Dynamic Ransomware Protection using Deterministic Random Bit Generator

[1]HaEun Kim, [2]Dongchang Yoo, [3] Ju-Sung Kang, [4†]Yongjin Yeom
Dept. of Information Securitiy, Cryptology, and Mathematics
and Dept. of Financial Information Security,
Kookmin University
Seoul, Korea
{[1]iop862, [2]yoodc8167, [3]jskang, [4]salt}@kookmin.ac.kr

***Abstract* — Ransomware has become a very significant cyber threat. The basic idea of ransomware was presented in the form of a cryptovirus in 1995. However, it was considered as merely a conceptual topic since then for over a decade. In 2017, ransomware has become a reality, with several famous cases of ransomware having compromised important computer systems worldwide. For example, the damage caused by CryptoLocker and WannaCry is huge, as well as global. They encrypt victims' files and require user's payment to decrypt them. Because they utilize public key cryptography, the key for recovery cannot be found in the footprint of the ransomware on the victim's system. Therefore, once infected, the system cannot be recovered without paying for restoration. Various methods to deal this threat have been developed by antivirus researchers and experts in network security. However, it is believed that cryptographic defense is infeasible because recovering a victim's files is computationally as difficult as breaking a public key cryptosystem. Quite recently, various approaches to protect the crypto-API of an OS from malicious codes have been proposed. Most ransomware generate encryption keys using the random number generation service provided by the victim's OS. Thus, if a user can control all random numbers generated by the system, then he/she can recover the random numbers used by the ransomware for the encryption key. In this paper, we propose a dynamic ransomware protection method that replaces the random number generator of the OS with a user-defined generator. As the proposed method causes the virus program to generate keys based on the output from the user-defined generator, it is possible to recover an infected file system by reproducing the keys the attacker used to perform the encryption.**

Keywords : Ransomware, Cryptovirus, Protection, Deterministic Random Bit Generator

I. INTRODUCTION

Ransomware is a kind of malware that encrypts files in a victim's computer and requires payment before restoration. It has become a very serious cyber threat. NO MORE RANSOM [1] is a project formed to fight against ransomware attacks, in which the Dutch National Police, Europol, McAfee, and Kaspersky Laboratories also participate. NO MORE RANSOM provides free recovery tools on its website. The tools provided are known to have recovered 28,000 infected devices thus far, but Warner Cry, a type of cryptovirus, is not on the recoverable list. Recent cryptoviruses such as WannaCry and Petya penetrate computer systems and encrypt user files using cryptographic algorithms, and subsequently require bitcoin payments. Once infected, it is infeasible to restore infected files since they are encrypted using strong cryptographic algorithms. In the 1990s, Young and Yung [4] introduced the basic idea of cryptovirology as malicious usage of the cryptographic technique. Their work shows that if a malware makes use of public key cryptography, the damage cannot be recovered without the help of the attacker. At the time their work was presented, it was not considered realistic for two main reasons: 1) public key encryption could not be implemented within a tiny virus and 2) the attacker had no way of collecting ransom money without revealing his/her identity. However, as these obstacles have recently been removed, cryptovirus attacks have become a reality. First, operating systems provide APIs to exploit strong cryptographic functions so that a few lines of code are sufficient to use public key encryptions such as RSA [7]. Second, crypto-currency enables attackers to collect money anonymously, with bitcoin being the most popular one used for malware. Consequently, malware is able to elude the police and are very active.

The WannaCry ransomware infected several important organizations worldwide, including the U.K.'s National Health Service. It spread rapidly, with medical staff in the U.K. reportedly seeing computers becoming infected in rapid succession. NHS staff shared screenshots of WannaCry demanding bitcoin payments to unlock the files for each computer. Any systems on a network can be infected through malicious email attachments or remote resource sharing. Thus, people are extremely concerned that their systems might become infected, with some people even considering pulling out their LAN cable or reducing the uptime of their computers as a precaution.

In response to the increases in ransomware attacks, antivirus research is actively underway. For example, reverse engineering

† Corresponding Author

is being applied to examine execution codes through static and dynamic analyses to find clues to recover infected files from ransomware. Monitoring critical API calls and dynamic analysis of abnormal patterns in execution are also being used to prevent malware from accessing cryptographic APIs. The limitation of these approaches is that user files cannot yet be successfully recovered after encryption.

In 2017, Palisse et al. [6] proposed methods for proactively protecting against ransomware using cryptographic techniques. Their methods are based on the vulnerability of block cipher modes of operation and also dynamic blocking of cryptographic API calls by ransomware. However, with their proposed methods, if the ransomware does not use a specific mode of operation, then the user cannot defend against the ransomware. Further, it is assumed that the method has effect about 50% only when prevention is done in advance. Kolodenkerz et al. [2] proposed a variant of the key escrow mechanism called PAYBREAK. PAYBREAK stores all random numbers generated by the system in a large log file so that users later perform exhaustive encryption key search from the log file. PAYBREAK hooks crypto APIs, particularly the random number generation API called *CryptGenRandom*(). When ransomware collect entropy and create encryption keys using *CryptGenRandom*(), PAYBREAK is very effective.

In this paper, we propose a dynamic method of protection against ransomware in which the encryption key is recovered upon infection. The proposed method is effective for ransomware that rely on *CryptGenRandom*(), as described in Section 2, and presents a more efficient defense mechanism than the methods in PAYBREAK. The proposed method requires implementation prior to infection. Subsequently, once infected, users can find the encryption key used by the ransomware and recover their files successfully.

II. RANSOMWARE USING CRYPTO API

Early types of ransomware programs contained their own encryption algorithms. However, the code size of recent ransomware is extremely small as they utilize encryption algorithms via cryptographic APIs. In fact, Microsoft Windows provides a cryptographic library as part of the operating system. This makes it easy for ransomware to encrypt a victim's files. Microsoft's Crypto API [7] supports public key and symmetric key encryptions and works with several Cryptographic Service Providers. However, this Crypto API does not distinguish between viruses and legitimate user applications requesting the Cryptographic Services. Therefore, ransomware can access the functions in the crypto API to encrypt the user's files.

Our main focus is ransomware that uses the MS Crypto API, particularly those that rely on *CryptGenRandom*(), which has become common in recent ransomware, including CryptoLocker and WannaCry. Such ransomware attacks employ a symmetric key cryptographic algorithm AES in crypto API to encrypt files. The symmetric key encryption algorithm contains the encryption function "Enc" and the decryption function "Dec". For messages or files to be encrypted as "M", the same key K_{file} for encryption and decryption are as shown in (1). The output value of the encrypt function "Enc" is the ciphertext denoted by "C". The symmetric key algorithm AES is known to be secure, which makes it impossible to decrypt without the encryption key. Thus, the key K_{file} used by the ransomware for encryption must be found to recover files.

$$\mathrm{Enc}_{K_{file}}(M) = C,\ \mathrm{Dec}_{K_{file}}(C) = \mathrm{M} \tag{1}$$

However, ransomware encrypt the file encryption key K_{file} using asymmetric key encryption. In the asymmetric key encryption algorithm RSA, the encryption function is "Enc′" and the decryption function is "Dec′." The encryption (public) key K_{pub} and the decryption (private) key $K_{private}$ are different, as shown in (2).

$$\mathrm{Enc'}_{K_{pub}}(K_{file}) = C',\ \mathrm{Dec'}_{K_{private}}(C') = K_{file} \tag{2}$$

Ransomware use their own public key K_{pub} to prevent exposure of the key K_{file}. Therefore, even if encryption key K_{pub} is exposed, the encrypted key K_{file} cannot be decrypted. K_{file} has to be obtained from its encrypted form $\mathrm{Enc'}_{K_{pub}}(K_{file})$ by decryption with private key $K_{private}$. In most cases, private key $K_{private}$, is stored only on the attacker's system. Therefore, any effort to decrypt the infected files using the information in the victim's computer is hopeless without $K_{private}$. At this point, if ransomware encrypt files using public key encryption, it can encrypt files without any effort to protect the key K_{file}. Nevertheless, the reason for using a symmetric key encryption algorithm is that the public key encryption algorithm is too slow.

We focus on the process of making the file encryption key K_{file}, which ransomware randomly generate using *CryptGenRandom*(), which is provided by the Windows API, and is known as a cryptographically secure pseudorandom number generator. The output of *CryptGenRandom*() is unpredictable because it generates random numbers using strong cryptographic algorithm that is started with a seed with multiple noise sources from the operating environment. After file encryption, the ransomware encrypts the key K_{file} with RSA public key K_{pub}, so that only the attacker can subsequently obtain the key K_{file} later. The scheme of ransomware attack on user files is depicted in Figure 1.

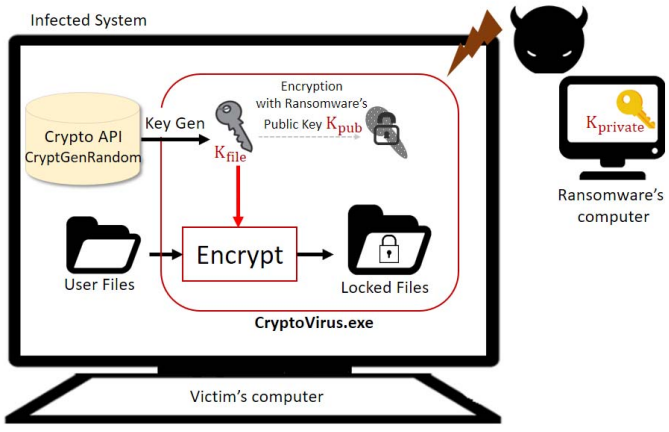

Fig. 1. Attack scheme of cryptoviruses: The attacker demands ransom from the user. Unless the user recovers the key K_{file} generated by Crypto API, the user cannot access the encrypted files.

In summary, ransomware attacks victim's computer as follows:

- Step 1. The ransomware resides victim's computer through the network.
- Step 2. The ransomware generates K_{file} from pseudorandom number generator *CryptGenRandom*(), provided by MS Windows.
- Step 3. The ransomware encrypts files using symmetric key algorithm with the encryption key K_{file}.
- Step 4. The ransomware encrypts K_{file} using an asymmetric key algorithm with its own public key K_{pub} to prevent exposure of the key K_{file}.
- Step 5. The ransomware demands money after encrypting files.

III. Our Prevention Scheme Against Ransomware

We propose a method that recovers user's file with minimal effort to find out the key K_{file}. In the case of the target ransomware, there are no effective methods except for key exhaustive search in order to recover the encrypted file. In PAYBREAK [2], all random bits generated from *CryptGenRandom*() are stored in order to determine the K_{file} used by the ransomware. However, this is very inefficient because not only ransomware but also other applications use *CryptGenRandom*() to generate secure pseudorandom numbers for their own purposes. Moreover, it is not possible to predict when a ransomware has penetrated the system, which necessitates that all values have to be saved continuously.

Thus, we propose replacing *CryptGenRandom*() with a deterministic random bit generator (DRBG). Using this method, the K_{file} can be obtained more effectively. The deterministic random bit generator gives the same output value starting with the same seed value. In this paper, it is assumed that *CryptGenRandom*() can be replaced with user's deterministic random bit generator.

A. The deterministic random bit generator

A deterministic random bit generator (DRBG) is a device or algorithm that outputs a sequence of binary bits that appears to be statistically independent and unbiased [5]. It makes it possible to output a long sequence of random bits from a secret seed with a small length:

$$\text{DRBG(seed)} = \text{pseudorandom bits} \quad (3)$$

The security of the DRBG is related to the entropy of the seed. An additional input is used in the current version of DRBG to increase its security. The security evaluation guidance for the entropy input is written in SP 800-90B. The document has now been updated to the second draft version [3, 8]. The DRBG operates with seed input value and updates the internal state to generate cryptographically secure random bits. There are value *v* and *reseed_counter* in the internal state of the DRBG. The value *v* is the input values of *Generate_Function,* which generates pseudorandom bits for the DRBG. *Generate_Function* extracts random bits from *v*, and calls *Reseed_Function* when a restarting with new seed is required. When *Reseed_Function* is called, *reseed_counter* is incremented and a new internal state that is new value of *v* is generated by from old value of *v* into *Reseed_Function*. *Generate_Function* then uses the updated internal state of *v* to generate random bits. The same process is used to update the internal state when a reseed is needed. The procedure employed by DRBG is depicted in Figure 2.

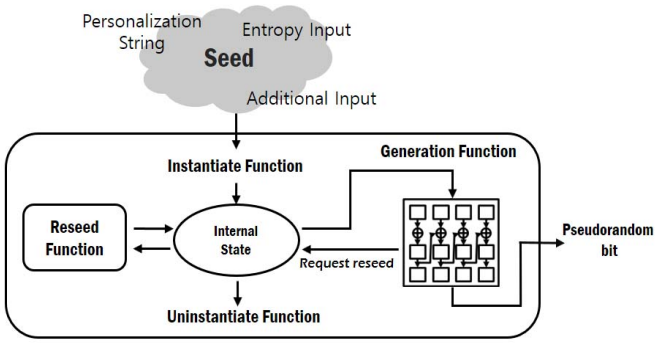

Fig. 2. Determinimistic Random Bit Generation (DRBG) Mechanism, A long sequence of psudorandom bits are generated from a secret seed with a small length: (SP 800-90A).

B. Defense procedure

The user can recover the random number sequence, which is the output value of the DRBG, by saving the seed. Thus, even if a ransomware obtains the K_{file} from the user's DRBG, the K_{file} can be recovered by reproducing from the seed. Therefore, users can defend against ransomware more efficiently using this method than previous schemes. The steps in the defense procedure are as follows.

First, the user sets the seed value, the initial state of the DRBG. As this seed value must be kept secure, it is suggested that it be encrypted with the public key, or backed up to an external device such as a mobile phone to protect the user's seed. Then, the user instantiates user DRBG and set it to replace *CryptGenRandom*(). At this point, the crypto API need to be hooked. Hooking is a method that is used to modify application behavior by overloading original functions with arbitrary new functionality. API hooking is used to modify applications without intervening in their source code [9]. Several types of hooking are possible [2]. Assuming that *CryptGenRandom*() is replaced, the ransomware's procedure working with user's DRBG instead of *CryptGenRandom*() throughout the system.

If a ransomware has penetrated deep beyond the point at which it was replaced, there may be difficulty recovering all of the random bits that have been used. We present a way for users to save their internal state in the log file in a manner that enables rapid recovery of the secret key. Saving *reseed_counter* and the internal state *v* by date results in faster recovery of the random number than extracting all the random numbers from the first used seed. Therefore, we present a method of saving internal state *v* and *reseed_counter* in a log file and sending the log file by e-mail to prevent encryption by the ransomware. Our scheme, which involves replacing *CryptGenRandom*(), is shown in Figure. 3.

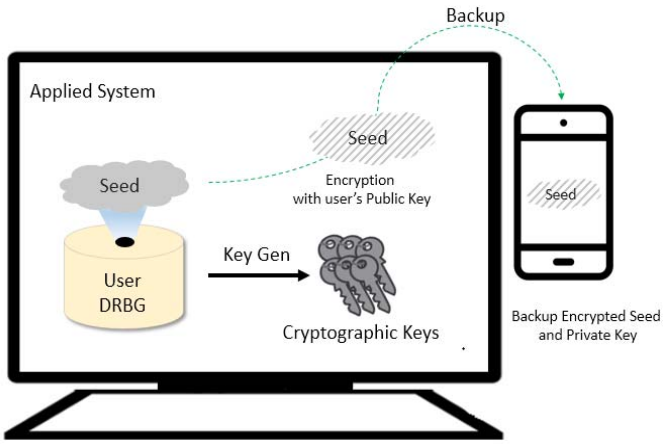

Fig. 3. Proposed prevention scheme; replacing "CryptGenRandom()" with a user DRBG that generates random numbers from a seed. The seed should be stored on an external device (such as a smartphone).

C. Recovery scheme

On penetrating a system, ransomware is supposed to use randomness from *CryptGenRandom*() to derive an encryption K_{file}. At this point, because *CryptGenRandom*() has been replaced with user DRBG, random bits used as K_{file} are generated from the user DRBG. Thu, the user can recover later all random bits if the seed is saved on an external device. As the ransomware does not know whether K_{file} is a recoverable key generated by the user's DRBG, it uses the K_{file} to encrypt the user's files. Following this process, the user can recover the random bits from the seed, internal state and *reseed_counter*. Because DRBG is used for many applications including ransomware, the user has to conduct an exhaustive key search to find the correct K_{file}. The user then successfully recovers all encrypted files using K_{file}. The secret key K_{file} recovery and the decryption scheme is shown in Figure. 4.

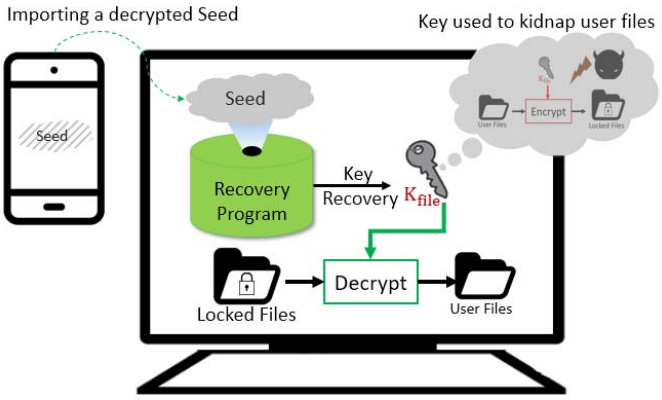

Fig. 4. Proposed scheme for recovering secret keys and decryption. Using a seed backed up on an external device, the key used to encrypt the user files is recovered.

To sum up, the procedure employed by the dynamic ransomware protection method using DRBG is as follows:

1) Set protection mode

- Step 1. Set a seed value and the initial state of DRBG
- Step 2. Backup the seed to an external device
- Step 3. Hook the Crypt API *CryptGenRandom*() to the user's DRBG
- Step 4. Save *reseed_counter* and the internal state *v* by date

2) Ransomware infection and file encryption

- Step 1. Ransomware randomly generates K_{file} based on the randomness from the user's DRBG
- Step 2. Ransomware encrypts files using symmetric key algorithm with K_{file}
- Step 3. Encrypt K_{file} using public key cryptography

3) Recovery scheme

- Step 1. Recover K_{file} from DRBG with the seed
- Step 2. Decrypt the encrypted files using K_{file}

D. Security of the proposed method

There are some security considerations when DRBG replaces *CryptGenRandom*(). First of all, the seed should be generated with high entropy in order not to be predictable. In modern cryptography, if the length of a seed is at least 256 bits, the DRBG is considered to generate secure and unpredictable random bits. To enhance the security level, the user should increase the length of the seed or use an additional input. The user can set the desired security level of DRBG. Second, if the seed is exposed, all applications that need cryptographically secure random numbers could be vulnerable. However, this will not occur if the seed is stored only on external devices. In addition, we recommend encryption of the seed with a public key to keep it secure.

IV. Conclusion and Future Work

In this paper, we proposed a ransomware defense method that enables recovery of the key value by replacing the crypto API. This method can be used to alleviate the infection concerns of users who connect to networks or the internet. The proposed method protects user's files from ransomware. In this paper, we proposed user's DRBG as an alternative to the Crypto API. The experimental implementation of the proposed prevention scheme will be considered in future studies.

Acknowledgment

This work was partly supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2017-0-01857, Research on the security of random number generators and embedded devices) and partly supported by the Korea Internet & Security Agency (KISA) (KISA2017-0102, Research on the use of cryptography for fostering domestic cryptographic industry).

References

[1] "NO MORE RANSOM", 2016. [Online]. Available: http://www.nomoreransom.org/

[2] A. Kharraz and E. Kirda, "Redemption: Real-time protection against ransomware at end-hosts," The 20th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2017)

[3] A. Palisse, H. Le Bouder, J. L. Lanet, C. Le Guernic, and A. Legay, "Ransomware and the legacy crypto API," In International Conference on

Risks and Security of Internet and Systems (pp. 11-28). Springer, Cham. (2016, September).

[4] A. Young and M. Yung, "Cryptovirology: Extortion-based security threats and countermeasures," IEEE Symposium on Security & Privacy, pp. 129-141, May 6-8, 1996.

[5] A. Young, "Cryptoviral extortion using Microsoft's Crypto API," International Journal of Information Security 5.2 2006: 67-76.

[6] E. Barker and J. Kelsey, "Recommendation for random number generation using deterministic random bit generators," NIST special publication 800-90A, 2012.

[7] E. Barker and J. Kelsey, "Recommendation for the entropy sources used for random bit generation," NIST DRAFT Special Publication 800-90B, 2012.

[8] E. Kolodenker et al., "PayBreak: Defense against cryptographic ransomware," Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. ACM, 2017.

[9] J. Berdajs and Z. Bosnić, "Extending applications using an advanced approach to dll injection and API hooking," Software: Practice and Experience 40.7 2010: 567-584.

[10] M. S. Turan, E. Barker, J. Kelsey, K. A. McKay, M. L. Baish, and M. Boyle, "Recommendation for the entropy sources used for random bit generation," (Second DRAFT) NIST SP 800-90B, 2016.