Project Group 3

The Paramount Importance of Secure Coding for Application Security

Abstract:

A recent report from the UK's Huawei Cyber Security Evaluation Center (HCSEC) identified major security issues within Huawei's software engineering process. This paper aims to address how secure coding practices can help mitigate these issues. Starting with a brief overview of the history of secure coding practices and moving on to cover some of the more complex advancements in cyber security, the paper will provide insight into how these technologies can be used to combat problems such as those faced by Huawei.

Introduction:

Secure coding is the practice of developing software in a manner which protects it against the introduction of security vulnerabilities. Secure coding aims to protect applications, software, and users from hackers exploiting these vulnerabilities, and prevents occurrences such as denial of service to legitimate users, compromise of confidential material, and system damage, among many others. As technology and applications play an increasingly important and prevalent role in our everyday lives, secure coding has only become more vital to enable the security of our digital infrastructure and maintain protection for users personal data (such as online banking systems) as well as users' personal safety (such as autonomous vehicles).

While working to become a leading international 5G partner, Huawei, a multinational technology company headquartered in Shenzhen, China, established the HCSEC at the request of the UK government to evaluate Huawei's technology for vulnerabilities before it was deployed to telecom networks in the UK.[1] Huawei was founded in 1987 and as of 2019 has an annual revenue of over $120 billion USD and provides services to over 170 countries.[2] With such a broad, international impact, security risks in services provided by a company of this scale have the potential to be catastrophic, and thus secure coding practices become even more vital to preserve the security of customer data, networks, and devices. The goal of secure coding practices is to prevent breaches in security and preserve the integrity of client data and network usage, such as those present in the Huawei build.

Origins:

Security vulnerabilities that have the potential to adversely affect code include, but are in no means limited to, buffer overflow, code injection flaws, format-string attacks, and integer overflow. Secure coding practices first came into question in the late 1980s, after the advent of some of the first breaches of computer security. Commonly (although not indisputably) referenced as the first computer program to exploit flaws in a utility program, the Morris worm was released on November 2, 1988 and has since revolutionized how computer security has been addressed.[3]

At a high level, the worm functioned by establishing a socket to the infecting machine and exploiting vulnerabilities in several utilities such as *fingerd* (which allows users to obtain information about other users), *gets* (which stores input into a buffer without bounds checks), and *sendmail* (through which the worm propagated).[4] Ultimately, the worm could re-infect the same machine numerous times until it overran the entire system. Although there is not a consensus on the worms' malicious intent, it nevertheless led the Defense Advanced Research Projects Agency (DARPA), a branch of the US Department of Defense, to create the Computer

Emergency Reponse Team Coordination Center (CERT/CC) to facilitate partnerships with government, industry, and academia to improve the security of computer systems and networks.[5]

The foundation of these groups led to the consideration of steps to prevent such flaws, such as limiting unsafe uses of *scanf/fscanf/sscanf* and *strcat/strpy* calls to prevent buffer overflow.[6] Such practices are especially important as they can be used to set guidelines to educate computer scientists on safe programming practices. This was something that Huawei did do successfully via the creation of the HCSEC in response to criticisms – they developed a set of guidelines for secure programming practices to be followed by their computer scientists.[7] While the success of actually implementing those guidelines in practice is subject to scrutiny, having this basis of safe coding practices is certainly an essential step.

In 2004, in response to increasing attacks on consumer credit cards such as the Visa attacks in 2003, the Payment Card Industry (PCI) Standards Council created and released a set of secure coding guidelines known as the PCI Data Security Standards (PCI-DDS). These standards were created to hold those processing credit card payments to a minimum set of security standards to protect consumers. Some of the main features safeguards include installation/maintenance of firewall configurations, encrypted transmission of cardholder data via P2P, restriction of cardholder data on a need-to-know basis, and required utilization of strong password protection on local hardware.[8]

Despite these standards, credit card attacks continue to occur, including the November 2014 attack on Sony Pictures Entertainment where confidential client and company information were compromised. Several lessons were learned as key takeaway messages from the Sony hack – the foremost is the importance of requiring two-factor authentication (2FA) for access to protected data, which was already in use for several years by most of Sony's major competitors, including Microsoft.[9] Had 2FA been used, it is unlikely that the stolen credentials would have provided adequate access to the hackers. Another lessons learned was the importance of separate storage of passwords and password-protected files. Although the majority of Sony's sensitive files were password protected, the passwords were stored in the same location, effectively rendering this protection useless.[10]

The examples above are intended to provide a brief snapshot into the history of secure coding practices, but are by no means exhaustive. Nevertheless, these events have led to the creation of widely accepted coding practices that have been adopted over the last 20-30 years to prevent such security breaches. In addition, they have provided the necessary foundation for the state of the art advancements in the field addressed in the following section.

State of the Art:

Secure coding practices have made incredible advancements in the past several years; this is especially important as a study published by Deloitte in 2018 shows the majority of CEOs and board members cite cyber security breaches as the greatest single threat to their company's success/reputation.[11] This section will look at advancements throughout the past five years to provide an overview of recent breakthroughs in secure coding practices and the achievements of the software development community.

One vital breakthrough was the introduction of Software Guard Extensions (SGEs) by Intel Corporation in 2015.[12] SGEs function by automating encryption of portions of computer memory by the CPU via the creation of a software *enclave*, a separated area of encrypted memory that is only decryptable inside of the CPU. This allows sensitive data stored in memory to be unreadable without CPU processing. SGEs runs in the background, require no user input, and utilize uniquely generated keys created via the *eGetKey* instruction. There are three levels of security in *eGetKey* – the SGE security version numbers, the device ID for which the encrypted code is authorized, and *OwnerEpoch*, which gives developers the ability to add even further variability to the keys.[13] One downside to the use of SGEs is their potential to be exploited via malware. In principle, malware could create its own enclaves that would prevent an OS or anti-malware software from being able to decrypt active, malicious code.[14] Future development in SGEs have somewhat mitigated this risk by removing the ability of enclaves to handle exceptions inside themselves; thus anti-malware software can retain the ability to determine if malware is running inside an enclave by examining inputs and outputs; OSs can also combat this by only allowing whitelisted programs the ability to utilize enclave creation API.[15]

Blockchain technology is another huge player in recent secure coding advancements, as it functions as a distributed ledger shared across numerous devices and allows for the creation of secure, decentralized storage solutions. As companies store ever increasing quantities of sensitive data, this naturally attracts hackers and others with malicious intent. Centralized data storage exposes creates unnecessary risk – similar to the way that someone storing their cash under a single mattress exposes them to unnecessary risk from theft. Blockchain eliminates the riskiest single point of failure for protecting data – an under-protected central data repository.[16] In addition, once data is stored into a blockchain ledger, it cannot be copied without the owner's permission, granted in the form of revocable private keys.[17] The privacy-enhancing abilities provided by blockchain technology are so powerful that the EU is contemplating mandating its use for companies to remain GDPR compliant.[18] NASA also now utilizes blockchain technology in air traffic management to provide a secure and efficient method to communicate with air traffic control.[19]

Finally, advancements in machine learning and artificial intelligence are now becoming mainstream in helping companies and computer scientists maintain data security. As machine learning becomes increasingly more adept at detecting patterns and abnormalities, it can be used to differentiate normal/desired behaviors from potentially malicious ones. There are currently two main approaches to using machine learning for this. Supervised machine learning utilizes labeled datasets, and allows learning of what malicious behavior looks like. Unsupervised machine learning utilizes unlabeled datasets, and identifies and calls out behavior that is different from what is expected.[20] Machine learning allows for the detection of predicted classes, interfaces, and behaviors, and will generate appropriate deployment strategies to mitigate security risks depending on the level of threat – anything from a simple generation report to execution of code or encryption as necessary.[21]

Analysis:

The Huawei build suffered from problems with code security for several reasons. Firstly, Huawei did not have a secure strategy to ensure that the source code they use is exactly what is compiled to the executable binaries that ran on UK network equipment. Huawei attempted to utilize binary equivalence as an intermediate to correct this, but failed to do so. In addition, the report ruled that the Huawei engineering process contained "significant new additional risks" and that the HCSEC "cannot give assurances that it will be able to effectively risk-manage any new technology Huawei develops for cellular networks."[22]

Several of the strategies mentioned above would certainly be suitable for use in real, complex projects such as the Huawei build, and have the potential to create assurances for users that Huawei can effectively manage security risks. The use of blockchain technology, which is now present on Huawei cloud services, can provide essential risk mitigation for client data by decentralizing where the data is stored and eliminating the "front door" access associated with centralized storage locations.[23] Enforcement of secure coding guidelines, which Huawei has now developed, but not universally implemented, can eliminate security risks on a line-by-line basis for unsafe invocations of function calls for *memcpy()*, *strcpy()*, and *sprintf()*.[24]

While there were few unsafe usages of these functions in the Huawei build compared to safe usages, even a few unsafe usages can leave their build open to vulnerabilities. Machine learning and data mining could be utilized both to analyze code for unsafe usages, while teaching AI to detect abnormalities from safe usage, as well as to even draw attention to these issues during coding inside of an IDE. Machine learning could also be utilized by Huawei to enhance current DevSecOps, which is a platform based on the premise that everyone in the software development life cycle is responsible for security – this in turn combines security with the initial development of software, as opposed to an afterthought after having a completed build. By utilizing DevSecOps, increased automation from the very beginning of the creation of build will reduce the chance of mistakes and vulnerability to unwanted behaviors or intrusions.[25]

Conclusions:

The HCSEC's ostensible deconstruction of the code utilized in Huawei's UK build found numerous flaws and security concerns, ranging from basic unsafe function calls to potentially catastrophic buffer overflow errors. However, the advent of advancement in secure coding practices provides significant room for Huawei to improve upon their build and address these vulnerabilities. By utilizing secure coding practices implemented through blockchain technology, machine learning, data mining and AI, Huawei has the potential to successfully implement the safe coding guidelines they have created along with the HCSEC and sufficiently address these concerns.

References:

[1] Huawei Cyber Security Evaluation Center (HCSEC). “Huawei Cyber Security Evaluation Center Oversight Board – 1st Annual Report (2015).” *Gov.UK Developer Documentation*, March, 2015. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/416878/HCSEC_Report.pdf

[2] Huawei Investment and Holding Co., Ltd. “2019 Annual Report.” *Huawei Investment and Holding Co., Ltd.* December, 2019. https://www-file.huawei.com/-/media/corporate/pdf/annual-report/annual_report_2019_en.pdf?la=en

[3] Boettger, Larry et al. “The Morris Worm: How it Affected Computer Security and Lessons Learned by it.” *SANS Institute – Escal Institute of Advanced Technologies,* December 24, 2000. https://www.giac.org/paper/gsec/405/morris-worm-affected-computer-security-lessons-learned/100954

[4] Spafford, Eugene. “The Internet Worm Program: An Analysis.” *Purdue University*. December 8, 1988. https://spaf.cerias.purdue.edu/tech-reps/823.pdf

[5] Cert Division – Carnegie Mellon University. “The CERT Division.” *Carnegie Mellon University*. January, 2020. https://www.sei.cmu.edu/about/divisions/cert/index.cfm#CERTRecentlyPublishedVulnerabilityNotes

[6] Huang, Yan and Shmatikov, Vitaly. “Buffer Overflow Attacks.” *Indiana University Bloomington*. February, 2010. http://homes.sice.indiana.edu/yh33/Teaching/I433-2016/lec10-bo.pdf

[7] Open Web Application Security Project. “OWASP Secure Coding Practices Quick Reference Guide.” *OWASP*. November, 2010. https://owasp.org/www-pdf-archive/OWASP_SCP_Quick_Reference_Guide_v2.pdf

[8] PCI Security Standards Council, LLC. “Maintaining Payment Security.” *PCI Security Standards Council*. January, 2020. https://www.pcisecuritystandards.org/pci_security/maintaining_payment_security

[9] Lomes, Natasha. “Sony finally says its adding 2FA to PlayStation Network.” *TechCrunch*. April 21, 2016. https://techcrunch.com/2016/04/21/sony-finally-says-its-adding-2fa-to-playstation-network/

[10] Risk Based Security. “A Breakdown and Analysis of the December, 2014 Sony Hack.” *Risk Based Security*. December 5, 2014. https://www.riskbasedsecurity.com/2014/12/05/a-breakdown-and-analysis-of-the-december-2014-sony-hack/

[11] Deloitte Touche Tohmatsu Limited. “Deloitte 2018 CEO and Board Risk Management Study.” *Deloitte US*. January, 2020. https://www2.deloitte.com/us/en/pages/risk/articles/ceo-board-of-directors-risk-management-survey.html

[12] Skillern, Raejeanne. “Intel SGX Data Protections Now Available for Mainstream Cloud Platforms.” *Intel IT Peer Network*. February 27, 2019. https://itpeernetwork.intel.com/sgx-data-protection-cloud-platforms/#gs.awom0s

[13] Davenport, Shaun and Ford, Richard. “SGX: The Good, the Bad, and the Downright Ugly.” *Virus Bulletin*. January 7, 2014. https://www.virusbulletin.com/virusbulletin/2014/01/sgx-good-bad-and-downright-ugly

[14] Cimpanu, Catalin. "Researchers Hide Malware in Intel SGX Enclaves." *Zero Day*. February 12, 2019. https://www.zdnet.com/article/researchers-hide-malware-in-intel-sgx-enclaves/

[15] Microsoft Corporation. "Microsoft Windows Development Center – CreateEnclave Function." *Microsoft Docs*. December 5, 2018. https://docs.microsoft.com/en-us/windows/win32/api/enclaveapi/nf-enclaveapi-createenclave

[16] Arnold, Andrew. "Enterprises Considering Blockchain As Data Privacy Solution." *Forbes*. January 2, 2019. https://www.forbes.com/sites/andrewarnold/2019/01/02/heres-why-more-enterprises-are-considering-blockchain-as-data-privacy-solution/#35f81920cb73

[17] Leibel, Stephan, et al. "A Review on Blockchain Technology and Blockchain Projects Fostering Open Science." *Frontiers in Blockchain*. November 19, 2019. https://www.frontiersin.org/articles/10.3389/fbloc.2019.00016/full

[18] Arnold, Andrew. "How Blockchain Can Help Brands Become GDPR Compliant." *Forbes.* November 20, 2018. https://www.forbes.com/sites/andrewarnold/2018/11/20/can-blockchain-help-brands-become-gdpr-compliant/#2f958f8f1203

[19] Reisman, Ronald. "Air Traffic Management Blockchain Infrastructure for Security, Authentication, and Privacy." *National Aeronautics and Space Administration*. January 7, 2019. https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20190000022.pdf

[20] Liebermann, Dan. "Machine Learning in Cyber Security – Fact, Fantasy, and Moving Forward." *Booz, Allen, Hamilton – SANS.* December, 2018. https://www.sans.org/cyber-security-summit/archives/file/summit-archive-1543964671.pdf

[21] Buczak, Anna and Guven, Erhan. "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection." *IEEE Communications Surveys and Tutorials, Vol. 18, No. 2.* Q2, 2016. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7307098

[22] Conlon, Steve and McGuirk, John. "HCSEC 2019 Annual Report." *Rivada Networks*. October, 2019. https://www.rivada.com/wp-content/uploads/2019/04/why-the-uk-hcsec-report-matters-to-ireland.pdf

[23] Huawei Investment and Holding Co., Ltd. "Huawei Cloud Blockchain Service." *Huawei Investment and Holding Co., Ltd.* December, 2019. https://www.huaweicloud.com/intl/en-us/product/bcs.html

[24] Danhieux, Pieter. "Huawei Security UK Problems Demonstrate the Need for Secure Coding." *Secure Code Warrior*. June 6, 2019. https://insights.securecodewarrior.com/huawei-security-uk-problems-demonstrate-the-need-for-secure-coding/

[25] Fieman, Joseph. "Ways in Which AI Will Advance DevSecOps." *TechBeacon*. March, 2019. https://techbeacon.com/devops/3-ways-ai-will-advance-devsecops