Work-in-Progress: Sequence-Crafter: Side-Channel Entropy Minimization to Thwart Timing-based Side-Channel Attacks

Abhijitt Dhavlle, Sahil Bhat, Setareh Rafatirad, Houman Homayoun and Sai Manoj P D
George Mason University, Fairfax, VA, USA
{adhavlle,sbhat6,srafatir,hhomayou,spudukot}@gmu.edu

ABSTRACT

The hardware security domain in recent years has experienced a plethora of side-channel attacks (SCAs) with cache-based SCAs being one of the dominant threats. These SCAs function by exploiting the side-channels which invariably leak important data during an application's execution. Shutting down the side channels is not a feasible approach due to various restrictions it would pose to system performance. To overcome such concerns and protect the data integrity, we introduce *Sequence-Crafter* (SC) in this work. The proposed Sequence-Crafter (SC) aims to minimize the entropy in the side channel leaked information rather than attempting to close the side-channels. To achieve this, we introduce carefully crafted perturbations into the victim application which will be randomly activated to introduce perturbations, thus resulting in misleading information which looks legit that will be observed by the attacker. This methodology has been successfully tested for Flush+Reload attack and the key information observed by the attacker is seen to be completely futile, indicating the success of proposed method.

ACM Reference Format:
Abhijitt Dhavlle, Sahil Bhat, Setareh Rafatirad, Houman Homayoun and Sai Manoj P D. 2019. Work-in-Progress: Sequence-Crafter: Side-Channel Entropy Minimization to Thwart Timing-based Side-Channel Attacks. In *2019 International Conference on Compliers, Architectures and Synthesis for Embedded Systems Companion (CASES'19 Companion), October 13–18, 2019, New York, NY, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3349569.3351543

1 INTRODUCTION

Security threats utilize the side-channels or covert channels to obtain the secret information from the system and are passive in nature. Side-channel attacks are a class of attacks that primarily exploit security of computing systems based on the obtained side-channel information as a result of design vulnerabilities rather than the exploits in the application [1]. Side-channels are inherent in any computing system and the foremost challenge in defending against side-channel attacks is that they cannot be completely terminated. Our work focuses on cache-based side-channel attacks which have exacerbated [2] with the features introduced in modern computing systems such as memory-sharing, co-location of applications, which were introduced for an efficient resource management and higher throughput. However, a large number of cache-based side-channel attacks rely on the timing information to determine the cache-access (hit or miss) patterns in order to obtain the accessed address and eventually the secret key from the cache [3]. For instance, Flush+Reload [3] depends on the assumption that the victim and the attacker share the same memory space and utilizes the cache-access timing information to retrieve the secret key from system.

To address the challenges of cache side-channel attacks, techniques such as static cache partitioning [4], partition locked cache [5], non-monopolizable (nomo) cache architectures [6] are proposed. These techniques can tremendously reduce the interference between attacker and victim's memory access, thus providing better defense. However, adopting such techniques require alterations in the cache design and also leads to performance degradation [4]. To overcome the limitations of the cache-partitioning, randomization of cache architectures are introduced. Conventional fully associative cache is one of the preliminary randomization based cache. Despite its security benefits, this technique incurs large delays and is power hungry. In a similar way, random permutation cache [5], newcache [7], random fill cache [8], and random eviction cache [4] strategies are implemented. Compared to cache-partitioning, randomization based solutions have shown higher robustness, yet the challenge of performance degradation is not addressed.

In this work, we introduce Sequence-Crafter, a defense for timing-based side-channel attacks such as Flush+Reload and Prime+Probe. In contrast to the existing works that focus on architectural changes or perturbing cache lines, the proposed Sequence-Crafter primarily focuses on minimizing the entropy of the side-channel information obtained by the attacker without interfering with the original functionality of the victim application. In the Sequence-Crafter the original application is wrapped with a protective application that is able to facilitate the perturbation of the cache-access timing information obtained by the attacker under the constraints of the achieved information looking similar to the normal timing information, yet leading to a wrong key. The proposed Sequence-Crafter introduces perturbations in the sequence (timing information) by executing dummy functions that do not affect the result of the key for the victim, but scrambling patterns observed by the attacker thereby reducing entropy and dissuading the attack. The proposed Sequence-Crafter technique is evaluated against Flush+Reload with different keys and the proposed Sequence-Crafter is found to be successful in defending against the attacks without any assumptions on the attacker capabilities.

2 SEQUENCE-CRAFTER

Our proposed defense mechanism, Sequence-Crafter is shown in Figure 1. Figure 1(a) shows the traditional Flush+Reload attack eavesdropping on the sequence of operations of a cryptographic application. The attack observes the sequence and then infers the

ACM ISBN 978-1-4503-6925-1/19/10…$15.00
https://doi.org/10.1145/3349569.3351543

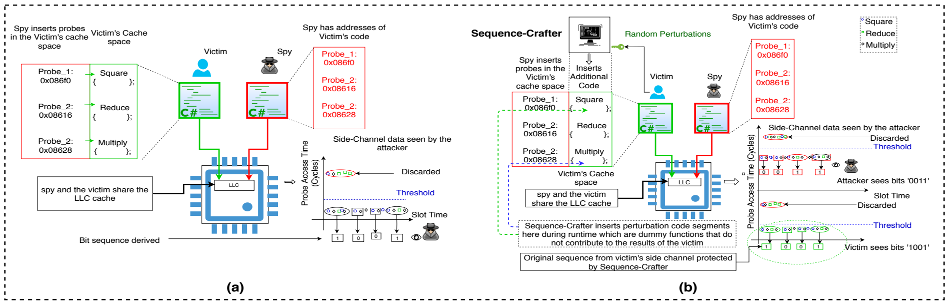
Spy inserts probes in the Victim's cache space
Victim's Cache space
Probe_1: 0x086f0
Square { };
Probe_2: 0x08616
Reduce { };
Probe_2: 0x08628
Multiply { };
Victim
Spy
Spy has addresses of Victim's code
Probe_1: 0x086f0
Probe_2: 0x08616
Probe_2: 0x08628
spy and the victim share the LLC cache
LLC
Side-Channel data seen by the attacker
Probe Access Time (Cycles)
Discarded
Threshold
Slot Time
Bit sequence derived
1 0 0 1
(a)
Sequence-Crafter
Random Perturbations
Inserts Additional Code
Spy inserts probes in the Victim's cache space
Victim's Cache space
Square
Reduce
Multiply
Attacker sees bits '0011'
Sequence-Crafter inserts perturbation code segments here during runtime which are dummy functions that do not contribute to the results of the victim
Original sequence from victim's side channel protected by Sequence-Crafter
Victim sees bits '1001'
(b)

Figure 1: (a) Traditional Flush+Reload attack on encryption algorithm where all the data leaked via side-channel is accessible to the attacker; (b) Victim is wrapped with Sequence-Crafter that injects perturbation code in the victim during run-time to reduce and perturb the sensitive information leaked thereby making SCAs laborious and time-consuming.

bits of the exponent used for encryption. The covert channel or the side-channel is not a 100% transparent medium, but rather it is a noisy channel due to other cache accesses. To reduce this noise, the attacker runs the attack multiple times and filters out the noise. Hence, without any protection barrier, the attacker is able to capture the leaked information. With our proposed Sequence-Crafter , the original sequence of operations is perturbed by calling dummy functions, the code for which is inserted in runtime by the Sequence-Crafter as it knows the memory addresses of the operations. There is no change made on the attacker's end. The attacker inserts the probes in the functions and tries to observe the sequence. The Sequence-Crafter calls fake functions which do not necessarily intervene in the encryption operation but rather just executes the function and exits after a while thus giving an impression to the attacker that the functions were called by the encryption application. This leads to wrong sequence of operations as observed by the attacker and hence a wrong captured key thwarting the attack.

Algorithm 1 Pseudocode illustrating the implementation of Sequence-Crafter

```
Input: Private Ecryption Key
Output: Decoded Incorrect Encryption Key
 1: Victim Program() {Victim pseudo code w/ Sequence-Crafter }
 2: Square func()      {        Probe 1 <= memory address }
 3: Multiply func()
 4:       {      Probe 2 <= memory address of Multiply func
 5:       - - Dummy call to Square func() - - }
 6: Reduce func()      {        Probe 3 }
 7: Attacker Program() {Code that decodes the secret key}
 8:       Loop 1: load i < n
 9:       clflush (Probe 1) ; clflush (Probe 2) ; clflush (Probe 1);
10:       Reloading time(t)
11:       jump Loop1; end;
12:    cmp t # threshold time(th)
13:    if( t > th) => Cache miss    else if( t < th) => Cache hit
14:    Incorrect Secret Key is Deduced
```

Algorithm 1 describes the pseudocode for the implementation of Sequence-Crafter . The traditional Flush+Reload attack on encryption application is not shown due to space restrictions where Line 5 would be absent, everything else remains pretty much the same. The attacker inserts probes in the Square, Reduce and Multiply function of the algorithm so whenever the function is called and hence the memory line is accessed, it knows and keeps track of it. With Sequence-Crafter, from within the Multiply function a fake call to the Square function is made which tricks the attack code into thinking that a real call was made. The dummy call is made multiple times to ensure that the call achieves a cache-hit as the attack code accepts only cache hit as a condition for considering or rejecting the sequence of Square, Reduce or Multiply.

Table 1: Impact of Sequence-Crafter on key extraction

Flush+Reload Attack	RSA and RSA	DSA and Elgamal
Original Key	F9D2EDC5	3DA77005
Without perturbation (seen by the spy)	F9D2EDC5	3DA77005
With Crafter (seen by victim)	F9D2EDC5	3DA77005
With Crafter(seen by spy)	F1D2ADC7	3FA7710D

3 RESULTS

The impact of the proposed Crafter on the data leakage in the covert channel can be clearly seen in Table 1. We ran our experiments with two types of encryption methods - RSA and RSA where both the public and private keys are RSA type, and DSA and Elgamal where the private key was Elgamal type. For discussion, we have only considered a short version of the original keys. Without the Crafter, the spy (attacker) is able to view the private key as shown in Table 1. However, with the proposed Crafter, the attacker observes a wrong sequence and hence the incorrect key is deduced compared to the original key used by victim, as in Table 1. The experiments so far have shown promising results to thwart SCAs.

4 CONCLUSION AND FUTURE WORK

We discussed how SCAs are a threat to data integrity and confidentiality and mentioned about the works in the past that have proposed mitigation techniques against the SCAs. But, there are many shortcomings that needed to be addressed and Sequence-Crafter can prove to be a feasible panacea. Our experimental results mentioned in Table 1 have shown promising outcomes in thwarting timing-based cache side-channel attacks and we are working on developing the further version of the Sequence-Crafter where random perturbations along with the number of bits to be perturbed would also be accommodated.

REFERENCES

[1] F. Brasser et al., "Advances and throwbacks in hardware-assisted security: Special session," in *Proceedings of the Inter. Conf. on CASES*, 2018.
[2] S. M. P. D. et al., "Adversarial attack on microarchitectural events based malware detectors," in *Proceedings of the DAC*, 2019.
[3] Y. Yarom and K. Falkner, "Flush+reload: A high resolution, low noise, L3 cache side-channel attack," in *USENIX Conf. on Security Symposium*, 2014.
[4] Z. He and R. B. Lee, "How secure is your cache against side-channel attacks?" in *IEEE/ACM Int. Symp. on Microarchitecture*, 2017.
[5] Z. Wang and R. B. Lee, "New cache designs for thwarting software cache-based side channel attacks," in *Int. Symp. on Computer Architecture*, 2007.
[6] L. Domnitser, A. Jaleel, J. Loew, N. Abu-Ghazaleh, and D. Ponomarev, "Non-monopolizable caches: Low-complexity mitigation of cache side channel attacks," *ACM Trans. Archit. Code Optim.*, vol. 8, no. 4, pp. 35:1–35:21, Jan 2012.
[7] F. Liu, H. Wu, K. Mai, and R. B. Lee, "Newcache: Secure cache architecture thwarting cache side-channel attacks," *IEEE Micro*, vol. 36, no. 5, pp. 8–16, Sep. 2016.
[8] F. Liu and R. B. Lee, "Random fill cache architecture," in *IEEE/ACM Int. Symp. on Microarchitecture*, 2014.