Available Online at http://www.journalajst.com

ASIAN JOURNAL OF SCIENCE AND TECHNOLOGY

ISSN: 0976-3376

Asian Journal of Science and Technology
Vol. 6, Issue 03, pp. 1231-1235, March, 2015

RESEARCH ARTICLE

IMPEDING MALWARE DETECTION AND ANALYSIS FOR BINARY EXECUTION IN CIPHERXRAY

***Aruna, K. Jayalakshmi, G. Sathyavathy, S. Rekhaswathi, G. and Sudhani, B.**

A.V.C College of Engineering, Mayiladuthurai, Mannampandal

ARTICLE INFO

Article History:
Received 28th December, 2014
Received in revised form
14th January, 2014
Accepted 19th February, 2015
Published online 31st March, 2015

Key words:
Binary analysis,
Cryptographical Operations,
Key Recovery,
Transient secrets,
Cryptographical algorithm.

ABSTRACT

Malwares have become progressively lurking, a lot of malwares are victimisation cryptographical algorithms (e.g., packing, encrypting C&C communication) to shield themselves from being analyzed. The utilization of cryptographical algorithms and really transient cryptographical secrets within the malware binary imposes a key obstacle to effective malware analysis and defense. To modify more practical malware analysis, forensics, and reverse engineering, we have got developed CipherXRay - a completely unique binary analysis framework that may mechanically determine and recover the cryptographical operations and transient secrets from the execution of doubtless obfuscated binary executables based on the avalanche impact of cryptographical functions, CipherXRay is ready to accurately pinpoint the boundary of cryptographical operation and recover really transient cryptographical secrets that solely exist in memory for one instant in between multiple nested cryptographical operations. CipherXRay will more determine bound operation modes (e.g., ECB, CBC, CFB) of the known block cipher and tell whether or not the known block cipher operation is encoding or secret writing in bound cases.

INTRODUCTION

In this paper, we present Cipher X Ray – a unique binary analysis framework which will accurately pinpoint the boundary of individual cryptological operation from multiple rounds of cryptological operations and recover actually transient secrets from the execution of a doubtless obfuscated binary executable rather than mistreatment instruction identification, we build Cipher X Ray upon one in every of the shaping characteristics of all (good) cryptological algorithms – the avalanche result, which refers to the required property of cryptological perform such that one bit amendment within the input or key would cause important change within the output. Cipher X Ray has the subsequent nice features:

- It is able to dependably discover, pinpoint and distinguish the operations of public key cryptographical algorithms (e.g., RSA), block cipher (e.g., AES), and hash (e.g., SHA-1), even though they are mingled with one another or non-cryptographic operations, from the execution of a given binary workable.
- It will accurately pinpoint the situation, size and boundary of the input, the output and therefore the key buffers of every of the multiple rounds of cryptographical operations and determine the precise time once the input, the output, the IV and therefore the key of every known cryptographical operation will be prepared. This permits to recover the data input, the data output, the secret key (e.g.,
- 256-bit AES key) or the non-public key (e.g., 1024-bit RSA key) employed in each identified cryptographical operation even though multiple cryptographic operations nested (e.g., encryption first and hash second) and therefore the cryptographical secrets are transient.
- Cipher X Ray will more determine sure operation modes (e.g., ECB, CBC, CFB) of the known block cipher and tell whether or not the known cipher operation is coding or decoding insure cases.
- Since the computer code obfuscation (e.g., self-modifying code) does not modification or take away the avalanche result of any cryptographical operations within the original binary executable, Cipher X Ray may well be effective on obfuscated binary workable as long because the avalanche result is detected.
- It is quite generic there in it's not obsessed with specific implementation.

We have through empirical observation evaluated the effectiveness of Cipher X Ray with Open SSL [1], standard Keepass X [2] password safe, malware Stuxnet, Kraken and Agobot, and variety of third party softwares with integral checksum and compression. Despite that KeePass X reencrypts all the sensitive knowledge at intervals 21 microseconds when they need been decrypted and used at run-time, Cipher X Ray is ready to recover not solely all the protected entry passwords

****Corresponding author: K. Aruna,***
A.V.C College of Engineering, Mayiladuthurai, Mannampandal – 609 305

however additionally the 256-bit master and the 128-bit IV that alter one to directly decipher the Kee Pass X password file exploitation Open SSL; Cipher X Ray is additionally able to recover all block cipher secret keys from the binary executables obfuscated by variety of packers (e.g., UPX, AS Pack, PE Compact). Cipher X Ray has with success recovered the key utilized by Agobot and therefore the secrets of proprietary ciphers utilized by Kraken and Stuxnet malware.
To the simplest of our information, Cipher X Ray is that the first binary analysis framework which will accurately.

- Pinpoint the boundary between multiple rounds of science operations;
- Recover actually transient science secrets and keys that exist in run-time memory for less than many small seconds and
- Recover the sort of science operations and sure modes of operation of block ciphers.

Related work

The three commonly used techniques to pinpoint the malware are:

Reverse Engineering

In this paper, Zhiqiang Lin, Xiangyu Zhang, Dongyan Xu proposed a reverse engineering [3] technique to mechanically reveal program information structures from binaries during this schema, REWARD technique is employed that is predicated on dynamic analysis. REWARDS executes the binary, monitors the execution, aggregates and analyzes runtime info, and finally recovers each the syntax and linguistics of information structures discovered within the execution. REWARD involves backward kind propagation and determination procedure. Type sink is employed that determine system calls, commonplace library calls, and type-revealing directions. Reward are often applied many application like mental image rhetorical and binary vulnerability fuzzy there have been some limitations of this technique as follows:

- REWARDS cannot win full coverage of information structures outlined during a program.
- A gift doesn't support the reverse engineering of kernel-level information structures.
- A gift doesn't work with obfuscated code.

Taint Checking

Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on artifact package during this paper, James Newsom and Dawn Song propose dynamic taint analysis for automatic detection of write attacks. This approach doesn't want ASCII text file or special compilation for the monitored program. To demonstrate this idea, Taint Check mechanism is enforced. Taint Check mechanism might improve automatic signature generation in many ways that. Taint CheckSurn [4] out no false positives for any of the numerous completely different programs that are tested. It monitors the execution of a program at a fine grained level, Taint Check will be won't to give further data concerning the attack. Taint Check is especially helpful in associate degree automatic signature generation system, it will be won't to change linguistics analysis primarily based signature generation, enhance content pattern extraction primarily based signature generation, and verify the standard of generated signatures.

Inspector Gadget

Unfortunately, malicious software is still an unsolved problem and a major threat on the Internet. An important component in the fight against malicious software is the analysis of malware samples:

Only if an analyst understands the behavior of a given sample, she can design appropriate countermeasures. Manual approaches are frequently used to analyze certain key algorithms, such as downloading of encoded updates, or generating new DNS domains for command and control purposes.

In this paper, we present a novel approach to automatically extract, from a given binary executable, the algorithm related to a certain activity of the sample. We isolate and extract these instructions and generate a so-called gadget, i.e., a stand-alone component that encapsulates a specific behavior. We make sure that a gadget [5] can autonomously perform a specific task by including all relevant code and data into the gadget such that it can be executed in a self-contained fashion.

Gadgets are useful entities in analyzing malicious software: In particular, they are valuable for practitioners, as understanding a certain activity that is embedded in a binary sample (e.g., the update function) is still largely a manual and complex task. Our evaluation with several real-world samples demonstrates that our approach is versatile and useful in practice
Some of the drawbacks are

- They are not able to recover the transient cryptographic secrets
- It can't accurately pinpoint the location, size, and boundary of the input, the output.

Architecture

Cipher X Ray will accurately pinpoint the boundary of individual science operation from multiple nested science operations. This permits it to recover transient science secrets that solely exist in between nested science operations.

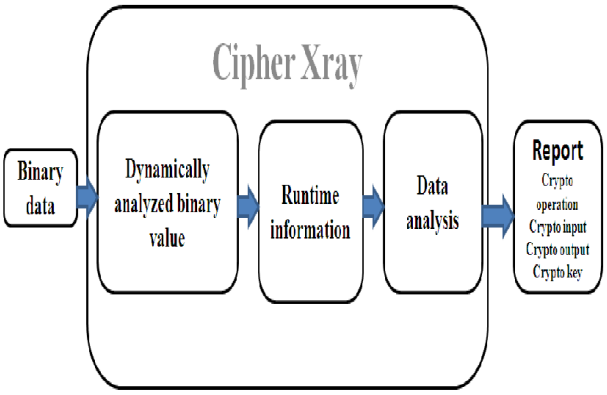

Figure 1. Architecture of Cipher X ray

Cipher X Ray is meant upon the avalanche result, that refers to the fascinating property of all science algorithms (e.g., public key science algorithms, hash functions) such a small modification (e.g., flipping one bit) within the input would cause vital changes within the output. Another nice feature of the avalanche result is that it permits us to accurately pinpoint the placement, size and boundary of each the input and output buffer.

MATERIALS AND METHODS

A. Avalanche Effect

In this paper we use the technique called *avalanche effect*, that refers to the fascinating property of all science algorithms (e.g., public key science algorithms, cruciate science algorithms, hash functions) specified a small amendment (e.g., flipping one bit) within the input would cause important changes (e.g., 0.5 the output bits flip) within the output.

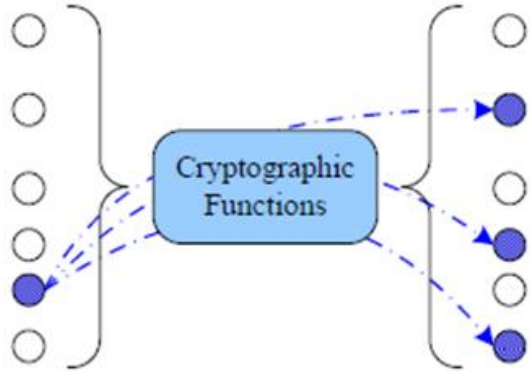

Figure 2. Avalanche effect of Cryptographic operation

Another nice feature of the avalanche impact is that it permits us to accurately pinpoint the placement, size and boundary of each the input and output buffers. Cipher Ray, a completely unique binary analysis framework that may mechanically establish and recover the science operations and transient secrets from the execution of doubtless obfuscated binary workable.

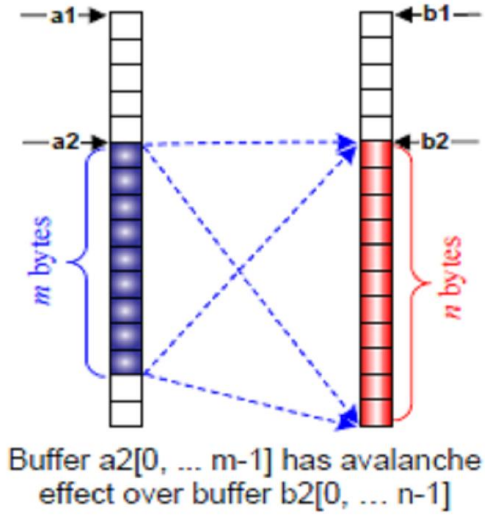

Figure 3. Taint Contribution Rate between buffers

Given two continuous buffers a[0, . . . m − 1] of m bytes and b[0, . . . n−1] of n bytes, if the knowledge be due any computer memory unit a[i] (i ∈ [0, m − 1]) touches one or additional bytes in buffer b[0, . . . n−1], we are saying a[i] taints bytes in buffer b[0, . . . n−1]. If there exists the avalanche impact from buffer a[0, . . . m − 1] to buffer b[0, . . . n − 1], then any bit in buffer a[0, . . . m − 1] should taint regarding 4n random bits in buffer b[0, . . . n − 1]. In addition, the bits in buffer b[0, . . . n − 1] tainted by every bit in buffer a[0, . . . m − 1] ought to be cohesive at intervals buffer b[0, . . . n − 1]. In different words, any computer memory unit a[i] would go away no quite $8n/2^8$=n/32 bit stainless in buffer b[0, . . . n − 1]. The chance that any computer memory unit a[i] would go away one computer memory unit in buffer b[0, . . . n − 1] stainless is

$$\left(\frac{\binom{8n-8}{4n}}{\binom{8n}{4n}}\right)^8 < \left(\frac{1}{2}\right)^{64} \quad (\text{for } n \geq 2)$$

Therefore, if there exists the avalanche result from the m-byte buffer a to the n-byte buffer b, then each computer memory unit in buffer a would taint nearly each computer memory unit in buffer b.

B. Detecting Block Cipher Modes of Operation

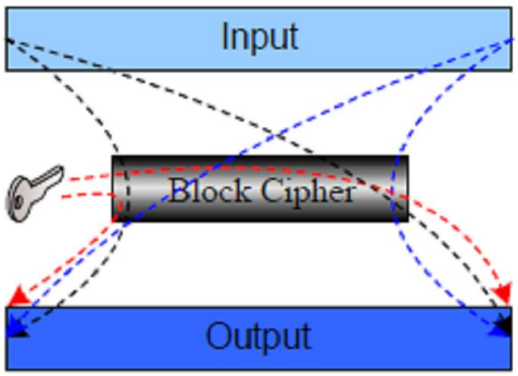

Figure 4. Block Cipher

When the input to a block cipher is longer than its block size, the input must be divided into multiple blocks before it is processed by the block cipher. The block cipher mode of operation determines however the block cipher operation are going to be applied to multiple blocks. Completely different mode of operation exhibits completely different pattern of the avalanche result between the input blocks and therefore the output blocks. this offers us a way to discover and differentiate the block cipher modes of operation.

Table 1. Patterns of the avalanche effect of different modes of operation

Mode of Operation	Encryption	Decryption
ECB	$\langle 1:n \rangle$	$\langle 1:n \rangle$
CBC	$\langle 1:n,\ 1:n,\ 1:n,\ldots \rangle$	$\langle 1:n,\ 1:1 \rangle$
CFB	$\langle 1:1,\ 1:n,\ 1:n,\ldots \rangle$	$\langle 1:1,\ 1:n \rangle$
OFB	$\langle 1:1 \rangle$	$\langle 1:1 \rangle$

Given input block *X* [0,…,n-1] and output block Y [0,…,n-1], the pattern of the avalanche impact between them might be:

1:1 each computer memory unit X[i] (i∈ [0,n-1]) from the input block X[0,…, n-1] has 100 percent contribution rate to solely computer memory unit Y [i] within the output block Y [0,…,n-1].

1: n each computer memory unit X[i] (i ∈ [0, n-1]) from the input block X[0,..,n-1] has 100 percent contribution rate to each computer memory unit Y [j] (j∈ [0,n-1]) within the output block Y [0,…,n-1].

Table1 shows the patterns of the avalanche impact between the input blocks and output blocks below completely different modes of operation. The cryptography and decipherment in ECB (Electronic Codebook)mode have actual an equivalent pattern of the avalanche effect between the input blocks and output blocks: <1:n>.Specifically, each computer memory unit in an exceedingly given input block impacts each byte within the corresponding output block and it's no impact on the other output block. The pattern of the avalanche impact in CBC (Cipher Block Chaining) mode cryptography is <1:n,1:n,1:n> in that each computer memory unit in an exceedingly given input block impacts every computer memory unit in the corresponding output block and every one the following output blocks. The pattern of the avalanche impact in CBC(Cipher Block Chaining) mode decipherment is <1: n,1:1> in that each computer memory unit in an exceedingly given input block impacts each computer memory unit in the corresponding output block and just one computer memory unit within the subsequent block. The pattern of the avalanche impact in CFB (Cipher Feedback)mode cryptography is <1:1,1:n,1:n> therein every computer memory unit in an exceedingly given input block impacts just one computer memory unit within the corresponding output block and each computer memory unit altogether ulterior output blocks.

The pattern of the avalanche impact in CFB(Cipher Feedback) mode decipherment is <1:1,1:n> therein every computer memory unit in an exceedingly given input block impacts just one computer memory unit within the corresponding output block and each computer memory unit within the ulterior output block. The cryptography and secret writing in OFB (Output Feedback) mode have actual an equivalent pattern of the avalanche impact between the input blocks and output blocks: <1:1>. Specifically, every computer memory unit during a given input block impacts just one computer memory unit within the corresponding output block and it's no impact on the other output block. Therefore, we are able to faithfully distinguish between ECB, CBC, CFB and OFB modes of operation supported the patterns of the avalanche impact as long because the block cipher input (and output) is not any but three blocks. Specifically, we make no distinction between block cipher in OFB (and CTR) mode and stream cipher since OFB (and CTR) mode will flip a block cipher into a stream cipher.

C. RSA encryption and Decryption

We have chosen to use RSA in our experiment on the general public key cryptography analysis. we've got used two programs in the experiment: one to cipher a brief message exploitation

```
RSAPrivateKey ::= SEQUENCE {
    version           Version,
    modulus           INTEGER,  -- n
    publicExponent    INTEGER,  -- e
    privateExponent   INTEGER,  -- d
    prime1            INTEGER,  -- p
    prime2            INTEGER,  -- q
    exponent1         INTEGER,  -- d mod (p-1)
    exponent2         INTEGER,  -- d mod (q-1)
    coefficient       INTEGER,  -- (inverse of q) mod p
    otherPrimeInfos   OtherPrimeInfos OPTIONAL
}
```

Figure 5. RSA Private Key Format

RSA public key and also the different to decipher the cipher text exploitation the corresponding RSA non-public key. we've got used 1024-bit RSA key within the experiment, and also the short message is a smaller amount than 1024 bits. Our primary goal is to recover the RSA non-public key, whose format is outlined in PKCS#1 [30] as shown in Figure 4.

By analyzing the execution log of the RSA public key encryption program, Cipher X Ray has been ready to establish both the modulus n and also the public exponent e of the general public key.

Table 2. RSA private key fields identified by cipherxray

Address Range	Offset Range	Size	Field
04186028 - 04186028	0 - 0	1	
0418602d - 0418602d	5 - 5	1	
04186030 - 04186031	8 - 9	2	
04186033 - 041860b2	11 - 138	128	n
041860b4 - 041860b7	140 - 143	4	e
041860b9 - 041860bc	145 - 146	2	
0418613d - 0418613d	277 - 277	1	
0418613f - 0418617e	279 - 342	64	p
04186180 - 04186180	344 - 344	1	
04186182 - 041861c1	346 - 409	64	q
041861c3 - 04186203	411 - 475	65	$d \bmod (p-1)$
04186205 - 04186205	477 - 477	1	
04186207 - 04186246	479 - 542	64	$d \bmod (q-1)$
04186248 - 04186248	544 - 544	1	
0418624a - 04186289	546 - 609	64	$q^{-1} \bmod p$

Table 2 shows the RSA non-public key fields known by Cipher X Ray from analyzing the execution log of the RSA private key secret writing program. It shows that Cipher X Ray has with success known the length field and worth field of modulus n, public exponent e, prime1 p, prime2 q, exponent1 d mod (p-1), exponent2 d mod (q-1) and constant q^{-1} mod p. Besides, the primary computer memory unit of the key, a kind field, the length field of the version and also the length field of the non-public exponent are known. However, the worth field of the private exponent d has not been known. Further investigation shows that the RSA implementation in Open SSL uses the

Chinese Remainder Theorem to calculate the modulo mathematical operation that doesn't use the private exponent d in any respect. Therefore, Cipher X Ray doesn't see any avalanche impact from the non-public exponent d. This further proves the accuracy of Cipher X Ray. Having two prime numbers p and Q known, it's trivial to derive the non-public output block. The cryptography and secret writing in OFB (Output Feedback) mode have actual an equivalent pattern of the avalanche impact between the input blocks and output blocks: <1:1>. Specifically, every computer memory unit during a given input block impacts just one computer memory unit within the corresponding output block and it's no impact on the other output block. Therefore, we are able to faithfully distinguish between ECB,CBC,CFB and OFB modes of operation supported the patterns of the avalanche impact as long because the block cipher input (and output) is not any but 3 blocks. Specifically, we make no distinction between block cipher in OFB (and CTR) mode and stream cipher since OFB (and CTR) mode will flip a block cipher into a stream cipher.

RESULTS AND DISCUSSIONS

Current implementation of Cipher X Ray assumes all the input, output and cryptological keys ar unbroken in continuous memory buffers. To the simplest of our information, all wide used software implementations (e.g., Open SSL) of cryptological algorithms follow this convention as a result of performance thought. In addition, all the malwares we've got experimented have unbroken the cryptological secrets in continuous buffer as well.

In principle, deliberate obfuscation (e.g., put the cryptological input, output and key in discontinuous buffers) will not take away the avalanche impact among the cryptological input, key and output, however makes it more durable to notice the avalanche impact.

Conclusion

This approach could persuade be quick and economical technique to research the characteristics of all science operations. It's been ready to notice public key cryptography, block cipher, and hash operations and pinpoint precisely once and wherever the science input, output, and keys are going to be within the memory although they exist for under a couple of microseconds.

While this new capability helps higher analyze subtle malwares protected by sturdy scientific discipline algorithms. For future enhancement the particular data can be retrived by removing the malware.

Acknowledgment

We would like to thank the anonymous reviewers for their comments and suggestions that helped us improve the quality of our paper.

REFERENCES

Gullasch, D., Bangerter, E. and Krenn, S. 2011.Cache Games Bringing Access-Based Cache Attacks on AES to Practice. In Proceedings of the 2011 IEEE Symposium on Security and Privacy (S and P 2011), pages 490–505, Oakland, CA, May 2011. [30] *J. Jonsson and B*. Kaliski. Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography February 2003.

KeePass, X. Cross Platform Password Manager. http://www.keepassx.org.

Kolbitsch, C., Holz, T., Kruegel, C. and Kirda, E. May 2010. Inspector Gadget: Automated Extraction of Proprietary Gadgets from Malware Binaries. In Proceedings of the 2010 IEEE Symposium on Security and Privacy(S and P 2010), pages 29–44. IEEE.

Newsome, J. and Song, D. February 2005. Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software. In Proceedings of the 12th Network and Distributed System Security Symposium (NDSS 2005).

Sharif, M., Lanzi, A., Giffin, J. and Lee, W. Feb 2008. "Impeding Malware Analysis Using Conditional Code Obfuscation," Proc. 15th Network and Distributed System Security Symp. (NDSS '08).

The Open SSL Project. http://www.openssl.org/.

Wang, Z., Jiang, X., Cui, W., Wang, X. and Grace. M. 2009. Re Format: Automatic Reverse Engineering of Encrypted Messages. In Proceedings of the 14th European Symposium on Research in Computer Security (ESORICS 2009), pages 200–215, September 2009.

Yan, W., Zhang, Z. and Ansari, N. October 2008. Revealing Packed Malware. IEEE Security and Privacy, 6(5):65–69.

****