ISBN: 978-981-18-1791-5
2021 the 11th International Workshop on Computer Science and Engineering (WCSE 2021)
doi: 10.18178/wcse.2021.06.005

Vulnerability Model and Attack Path Prediction of the UEFI Firmware Platform Based on Risk Propagation

Weihua Jiao [1 +], Qingbao Li [1], Zhifeng Chen [1] and Fei Cao[1]

[1] State Key Laboratory of Mathematical Engineering and Advanced Computing, China

Abstract. Targeted at the situation of rampant attack on UEFI Platform Firmware, this paper systematically analyzes the Security mechanisms of UEFI platform firmware. Then the vulnerability factors of UEFI firmware are described by modeling language, and a vulnerability model of UEFI firmware platform based on risk propagation (VMURP) is proposed. This paper introduces an improved PageRank algorithm to this model to reduce the influence of subjective factors which influences the accuracy of model. Based on VMURP model, an innovative method is proposed which using security configuration vector and attack vector to evaluate attack paths. Then, we use this method and VMURP to predict the most possible attack path of specific UEFI firmware platform. Finally, verify the rationality of the model and the validity of the prediction by experimental analysis. This study is helpful to quickly evaluate the vulnerability of UEFI firmware platform and predict possible attacks, gives platform managers more targeted guidance and suggestions to strengthen the security mechanisms.

Keywords: UEFI, vulnerability model, risk propagation, attack path, prediction

1. Introduction

Unified Extensible Firmware Interface (UEFI) [1] is the new generation of basic I/O system. It defines the interface specification between the operating system and the firmware platform, aiming to replace legacy BIOS. Compared with legacy BIOS, UEFI BIOS has many technical advantages (e.g., pre-operating system environment, independent drivers, good extensibility, BootServices and RuntimeServices). While these characteristics make UEFI BIOS face more threats than legacy BIOS.

Since 2012, many vulnerabilities related UEFI platform firmware and proof of concept attack have been exposed. In 2018, the first wild UEFI rootkit-LoJack [2] was discovered. Kaspersky Security Bulletin 2019 [3] points out that there will be more UEFI malware and infections in the future. UEFI Security Guidance In Modern Computer Security Solutions [4] and UEFI Trust Chain White Paper [5] points out that UEFI will not only replace legacy BIOS on PC in the future but also will be widely used in smartphones, wearable devices, airport flight control, medical monitoring and key infrastructure. With the wide application and increasingly serious security issues of UEFI, modelling and evaluation of UEFI firmware platform will be helpful for platform manager to analyse the security characteristics of the platform. What's more, it's helpful to build attack scenarios and predict the possible attacks on the platform and provide more targeted guidance for platform managers to strengthen the security policy of UEFI firmware platform.

The main contributions of this paper are as follows: (1) We use model language to describe the composition of UEFI firmware platform, related protection, risk propagation, vulnerability and attack behaviours, then build a vulnerability model of UEFI firmware platform. (2) Evaluate the value of each module by combining expert scoring and improved PageRank algorithm [6]. This method not only highlights the characteristics of UEFI firmware platform but also reduces the impact of subjectivity. (3) Evaluate the expected loss and cost of attack path based on the configuration vector and security loss of UEFI platform. (4) Finally, using the model and method above to predict the most likely attack path of UEFI firmware platform with specific security configuration.

The following is how the rest of the paper is organized: Section 2 researches the protection measures relate to UEFI firmware systematically. Section 3 defines the modelling elements and proposes VMURP

+ Corresponding author. Tel.: +18615488831.
E-mail address: jiao_weihua@163.com.

module. Section 4 evaluates the expected loss and cost of the attack path and predicts the most likely attack path according to the security configuration of the specific platform. Section 5 is the experimental analysis based on known UEFI attacks and the experimental platform. Section 6 is the conclusion.

2. Related Work

UEFI firmware platform has a variety of protection measures in the boot process. Figure 1 shows the simplified boot process and security strategy of UEFI firmware platform. Besides， there also has protection in the process of update of UEFI firmware and resume of UEFI platform. Researching these protection is the basis of the study of security of UEFI firmware platform.

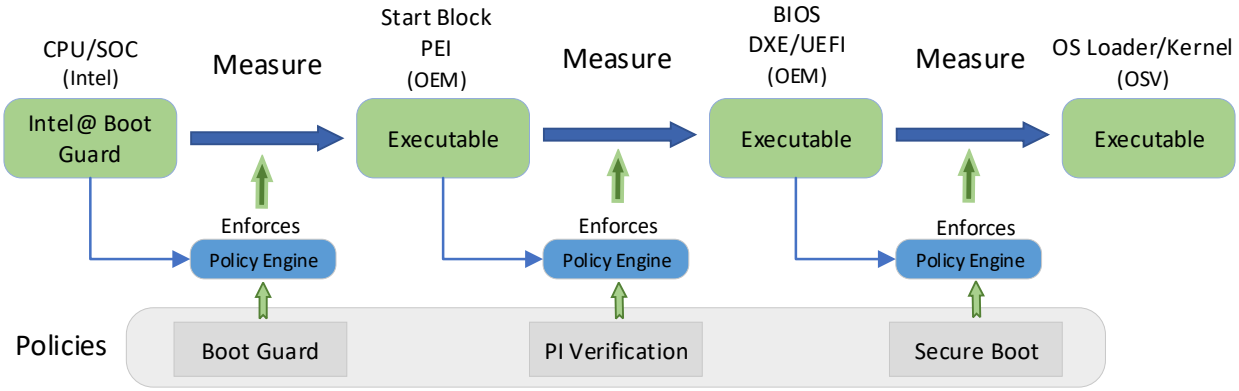

Fig. 1: Simplify boot process

Boot Guard [7] is a boot integrity protection based hardware to prevent unauthorized software or malware from taking over the boot block that is critical to system functions, thus providing a higher level of platform security based hardware. Boot Guard includes measured boot (MB) and verified boot (VB). The main function of MB is to store the measured value of the initial boot block (IBB) to the trusted platform module (TPM) [8], and to provide some security functions based hardware [9]. VB is to use some boot key to verify IBB [10] [11]. It includes two stages: CPU boot ROM verify Authenticated Code Module (ACM) and ACM verify IBB.

OBB verification. After IBB is executed correctly, the code will be executed in DXE needs to be verified to ensure that the core codes such as DEX core and DXE dispatcher are not tampered [12]. The verification results are written to the hand off block (HOB). Before executing the code of DXE, another module related to boot analyzes the verification result in HOB. If the check result shows fails, the DXE code is rejected to run.

Secure boot. The main function of UEFI Secure Boot is to confirm whether EFI driver or application is trusted by digital signature. Ensuring not to execute any code unless it is signed by a "trusted" key, whether it is an operating system boot loader, a driver in PCI Express flash memory, a driver on disk, or an update image. The trust anchors of Secure Boot include platform key (PK), key exchange key (KEK), signature database (db) and reject signature database (dbx). The holder of PK can update the signature of KEK or close Secure Boot. The KEK is used to update the signature of the db and dbx. The db lists the hash, signature or signature key of the image of UEFI application, operating system loader and UEFI driver that can be loaded on the device [13]. The dbx lists what are not allowed.

UEFI secure update. Manufacturers need to update firmware frequently to fix bugs, patch vulnerabilities, and support new hardware. UEFI attempted to define Capsule Update [14] as part of RuntimeServices to standardize the firmware update process. A firmware update key is saved on the platform. During the update process, only the image authenticated by signature can be written to flash chip. NIST 800-147 and NIST 800-147b summarize the security firmware update mechanism of PC and server respectively.

Protect register. SPI range protection register is a mechanism provided by Intel to protect BIOS firmware in flash memory of motherboard from random rewriting. In this mechanism, the address of protected area depends on the value of PR0-PR4. BIOS control register (BIOS_CNTL) contains the three fields: BWE,

BLE and SMM_BWP. BWE determine whether serial interface flash is writable. When BLE is enabled, the BWE bit is locked to disabled. The state of SMM_BWP determines whether non-SMM code is allowed to flash.

The content above make a relatively complete introduces of the BIOS protection measures in the boot process. We show it in Figure 2.

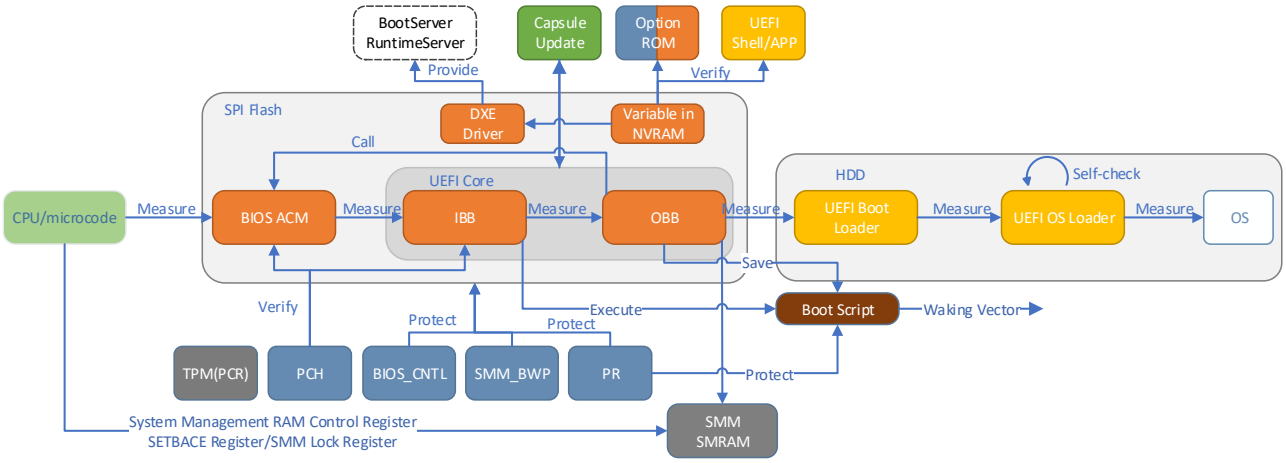

Fig. 2: Protection of UEFI platform firmware

3. Establish Model

According the research of protection mechanisms of UEFI platform above, we know there is risk propagation [15] between each module. Before structuring the model, we abstract UEFI firmware platform module, protection measures, risk propagation, vulnerability, and attack behaviour into five-tuple model: (*Mod*, *Gua*, *Risk*, *Vul*, *Att*). The parameters are as follows:

Mod: The collection of modules of UEFI firmware platform. Any platform module $C \in Mod$ can be expressed as $C=(Name, GUID, Val)$. *Name* is the name of the module, *GUID* is the unique global identifier of this module. *Val* is the value of this module, present the importance of this model for the whole platform. These modules include CPU microcode, BIOS ACM, a variety of UEFI image files, TPM chip, PCH, etc.

Gua: The collection of security protection measures for UEFI firmware platform. As for any protection $g \in Gua$, we can express it as $g = (GUID, S, <C_i, C_j>)$. *GUID* is the unique identifier of the protection measure. *S* refers to whether this safety protection measure is enabled or disabled. When *g* is enabled *S* is set to 1, otherwise is set to 0. C_i is the dependency of protection measures. When some model has its own protection measures and have no dependency, C_i is empty. C_j is the object to be protected.

Risk: The collection of risk propagation among modules of UEFI firmware platform. If attacker obtained the operation rights of C_j, C_i is also under the control of the attacker when C_j has operation rights of C_i. By exploiting risk propagation, the attacker can get more extra benefits with less cost. We use $r = (<C_i, C_j>, G, \Psi_{i,j})$ to represent the risk propagation. $< C_i, C_j >$ mean that C_i can propagate risk to C_j. *G* is the set of protection mechanisms affected by *r*. Ψ indicates the degree of protection mechanism destroyed, which is generally measured according to the level of control right, $\Psi \in (0,1]$.

Vul: Vulnerability in modules of UEFI firmware platform. It refers to a weakness in the firmware or boot process of UEFI platform. Attackers can use this vulnerability remotely or locally to gain control of a module. Any vulnerability can be expressed by $Vul = (GUID, type, \Psi)$. *GUID* is the globally unique identifier of the vulnerability. *type* is the usage of this vulnerability, where *R* is remote and *L* is local. Ψ is the level of control right that an attacker can acquire by exploiting the vulnerability, $\Psi \in (0,1]$.

Att: The collection of atomic attack behaviors. There are two types of behaviors, using vulnerabilities to invade UEFI firmware platform and using risk propagation to expand the impact of attack. Each behavior can be expressed as a six-tuple: $a = (ID, name, C, P, type, cost)$. *ID* is the unique number of the attack behavior. *name* is the name of this attack behaviour. *C* is the target module of this attack behaviour. *P* is the probability of attack successfully, which varies with the difference of attacker's ability. *type* refers to the

vulnerability exploiting or risk propagation exploiting. *cost* is the cost required to complete the atomic attack behaviors.

We formalize the UEFI firmware platform module and its protection measure, risk propagation, vulnerability and attack behaviour. In order to make module more accurate and less affect by subjectivity we also need define the *Val* of module and module security score (MSS).

Definition 3.1 *Val* of module: *Val* is the attribute of $C \in Mod$. When determining the *Val*, we need to combine the characteristics of the research object and try to avoid the influence of subjectivity. Because each module plays a different role in UEFI firmware platform different from that of ordinary network. We present a new method refer to PageRank to evaluate *Val*, we call it MBPE algorithm. The *Val* of nodes in MBPE is mainly determined by two factors. The first factor is the risk propagation between nodes. Referring to the idea of PageRank, if C_i can influence C_j, C_j will give a feedback to C_i. The other factor is the inherent importance of the node, which is scored by experts. *Val* of C_i is expressed as follows:

$$Val_{C_i} = \delta \sum_{C_j \in C_i^{out}} Val_{C_j} \frac{e\langle C_i, C_j \rangle}{\sum_{C_k \in C_j^{in}} e\langle C_k, C_j \rangle} + (1-\delta) E(C_i) \tag{1}$$

C_i^{out} refers to all the risk propagation related to C_i and point to other nodes from C_i. $E(C_i)$ is the initial *Val* of C_i according to the expert score. δ is a scaling factor, which means the contribution rate of risk propagation. Applying (1) to every node, we can get a system of linear equations with n (n is the number of nodes) unknowns. The *Val* of each node can be obtained by solving the equations or iterative calculation.

Definition 3.2 MSS: MSS indicates the security level of a UEFI firmware platform module. The security score of each module is mainly affected by two aspects: the protection policy and the current access level of attacker to this module. A module may have a variety of protection mechanisms. We give weight λ to these protection mechanisms according to the importance of each mechanism and the sum of these weights is 1. The parameter *S* in *Gua* represent whether a platform supports this protection mechanism or whether it is enabled. If this protection mechanism is enabled, *S* is set to 1. If it is not supported or disabled, it is set to 0. The attacker's current access level to the module is represented by φ. The value of φ and Ψ refer to table 1. Therefore, the MSS can be expressed as:

$$MSS = \frac{1}{\phi} \sum_{i=1}^{n} \lambda_i S_i \tag{2}$$

Table 1: Permission and corresponding value

Permission	No access	Read	Call	Disable function	Disable function and tampering
Value	0	0.1	0.5	0.7	1

On this basis above, we establish VMURP module. This module can be described by a directed network $G = (V, E)$. V is the set of nodes, for any $v \in V \wedge V \neq null$ represents a platform module. The concrete definition is shown in Table 2 in section 5. Each node has triple attributes: module security score (MSS), value (*Val*) and potential vulnerability of the module that can be exploited by attackers. *Val* and MSS will be described in detail later. If an attacker invades the system by exploiting the vulnerability of a node, then he can extend the attack path by using risk propagation. For any $e = < C_i, C_j > \in E$ represents a *r*, and the size of *e* is the corresponding Ψ value of *r*. If C_i has direct access to C_j, Ψ is determined by the level of access. If C_i is only a protection module (e.g. register), Ψ is determined according to the access rights of the attacker to C_j after losing the protection of C_i. Figure 3 is an example of a node. The vulnerability exploitation represented by a dashed line is to express specific attack behaviour in case study.

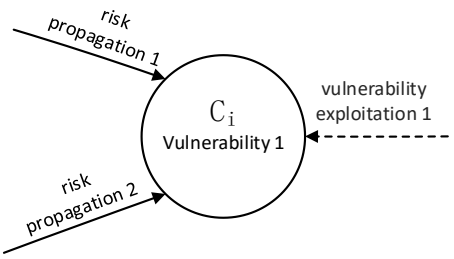

Fig. 3: Example of nodes

This model avoids the problem of too large scale of traditional attack graph model, makes the attack path clearer and less influenced by subjectivity. Assessment of attack path based on VMURP considers risk propagation and differences of the configuration of different platform. That makes the assessment of platform vulnerability and the cost of attack path more accurate. Besides, this module can provide more powerful evidence for the prediction of attack path.

4. Assessing and Predicting Attack Path

4.1. Attack Path

An attack path contains attack behaviors and modules that illegally accessed. Attack behaviors includes the original attack by vulnerability exploiting and exploitation of risk propagation. We use a directed network to represent the complete attack path, but in which the process of original attack only has one endpoint. If we do not consider the process of original attack, attack path G' (G'= (V', E'), $V^{'} \subseteq V \wedge E^{'} \subseteq E$) is a sub graph of VMURP. The nodes in the attack path is composed of modules controlled by attacker in some degree. The edge is composed of the process of vulnerability exploitation and risk propagation. Figure 4 is an example of a combination attack path.

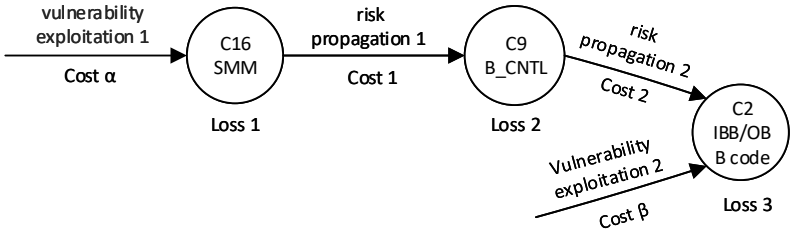

Fig. 4: Example of attack path

4.2. Assessing Attack Path

Expectation of attack cost (EAC) refers to the money, time and other measures that an attacker needs to pay to achieve the expected goal. The attack process is mainly divided into two parts represented by *type*. The cost of the original attack varies with the security configuration of specific platform. Each original attack has its corresponding protection that need to be bypassed or cracked. If the protection are enabled, the attack cost will be higher than that disabled. We take the cost of all the original attacks in an attack path as an attack cost vector $\alpha = [cost_1, cost_2, ..., cost_n]$ with corresponding protection enabled. Generally, it can be obtained by historical data analysis, estimation or actual attack measurement. The situation of whether corresponding protection is enabled as the safety configuration vector $b = [\theta_1, \theta_2, ..., \theta_n]$. When the corresponding protective measures are enabled, the value of θ is 1. When the protection is disabled, the value of θ meets $0 < \theta < 1$. The cost of all the original attacks is equal to the product of two vectors. The cost of a complete attack path (EAC) also needs to add the cost of all risk propagation exploitation. The formula is as follows:

$$EAC = \mathbf{A} \bullet \mathbf{B} + \sum_{j=1}^{m} Cost_j \tag{3}$$

Expectation of attack loss (EAL) represents the loss of the UEFI firmware platform when the attacker successfully exploits the relevant vulnerabilities and risk propagation to achieve the expected attack purpose. The higher the expected value, the attack is more destructive. We use module security loss to measure the

influence of attack. The measurement standard of single module loss mainly includes two aspects: *Val* of attacked module and loss of MSS. The EAL is the sum of all related module losses. The calculation formula is as follows:

$$EAL = \sum_{i=1}^{n} Val_i \left(MSS_i - MSS_i^{'} \right) \tag{4}$$

4.3. Predicting of attack path

Based on the analysis of protection, UEFI vulnerability and attack cases, we predict the most likely attack path combined with the configuration of specific platform.

To predict the attack path, the security configuration of the platform needs to be detected first. This process can be carried out by using Chipsec [16]. We should pay more attention to the security protection measures that disabled or not supported, which may be the configuration vulnerabilities that attackers can exploit to invade the UEFI firmware platform. The corresponding nodes of this protection is the possible node of attack. Then, according to risk propagation, we can find the possible successor nodes in the attack path. There may be more than one successor node, so attack path may have several branches. If all attack path are stored in the data structure of tree, we may obtain a forest.

Every risk propagation *r* has its attribute *g*, and g has its corresponding λ in (2). We regard *r* as effective risk propagation when it satisfy $0.5 \le |e| \lambda$, which not only improve the efficiency of the algorithm but also don't miss the most likely attack path. Generally. Attacker will choose the attack path with low EAC but high EAL. Calculating the Calculate the ratio of EAL and EAC (*LC*) of all possible attack paths. The attack path with the largest *LC* is the most likely attack path that the attacker is likely to take.

$$LC = \frac{EAL}{EAC} \tag{5}$$

5. Experiment and Analysis

Val of each node in VMURP are calculated according to (1), the results are shown in table 2.

Table 2: *Val* of each node in VMURP

Node	Module	*Val*	Node	Module	*Val*	Node	Module	*Val*
C0	CPU/Microcode	9.83	C6	TPM	9.41	C12	DXE Driver	8.58
C1	BIOS ACM	8.28	C7	PCH	7.57	C13	Update Image	7.83
C2	IBB and OBB Code	9.32	C8	ACPI	5.81	C14	Option ROM	6.87
C3	UEFI Boot Loader	8.20	C9	SMM/SMRAM	9.17	C15	UEFI APP	6.16
C4	OS Loader	8.15	C10	BIOS_CNTL	9.28	C16	Flash Descriptor	8.63
C5	OS Kernel	9.39	C11	Boot Script	6.87			

According to the results above, we can see that the *Val* of CPU/microcode, TPM security chip, BIOS code and SMM/SMRAM rank in the top four. But unfortunately, only some high-end platforms are equipped with TPM chips. Even if each platform supports BIOS write protection, only 32% of the more than 200 motherboards and PC we tested have enabled BIOS write protection.

It is because of the existence of modules with high value and impact but with vulnerability (e.g. BIOS code and SMM), make them the most favorite target for attackers to carry out attacks. We have counted some proof of concept attacks, wild attack cases and vulnerabilities reported, the frequency of each module appeared in attack path can be known according statistics shown in Figure 5. It can be seen that the protection register appears most frequently, that is because whether tampering UEFI code or NVRAM attacker must solve the problem of protection register first. Which can make the attacker obtain the persistent attack effect. SMM is the highest privilege of UEFI firmware platform. Under SMM, the behavior of attacker is transparent to the operating system that make the impact of the attack greater and more covert. According to the analysis above, we can know that BIOS re-flash and obtain SMM privilege may be the most frequently means used by attackers. We can see the consistency between analysis and reality, proving the effectiveness of model.

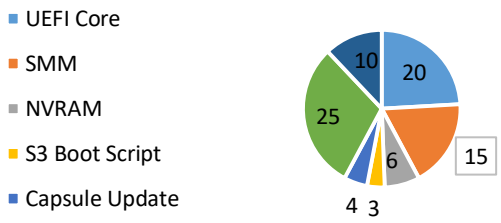

Fig. 5: Frequency of modules appeared in attack paths

Because GIGABYTE-H81M motherboard is wiled used now and presents the configuration of most motherboards, we choose it as the experimental platform and predict the most likely attack path of it. Table 3 list the security configuration related UEFI firmware of GIGABYTE-H81M.

Table 3: Security configuration of GIGABYTE-H81M

Project	GIGABYTE-H81M
bios_smi	[-] SMM BIOS region write protection has not been enabled (SMM_BWP is not used)
bios_wp	BIOSWE = 0 << BIOS Write Enable ; BLE = 0 << BIOS Lock Enable
bios_pr	[!] None of the SPI protected ranges write-protect BIOS region
smm	[+] PASSED: Compatible SMRAM is locked down
spi_desc	[-] FAILED: SPI flash permissions allow SW to write flash descriptor
spi_lock	[+] PASSED: SPI Flash Controller locked correctly
s3bootscript	[-] FAILED: S3 Boot-Script and Dispatch entry-points do not appear to be protected [!] Additional testing of the S3 boot-script can be done using tools.uefi.s3script_modify
Secureboot	[*] Secure Boot appears to be disabled [-] Some required Secure Boot variables are missing
TPM	Don't support
…	…

As for GIGABYTE-H81M, BIOS flash is not protected, S3 Boot-Script also don't protected and Secure Boot is disabled. These all may be the possible attack point. Attacker can use re-flash firmware to gain "permanent" control. Attacker can tamper OS Loader or other executable file directly to control OS. Assessment of possible attack path are shown in Table 4. The most likely attack paths we predict and characteristic of them are shown in Table 5. The most likely attack path predicted by the model is highly consistent with the experience, which verifies the effectiveness of the model.

Table 4: Assessment of possible attack path

	Platform	Attack path	EAL	EAC	LC
1	GIGABYTE-H81M	C10→C2→C3→C4→C5	23.2	12.0	1.93
2		C10→C2→C12→C5	26.1	10.5	2.48
3		C10→C2→C4→C5	22.8	10.5	2.17
4		C11→C10→C2→C3→C4→C5	26.2	14.0	1.87
5		C11→C10→C2→C12→C5	29.1	12.5	2.32
6		C11→C10→C2→C4→C5	25.8	12.5	2.06
7		C4→C5	6.7	2.8	2.39

Table 5: Result of prediction

Attack path	Characteristic
C10→C2→C12→C5	Attacker can gain "permanent" control.
C4→C5	Sample and easy to implement.

6. Conclusion

In this paper, we research the security protection mechanism of UEFI firmware platform systematically, and establish VMURP model based on this. PageRank algorithm is introduced into model to reduce the

influence of subjective factors. Based on the attributes of specific UEFI firmware platform, the loss and cost of attack path are evaluated more accurately. Finally, we propose an innovative method to predict the most likely attack path of specific UEFI firmware platform. Providing more targeted guidance for platform managers to strengthen the platform security configuration.

However, the model proposed in this paper is only for the current UEFI firmware platform, and it needs to be upgraded when the platform technology innovation. Only simple attack path can be predicted in this paper, prediction of combination attack remain to be further discussed. Therefore, the vulnerability modelling and security research of UEFI firmware platform is a continuous process. We look forward to more research and progress in this field.

7. Acknowledgements

Thanks are due to Li for valuable discussion and to Chen for assistance with the experiments. Thanks for the valuable guidance of Fei Cao and important suggestions of Ming Liu. Last, we thank the anonymous reviewers for their comments to improve the quality of this paper.

8. References

[1] UEFI Forum, Unified Extensible Firmware Interface Specification. Version 2.8. https://uefi.org/sites/default/files/resources/UEFI_Spec_2_8_final.pdf, 2020.

[2] Urban. Schrott, LOJAX: First UEFI rootkit found in the wild, courtesy of the Sednit group, https://www.welivesecurity.com/wp-content/uploads/2018/09/ESET-LoJax.pdf, 2018.

[3] GReAT, Kaspersky Security Bulletin 2019-Advanced threat predictions for 2020, https://securelist.com/advanced-threat-predictions-for-2020/95055/, 2019.

[4] Richard. Wilkins, Brian. Richardson, UEFI SECURE BOOT IN MODERN COMPUTER SECURITY SOLUTIONS, https://uefi.org/sites/default/files/resources/UEFI_Secure_Boot_in_Modern_Computer_Security_Solutions_2019.pdf, 2019.

[5] Richard. Wilkins, Toby. Nixon, THE CHAIN OF TRUST, https://uefi.org/sites/default/files/resources/UEFI%20Forum%20White%20Paper%20-%20Chain%20of%20Trust%20Introduction_2019.pdf,2019.

[6] Page, Lawrence, Sergey. Brin, Rajeev. Motwani, and Terry. Winograd, The pagerank citation ranking: Bringing order to the web, Tech. Rep.no., Stanford Digital Library Technologies Project, 1998.

[7] Intel, New Microarchitecture for 4th Gen Intel@ CoreTM Processor Platforms, https://www.intel.cn/content/dam/www/public/us/en/documents/product-briefs/4th-gen-core-family-mobile-brief.pdf, 2013.

[8] TCG, Trusted Platform Module Library Part 1: Architecture, https://trustedcomputinggroup.org/wp-content/uploads/TCG_TPM2_r1p59_Part1_Architecture_pub.pdf, 2019.

[9] Maxfield. Chen, Hardware Root of Trust - Bios and UEFI, https://maxfieldchen.com/posts/2020-05-31-Hardware-Root-Of-Trust-Bios-UEFI.html, 2020.

[10] William. Futral,James. Greene, Intel® Trusted Execution Technology for Server Platforms, 2013, pp 15-36.

[11] Trammell. Hudson, Chris. Pick, Anisse. Astier, safeboot, Vol. Chain of Trust, https://safeboot.dev/chain-of-trust/.

[12] UEFI Forum, UEFI Platform Initialization Specification, Version 1.7, https://uefi.org/sites/default/files/resources/PI_Spec_1_7_A_final_May1.pdf, 2020.

[13] Microsoft, Secure boot, https://docs.microsoft.com/en-us/windows-hardware/design/device-experiences/oem-secure-boot, 2019.

[14] Meenakshi. Agrawal, Udit Kumar, Capsule update with MM, https://www.uefi.org/sites/default/files/resources/NXP_Capsule%20update%20with%20MM_Fall%202018%20Plugfest.pdf, 2018.

[15] Yongzheng. Zhang, Binxing. Fang, Yue. Chi, Xiaochun. Yun, Risk Propagation Model for Assessing Network Information Systems [J]. Journal of software, 2007, 18(1):137-145.

[16] Mikecb, Harmon-tech, Chipsec, https://github.com/chipsec/chipsec.