaws in a SOTA mechanism to launch attacks against financial and government institutions in South Korea. Government officials estimated that the attack cost the economy US$750 billion total [29], [30]. If these attacks are directed at vehicles, the results could have included loss of life.

The third point that Chrysler's security breach showed is that unique consideration must be given to any change in a security system's design. One such consideration is the need to achieve both security and customizability of updates. Another issue is that ECUs differ greatly in terms of resources, i.e., computing power, memory, and access to the Internet. While some ECUs are powerful enough to check a large number of signatures in a certain time frame, others may only be able to check one signature in the same amount of time. Finally, a system must also be flexible enough to be applied to a wide variety of deployment scenarios.

Collectively, these three points were the inspiration for and the impetus behind the design of Uptane. Uptane was designed in collaboration with the manufacturers and suppliers responsible for 78% of the vehicles on U.S. roads, as well as with governmental regulators. It was cooperatively developed by researchers from the Tandon School of Engineering at New York University (NYU), the University of Michigan Transportation Research Institute (UMTRI), and the Southwest Research Institute (SWRI). In January 2017, Uptane was formally introduced to the automotive community at events in Ann Arbor, Michigan, and Brooklyn, New York [31], [32].

Since its introduction, Uptane has been implemented by several automotive suppliers, e.g., Lear Corporation, OTAinfo, and Advanced Telematic Systems [33], the latter of which is the first European company to utilize the technology (and discussions continue with a number of other manufacturers). The magazine, *Popular Science,* recently named Uptane one of 2017's "most important innovations in security" [34].

Updating ECU Software

Though an original equipment manufacturer (OEM), e.g., Mercedes-Benz or Volkswagen, decides which ECUs should reside in a vehicle, automotive ECUs are typically produced by third-party suppliers, such as DENSO or Rolls-Royce. Suppliers are also responsible for developing, maintaining, and updating the software for ECUs, which OEMs distribute.

An OEM uses a repository to distribute software updates to ECUs in the form of images and metadata. An image is an archive of code and/or data that enables an ECU to function. A metadata file contains information, such as hash values and lengths, for verifying the authenticity of an image, as well other metadata files. Figure 1 shows a signed metadata file. Although it is reasonable to expect metadata and images to be delivered over-the-air from the repository to the vehicle, Uptane is designed to be completely agnostic with respect to the transport mechanism, i.e., updates may be delivered using a cellular connection, USB flash drive, or a laptop connected to the on-board diagnostics (OBD)-II port.

Ideally, all metadata is signed using offline or private keys that are not accessible from the repository, so attackers cannot compromise images without being detected. To the best of our knowledge, OEMs do not typically, in practice, sign metadata. This indicates that images can be reflashed to ECUs if: 1) the attackers have a man-in-the-middle connection to ECUs, and 2) they know the fixed challenge-response algorithms used to unlock ECUs [35], [36]. Though manufacturers may believe their fixed algorithms are exclusive, the automotive community may, nevertheless find them [35], [36]. For example, until security researchers exploited a wireless connection to overwrite software on its ECUs, allowing the researchers to remotely obtain physical control of the vehicles, it appears that Tesla, Inc.'s images were unsigned [37]. However, while it is important to sign metadata, ideally by using offline keys, in practice, the problem appears to be more nuanced.

Limitations of Existing SOTA Systems

The problem with existing SOTA systems is that they are inadequate when dealing with the operations of the automotive industry. They provide either security or customizability, but they cannot provide both. A secure system uses an offline key to sign all metadata. Typically, OEMs would use the Pretty Good Privacy (PGP)/GNU Privacy Guard (GPG) or RSA cryptosystem for this purpose. While these systems provide some compromise resilience, the downside is that the OEM loses the ability to dynamically choose different updates for different vehicles depending on context and fresh information. This is due to offline keys being required to sign all metadata, which can become expensive when updates are random or frequent, thus necessitating human intervention. Signing with offline keys requires human intervention, which can become expensive if updates are frequent.

One solution to this might be to sign different updates for different vehicles ahead of time, given what ECUs have already been installed on a vehicle; however, this assumes that there are only a few possibilities for what ECUs have installed. Another downside is that this system provides only a weak form of compromise resilience; a compromise of the single signing key is enough to enable attackers to install malware on all vehicles maintained by the OEM.

The second system uses a single online key that is accessible from the repository. Typically, OEMs would use the SSL/TLS or Client-Update Protocol transport mechanism for this purpose, where all updates are dynamically encrypted in transit. The upside is that automated processes on the repository can choose different updates for different vehicles. The drawback is that, if compromised, attackers can use this online key to install malware on all vehicles. In this case, the presence of a hardware security module (HSM) would not help because attackers could use the HSM to sign new metadata for malicious images, even if they do not have direct access to the private key itself.

Uptane: Security and Customizability

The auto industry's need to find an update system that does not require choosing between security and customizability was the motivation for creating Uptane, the first software update security system for automobiles that provides both. The key to its design is the use of two different repositories.

Uptane uses at least six design principles to provide compromise resilience. The first is a separation of duties, with different types of metadata being signed by different roles so the impact of a key compromise will be limited to only the responsibilities assigned to that role. There are four top-level roles on a repository, as illustrated in Figure 2 and summarized in Figure 3. The second principle is to require a threshold number m of signatures from n independent keys to sign a metadata file. This is an application of the two-man rule: the larger this threshold number, the more difficult it should be for attackers to compromise keys and sign a new metadata file. The third principle is implementing a process by which keys are revoked if they are compromised. Keys can be revoked either explicitly or implicitly—the former, by publishing new metadata, and the latter by adding a signed expiration time stamp to a metadata file. The fourth principle is to further minimize risk by using offline keys for high-value roles. Using offline keys for the root and targets roles, for which a compromise could mean the installation of malicious images, can provide additional security. The fifth principle is a selective delegation of trust, i.e., developers are trusted with signing for only a subset of images so that the key compromise of a single developer does not affect all ECUs. Delegations are also useful for distributing, revoking, and replacing public keys belonging to suppliers and their developers. The sixth principle

In 2015, security researchers showed how to remotely exploit Jeeps and commandeer their dashboard functions, steering, transmission, and brakes.


```
{
 "signatures": [
  {
   "keyid": "ce3e02e72980b09ca6f5efa68197130b381921e5d0
             675e2e0c8f3c47e0626bba",
   "method": "ed25519",
   "sig": "9095bf34b0cbf9790465c0956810cb3729bc96beed
           8ee7e42d98997b1e8ec0a6780e57556570687df4a5
           59d563a569258eac15fd9832b2e8e6d048cc32b5f603"
  }
 ],
 "signed": {
  "_type": "Targets",
  "delegations": {
   "keys": {},
   "roles": []
  },
  "expires": "2030-01-01T00:00:00Z",
  "targets": {
   "supplier-A-ECU-B.img": {
    "hashes": {
     "sha256": "141f740f53781d1ca54b8a50af22cbf74e44c2
                1a998fa2a8a05aaac2c002886b"
    },
    "length": 28
   }
  },
  "version": 1
 }
}
```


Figure 1 An example of a signed metadata file.

AN OEM USES A REPOSITORY TO DISTRIBUTE SOFTWARE UPDATES TO ECUS IN THE FORM OF IMAGES AND SIGNED METADATA.

is to use a diversity of signing and hashing algorithms, which allows for surviving a compromise of all but one of these algorithms. Using these design principles, an OEM maintains two repositories: one for security and one for customizability.

The image repository serves images for all ECUs on vehicles maintained by the OE and holds metadata that can verify their authenticity. The OE uses offline keys to sign this metadata. Images are delegated to their respective tier-one suppliers so that the impact of a key compromise is limited to the affected supplier. If a tier-one supplier does not sign its images, then the OEM signs on its behalf. This repository provides an immutable source of information about images that attackers cannot modify without having compromised offline keys.

The director repository controls the images that should be installed next on any given ECU. When a vehicle's software needs to be updated, it first provides the director repository its vehicle version manifest, or the signed information about existing images. According to this manifest, automated processes running the director repository perform dependency resolution [40], choosing which images should be installed next.

The director repository uses an independent source of information about which images are available from the image repository to ensure that compromise of the image repository does not affect the director repository. The director repository uses online keys to sign instructions using the targets, snapshot, and time stamp metadata. In addition to the image and director repositories, the OEM may also maintain a time server since ECUs typically do not have real-time clocks. Accurate time stamps help ECUs avoid freeze attacks, where attackers continually replay previously downloaded metadata.

Conceptually, there are two types of ECUs, one being more powerful than the other. In this context, *powerful* is defined as not only having more speed and/or memory, but also a possible Internet connection. These powerful ECUs, also known as *primaries*, download and verify metadata and images before distributing them to secondaries. Secondaries double-check the metadata and images distributed by primaries. There are two types of metadata verification depending on an ECU's security and cost requirements. With full verification, as displayed in Figure 4, images chosen for installation by the director repository are checked to see if they match the corresponding images available on the repository. This entails checking that the hashes and sizes of the images in the target's metadata

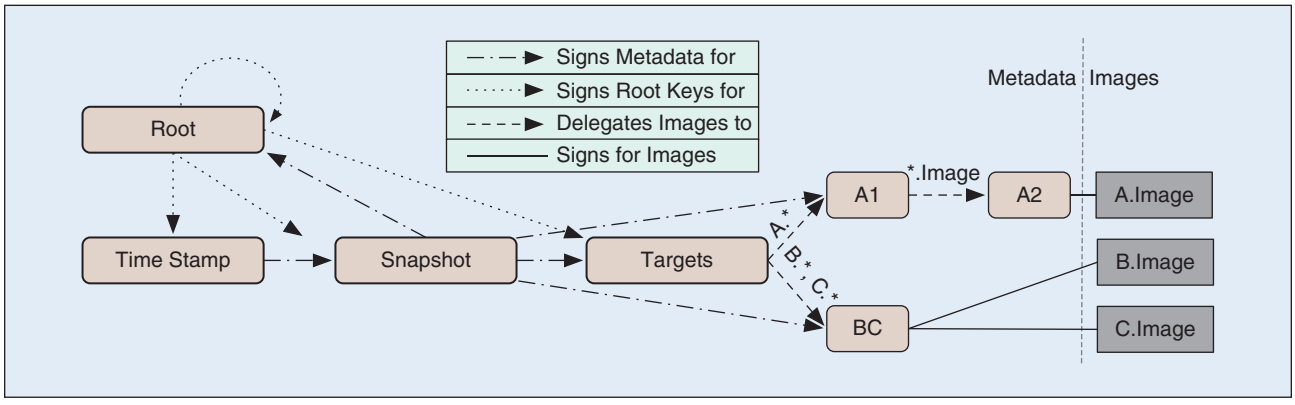

FIGURE 2 The separation of duties between signing metadata roles [38], [39].

Role	Responsibilities
Root	The root role is the locus of trust. It indicates which keys are authorized for the targets, snapshot, and time stamp roles. It also lists the keys for the root role itself.
Targets	The targets role provides crucial metadata about images, such as their hashes and lengths. This role may delegate the signing of images to their respective suppliers.
Snapshot	The snapshot role indicates the latest versions of all metadata on the repository. This prevents an ECU from installing an outdated image.
Time Stamp	The time stamp role is responsible for indicating if images or metadata have changed.

FIGURE 3 The list of responsibilities for top-level roles [38], [39].

signed by the director repository match the hashes and sizes of the same images in the target's metadata on the image repository. To prevent security attacks, primaries always perform full verification on behalf of secondaries, while secondaries perform either full or partial verification, checking only the signature on the target's metadata file from the director repository. This simple security check is designed for computationally weak ECUs. Safety-critical ECUs, ones where a compromise can jeopardize the safety of the vehicle, should use full verification; all other ECUs should use partial verification. ECUs that perform neither full nor partial verification should not be updated via SOTA.

Figure 5 offers a thumbnail sketch of the security effectiveness of Uptane, based on the type of attack and the type of ECUs compromised. When there are only man-in-the-middle attacks, but no key compromise, then attackers do not pose a serious threat. If the attackers compromise the director repository, the following two scenarios are possible.

In the first scenario, where attackers have not compromised primaries, the worst-case scenario would mean that ECUs are unable to work together due to attackers being able to control which images are to be installed on which ECUs. However, this action can be limited by suppliers who include metadata that prevent ECUs from installing incompatible or conflicting images. Regardless, attackers will not be able to install malicious images because primaries always perform full verification on behalf of secondaries.

The situation becomes a bit more critical during the second scenario where attackers have also compromised primaries. In this case, primaries may no longer be able

Uptane uses at least six design principles to provide compromise resilience.

to perform full verification on behalf of secondaries, so it may be possible for malicious images to be installed on partial verification ECUs. However, in this scenario, attackers would still be unable to install malicious images on full verification ECUs.

The latter scenario would only be possible under the following conditions: 1) if attackers compromise the offline keys used by the tier-one supplier who maintains

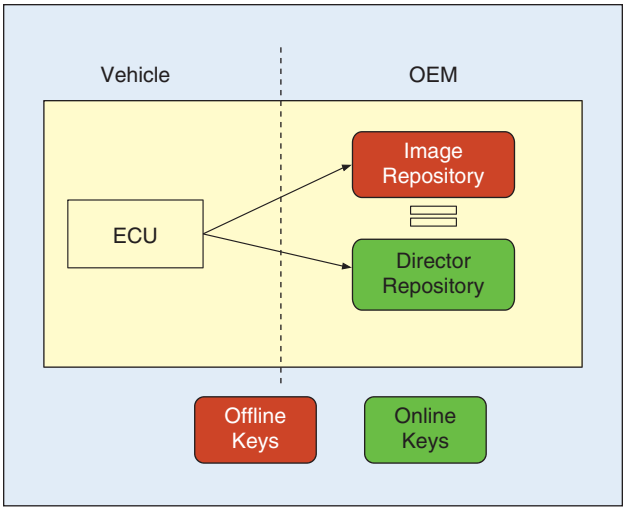

Figure 4 Uptane uses two repositories to provide OEMs with both security and customizability [38], [39].

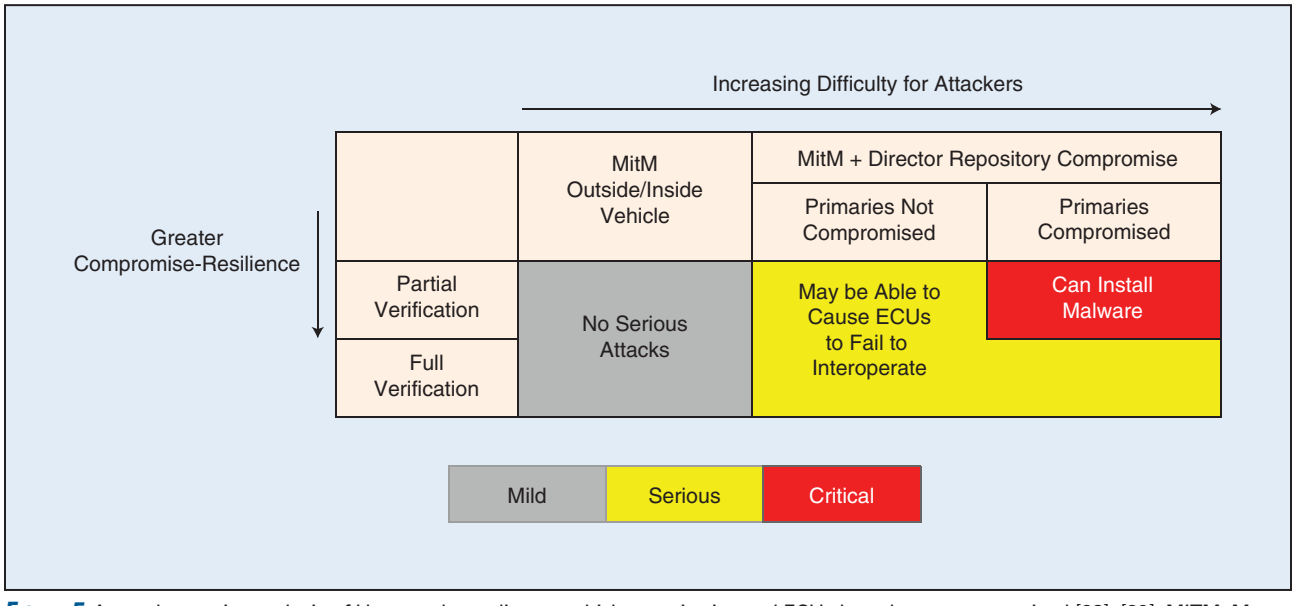

Figure 5 A rough security analysis of Uptane, depending on which repositories and ECUs have been compromised [38], [39]. MiTM: Man-in-the-Middle.

Uptane uses two repositories to separate security and customizability.

the ECU, and 2) if they upload signed metadata and malicious images to the OEM, who would then update both the image and director repositories. However, note that if the OEM verifies new images from suppliers using out-of-band processes, e.g., a video conference call where hashes are verbally verified, then they can detect malicious images and infer that a key compromise has occurred.

As stated previously, Uptane uses two repositories to separate security and customizability. It offers basic security guarantees for all ECUs and greater compromise-resilience for ECUs that can afford additional computation and memory.

Conclusions

Uptane provides a safe software update security strategy for automobiles that has proven successful in other domains. We share the automotive industry's commitment to creating a product that minimizes safety risk for those who use it. We recognize that subjecting the system to a critical, rigorous, and open review has long been the most reliable way to guarantee its security. Accordingly, after introducing Uptane in January 2017, we extended an invitation to the security community to find any design flaws before the black-hat hackers use them against us. This process has been underway for more than a year with, to date, no major flaws reported. Comments about our system may be left on our Google Docs, and any issues should be reported by sending pull requests on our GitHub projects, via our website at https://uptane.github.io.

As we head into our third year of iterative designs and modifications with the automotive industry, we strongly believe that Uptane is the most comprehensive and robust solution to securing SOTA on vehicles. It is our hope that, as the project continues to advance, others will join our efforts to solve this problem facing automobiles, as it is so critical to our national security. Because of this, we have made Uptane open source, royalty-free, and patent-unencumbered. The choice now is up to OEMs and suppliers, and we strongly urge them to carefully consider the security of the software update systems they might employ. Given the stakes, hoping for the best is not an acceptable strategy.

Acknowledgments

Uptane is supported by the U.S. Department of Homeland Security Grants D15PC00239 and D15PC00302. The views and conclusions contained herein are the authors' and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Department of Homeland Security or the U.S. government. Uptane is the result of collaboration between multiple research organizations, led by Justin Cappos (NYU), Sam Lauzon (UMTRI), and Cameron Mott (SWRI). We are grateful for the efforts of Akan Brown (NYU), Sebastien Awwad (NYU), Damon McCoy (NYU), Russ Bielawski (UMTRI), Sam Weber (NYU), John Liming (SWRI), and André Weimerskirch (UMTRI/Lear Corporation). We deeply thank contributors from various OEMs and suppliers who participated in our workshops and forums and who helped to iterate upon and improve Uptane.

Author Information

Trishank Karthik Kuppusamy (trishank@nyu.edu) received his Ph.D. degree in computer science from the Tandon School of Engineering at New York University in 2017. In the past, he worked on the research and development of The Update Framework (TUF), which is being integrated by Haskell, OCaml, Ruby, Rust, and Python, and has been used in production by LEAP, Flynn, VMware, Kolide, DigitalOcean, Cloudflare, CoreOS, and Docker. He led the research efforts for Uptane, a variant of TUF for use with automobiles. He is currently the chief security solutions engineer at Datadog.

Lois Anne DeLong (lad278@nyu.edu) received her B.A. degree in 1977 and her M.A. degree in 2008, both from New York University. She is a research associate and technical writer for the Secure Systems Lab at New York University's Tandon School of Engineering. She has served as a writer and editor for technical journals, and has also taught technical writing and basic composition courses.

Justin Cappos (jcappos@nyu.edu) is an assistant professor at the Tandon School of Engineering at New York University. His research interests include improving the security of real-world systems in a variety of practical applications. His research has led to the production of widely used software, including those of Docker, Git, Python, and most Linux distributions.

References

[1] A. Greenberg. (2015, July 24). After Jeep hack, Chrysler recalls 1.4M vehicles for bug fix. *Wired*. [Online]. Available: https://www.wired.com/2015/07/jeep-hack-chrysler-recalls-1-4m-vehicles-bug-fix/

[2] D. Goldman. (2015, July 24). Chrysler recalls 1.4 million hackable cars. CNN. [Online]. Available: http://money.cnn.com/2015/07/24/technology/chrysler-hack-recall/

[3] B. Arkin. (2012, Sept. 12). Adobe to revoke code signing certificate. Adobe. [Online]. Available: https://blogs.adobe.com/conversations/2012/09/adobe-to-revoke-code-signing-certificate.html

[4] Apache Infrastructure Team. (2009, Aug. 28). Apache.org incident report for 8/28/2009. [Online]. Available: https://blogs.apache.org/infra/entry/apache_org_downtime_report

[5] Apache Infrastructure Team. (2010, Apr. 09). Apache.org incident report for 04/09/2010. [Online]. Available: https://blogs.apache.org/infra/entry/apache_org_04_09_2010

[6] Debian. (2003, Dec. 2). Debian investigation report after server compromises. Debian. [Online]. Available: https://www.debian.org/News/2003/20031202

[7] Debian. (2012, July 25). Security breach on the Debian wiki 2012-07-25. Debian. [Online]. Available: https://wiki.debian.org/DebianWiki/SecurityIncident2012

[8] P. W. Frields. (2008, Aug. 8). Infrastructure report, 2008-08-22 UTC 1200. [Online]. Available: https://www.redhat.com/archives/fedora-announce-list/2008-August/msg00012.html

[9] J. K. Smith. (2011, Jan. 23) Security incident on Fedora infrastructure on 23 Jan 2011. Fedora Project. [Online]. Available: https://lists.fedoraproject.org/pipermail/announce/2011-January/002911.html

[10] The FreeBSD Project. (2012, Nov. 27). FreeBSD.org intrusion announced November 17th 2012. Free BSD. [Online]. Available: http://www.freebsd.org/news/2012-compromise.html

[11] Gentoo Linux. (2003, Dec. 2) Rotation server compromised. Gentoo. [Online]. Available: https://forums.gentoo.org/viewtopic.php

[12] GitHub, Inc. (2012, Mar. 4). Public key security vulnerability and mitigation. GitHub, Inc. [Online]. Available: https://github.com/blog/1068-public-key-security-vulnerability-and-mitigation

[13] B. M. Kuhn. (2003, Dec. 23). IMPORTANT: Information regarding savannah restoration for all users. [Online]. Available: https://savannah.gnu.org/forum/forum.php?forum_id=2752

[14] GNU Savannah. Compromise2010. Free Software Foundation. [Online]. Available: https://savannah.gnu.org/maintenance/Compromise2010/

[15] L. McVoy. (2003, Nov. 5). BK2CVS problem. [Online]. Available: http://lkml.iu.edu//hypermail/linux/kernel/0311.0/0621.html

[16] J. Corbet. (2011, Aug. 31). The cracking of Kernel.org. The Force Field. [Online]. Available: https://www.linuxfoundation.org/blog/thecrackingofkernelorg/

[17] Microsoft Corporation. (2012, June 6). Flame malware collision attack explained. Microsoft Corporation. [Online]. Available: https://blogs.technet.microsoft.com/srd/2012/06/06/flame-malware-collision-attack-explained/

[18] L. Voss. (2014, Mar. 21). Newly Paranoid Maintainers. NPMJS. [Online]. Available: http://blog.npmjs.org/post/80277229932/newly-paranoid-maintainers

[19] R. Naraine. (2013, June 26). Opera software hit by 'Infrastructure Attack;' Malware signed with stolen cert. Security Week. [Online]. Available: http://www.securityweek.com/opera-software-hit-infrastructure-attack-malware-signed-stolen-cert

[20] H. Magnusson. (2010, Dec. 24). The PHP project and code review. bjori doesn't blog. [Online]. Available: https://bjori.blogspot.com/2010/12/php-project-and-code-review.html

[21] PHP. (2011, Mar. 19). Php.net security notice. PHP. [Online]. Available: https://secure.php.net/archive/2011.php#id2011-03-19-1

[22] PHP. (2013, Oct. 24). A further update on php.net. PHP. [Online]. Available: https://secure.php.net/archive/2013.php#id2013-10-24-2

[23] Red Hat, Inc. (2008, Aug. 22). Infrastructure report, 2008-08-22 UTC 1200. [Online]. Available: https://rhn.redhat.com/errata/RHSA-2008-0855.html

[24] RubyGems. (2013, Jan. 13). Data verification. RubyGems. [Online]. Available: http://blog.rubygems.org/2013/01/31/data-verification.html

[25] SourceForge. (2012, Sept. 25). PhpMyAdmin corrupted copy on Korean mirror server. SourceForge. [Online]. Available: https://sourceforge.net/blog/phpmyadmin-back-door/

[26] M. Mullenweg. (2011, June 11). Passwords reset. Word Press. [Online]. Available: https://wordpress.org/news/2011/06/passwords-reset/

[27] T. K. Kuppusamy, S. Torres-Arias, V. Diaz, and J. Cappos, "Diplomat: Using delegations to protect community repositories," in *Proc. 13th USENIX Symp. Networked Systems Design Implementation (NSDI '16)*, Santa Clara, CA, 2016, pp. 567–581.

[28] T. K. Kuppusamy, V. Diaz, and J. Cappos, "Mercury: Bandwidth-effective prevention of rollback attacks against community repositories," in *Proc. 2017 USENIX Annu. Technical Conf. (ATC '17)*, Santa Clara, CA, pp. 673–688.

[29] A. Hern. (2013, Oct. 16). North Korean 'cyberwarfare' said to have cost South Korea £500m. *The Guardian*. [Online]. Available: https://www.theguardian.com/world/2013/oct/16/north-korean-cyberwarfare-south-korea

[30] Wolfram Alpha, LLC. (2013, Oct.). 470m gbp to usd in Oct. 2013. Wolfram Alpha. [Online]. Available: https://www.wolframalpha.com/input/?i=470m+gbp+to+usd+in+oct+2013

[31] J. Detsch. (2017, Jan. 18). Are software updates key to stopping criminal car hacks? *Christian Science Monitor*. [Online]. Available: https://www.csmonitor.com/World/Passcode/2017/0118/Are-software-updates-key-to-stopping-criminal-car-hacks

[32] L. Mathews. (2017, Jan. 19). Uptane will protect your connected car from hackers. *Forbes*. [Online]. Available: https://www.forbes.com/sites/leemathews/2017/01/19/uptane-will-protect-your-connected-car-from-hackers/#7a532f1019be

[33] E. Eitel. (2017, June 16). ATS is integrating the Uptane security framework for over-the-air software updates to connected vehicles. Advanced Telematic Systems. [Online]. Available: https://www.advancedtelematic.com/en/press-releases/ats-is-integrating-the-uptane-security-framework-for-over-the-air-software-updates-to-connected-vehicles.html

[34] K. D. Atherton and R. Feltman. (2017, Oct. 17). The year's most important innovations in security. *Popular Sci. Mag*. [Online]. Available: https://www.popsci.com/top-security-innovations-2017

[35] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," in *Proc. 31st IEEE Symp. Security Privacy,* Oakland, CA, 2010, pp. 447–462.

[36] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. 20th USENIX Security Symp.,* San Francisco, CA, 2011.

[37] A. Greenberg. (2016, Sept. 27). Tesla responds to Chinese hack with a major security upgrade. *Wired*. [Online]. Available: https://www.wired.com/2016/09/tesla-responds-chinese-hack-major-security-upgrade/

[38] T. K. Kuppusamy, A. Brown, S. Awwad, D. McCoy, R. Bielawski, S. Weber, J. Liming, C. Mott, S. Lauzon, A. Weimerskirch, and J. Cappos. (2017). SOTA #5: Uptane design overview. Tandon School of Engineering, New York University. [Online]. Available: https://docs.google.com/presentation/d/1R3jSDcqbqUlwJgbOLOKwHReoy2wnj8GrXlKCdcLNXAA/edit?usp=sharing

[39] T. K. Kuppusamy, L. A. DeLong, and J. Cappos. (2017). Securing software updates for automotives using Uptane. [Online]. 42*(2)*, pp. 63–67. Available: https://ssl.engineering.nyu.edu/papers/kuppusamy_login_2017.pdf

[40] D. Burrows. (2005). Modelling and resolving software dependencies. Debian. [Online]. Available: https://web.archive.org/web/*/https://people.debian.org/~dburrows/model.pdf

VT