Performance Counter Monitor 를 이용한 FLUSH+RELOAD 공격 실시간 탐지 기술

조종현, 김태현, 신영주
광운대학교 컴퓨터정보공학부 정보 및 사이버보안 연구실
e-mail : whwhdgus94@naver.com, taehyun9203@gmail.com, yjshin@kw.ac.kr

Real-time detection on FLUSH+RELOAD attack using Performance Counter Monitor

Jong-Hyeon Cho, Tae-Hyun Kim, Youngjoo Shin
Information and Cyber Security Lab, School of Computer and Information Engineering
Kwangwoon University

요 약

캐시 부채널 공격 중 하나인 FLUSH+RELOAD 공격은 높은 해상도와 적은 오류로 그 위험성이 높고, 여러가지 프로그램에서도 적용되어 개인정보의 유출에 대한 위험성까지 증명 되었다. 따라서 이 공격을 막기위해 실시간으로 감지 할 수 있어야 할 필요성이 있다. 본 연구에서는 4 가지 실험을 통하여 이 FLUSH+RELOAD 공격을 받을 때 PCM(Performance Counter Monitor)를 사용해 각각의 counter 들의 값의 변화를 관찰하여 3 가지 중요한 요인에 의해 공격 탐지를 할 수 있다는 것을 발견하였다. 이를 이용하여 머신 러닝의 logistic regression 과 ANN(Artificial Neural Network)를 사용해 결과에 대한 각각 학습을 시킨 뒤, 실시간으로 공격에 대한 탐지를 할 수 있는 프로그램을 제작하였다. 일정한 시간동안 공격을 진행하여 모든 공격을 감지하는데 성공하였고, 상대적으로 적은 오탐률을 보여주었다.

1. 서론

FLUSH+RELOAD[1,2]는 최근 Meltdown 같은 화두가 되는 캐시 부채널 공격 중 하나로써, 다른 공격들에 비해 높은 해상도와 낮은 잡음을 가진 강력한 공격이다. 이 공격은 CPU 의 LLC(Last Level Cache) 를 이용하여 공격을 진행한다. CPU 캐시는 데이터 접근을 빠르게 하려고 CPU 내부와 인접한 곳에 탑재하는 작은 메모리이다. FLUSH+RELOAD 공격이 진행되면 LLC 라인의 데이터가 비워지게 되어 캐시 miss 가 많이 발생하게 되는데, 이것을 이용하여 공격에 대한 탐지가 가능할 것임을 예측하였다. 캐시 hit 와 miss 에 대한 정보를 확인하기 위해 우리는 인텔 프로세서에서 제공하는 PCM(Performance Counter Attack)을 사용하였다. PCM 을 사용하여 FLUSH+RELOAD 공격 여부에 따라 각각의 카운트를 확인하고, 공격을 받을 경우 캐시 hit 와 miss 의 차이를 이용하여, 공격탐지를 진행하려 한다. 이를 위해 머신 러닝을 이용하여 컴퓨터를 학습시켜 백그라운드에서 실행시킬 수 있는 프로그램을 제작하려 한다.

본 논문의 구성은 다음과 같다. 2 장에서는 FLUSH+RELOAD 캐시 부채널 공격, PCM, logistic regression 에 대한 배경지식을 설명한다. 3 장에서는 관련된 4 가지 실험과 실험에 관한 결과를 나타낸다. 4 장에서는 3 장의 결과를 이용한 탐지 프로그램에 대한 설명을 진행한다. 마지막 5 장에서는 결론 및 향후 연구계획을 기술한다.

2. 배경지식

2.1 FLUSH+RELOAD ATTACK

FLUSH+RELOAD ATTACK 은 공격자(spy)와 공격 대상자(victim)가 같은 인텔 프로세서를 공유한 환경에서 페이지 공유와 인텔 아키텍처의 캐시 inclusive 성질을 이용한 `clflush` 명령어(L3 캐시를 포함한 모든 레벨의 캐시에 들어있는 특정 메모리 라인을 제거하는 명령어)를 이용한 공격이다. FLUSH+RELOAD 공격은 크게 세 단계로 나누어지게 된다. 공격의 첫 단계는 spy 가 감시한 특정 메모리 라인을 모든 레벨의 캐시에서 제거한다. 두 번째 단계에서 spy 는 victim 이 세 번째 단계 전에 특정 메모리 라인에 접근할 수 있도록 기다린다. 세 번째 단계에서는 spy 는 특정 메모리 라인을 reload 하고 reload 한 시간을 측정한다. 만약 기다리는 시간 동안 victim 이 특정 메모리 라인에 접근했다면, 데이터는 캐시 안에서 이용할 수 있고, reload 시간은 짧아진다. 반대로 기다리는 시간 동안 victim 이 특정 메모리 라인에 접근을 안 했다면, 데이터는 메인 메모리로부터 가져오므로, 시간은 오랜 시간이 걸리게 된다. 이 시차를 이용하여 해당 비트 값

을 알아내어 어떠한 입력을 받았는지 알아낼 수 있는 간단하면서 강력한 공격이라 할 수 있다. 이 공격을 이용하여 개인 키 값과 응용 프로그램에서의 활동 내용 등 다양한 정보를 추출할 수 있다.

2.2 Performance Counter Monitor (PCM)

Intel Performance Counter Monitor(PCM) [3]는 인텔 프로세서의 퍼포먼스 카운터 값을 보여주기 위한 도구로써 PAPI 와 유사하다. PAPI 는 Performance application programming interface(PAPI)의 약자로써 이것은 프로세서 아키텍처에서 보이는 퍼포먼스 카운터에 접근하기 편한 인터페이스를 의미한다. PAPI 는 CPU 카운터에 제한되지 않고, CUDA, 네트워크와 같은 다른 요소에서도 동작한다. Intel PCM 과 PAPI 의 차이점은 Intel PCM 은 오직 Intel 하드웨어에서만 지원을 하지만, QPI(Quick Path Interconnect)와 같은 Intel 프로세서의 UNCORE 부분도 볼 수 있다.

2.3 Logistic Regression

단일 데이터(x)에 대한 Logistic Regression[4]은 분류하고자 하는 데이터에 대한 가중치(W)를 곱한 뒤 바이어스(b) 더한 값을 linear H(x)이라 하면, logistic regression 의 특성인 binary classification 을 위해 linear H(x)는 모든 값을 0~1 사이로 만들어져야 한다. sigmoid 함수(g(z))는 0~1 사이로 값을 결정하는 역할을 하며, sigmoid 함수에 대입하여 나온 값 sigmoid(linear H(x))를 가정 값(H(x))이라고 부르며 이 값의 범위는 0~1 이다. 만약 이 가정 값이 0.5 보다 크면 예측 값은 1 이며, 0.5 보다 작으면 예측 값이 0 이다. 코스트 함수인 C(H(x), y)의 값은 실제 값과 측 값에 대한 평균을 의미한다. 그래서 실제 값(y)과 예측 값(H(x))이 코스트 함수에 대입해서 실제의 값과 예측 값이 같거나 혹은 비슷하면 cost 의 값은 작아지고, 예측 값이 틀리면 cost 값이 커지는 것을 의미한다.

$$\text{linear H(x)} = \text{Wx} + \text{b}$$

$$\text{g(z)} = \frac{1}{(1 + e^{-z})}$$

$$\text{H}(x) = \frac{1}{1 + e^{-(Wx+b)}}$$

$$\text{C(H(x), y)} = \begin{cases} -\log(H(x)) & (y = 1) \\ -\log(1 - H(x)) & (y = 0) \end{cases}$$

2.4 ANN(Artificial Neural Network)

인공 신경망(ANN)[5]은 생물학의 신경망에서 영감을 얻은 학습 알고리즘이다. 일반적으로 사용되는 인공신경망은 입력층과 은닉층 그리고 출력층 이렇게 세 가지 층으로 구분된다. 그리고 각 층은 node 들로 구성되어 있다. 입력층은 예측 값을 추측하기 위한 예측변수의 값들을 입력하는 역할을 한다. 은닉층은 모든 입력 노드부터 입력 값을 받아 가중 합을 계산하고, 이 값을 전이함수에 적용하여 출력층에 전달하게 된다. 각 입력 노드와 은닉 노드들은 모두 가중치를 가지는 망으로 연결되어 있으며 은닉 노드와 출력 노드도 마찬가지로 연결되어 있다. 이 가중치를 연결 강도로 표현되며 무작위로 초기에 주어졌다가 예측값에 근접한 값으로 조정되게 된다. 이 기법을 사용하여 우리는 은닉층을 2 개 사용하여 훈련을 진행하도록 했다.

3. 실험

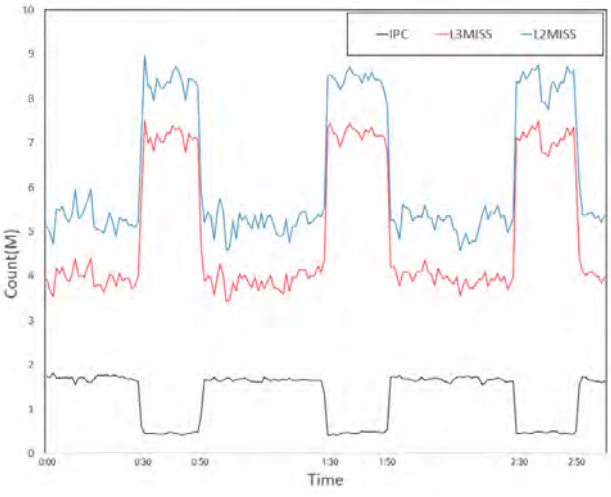
3.1 환경 설정

PCM 의 어떠한 카운터가 FLUSH+RELOAD 공격을 받을 때 영향을 받는지 몇 가지 실험을 위해 spy, victim 로 나누어 컴퓨터 두 대로 진행했다. spy 컴퓨터는 Intel® Core™ i5-5250U 1.6GHz 프로세서를 탑재하고 8GB DDR3 메모리를 사용하는 MacBook Air(2015 년도 모델)를 사용하였다. victim 컴퓨터는 Intel® Xeon® CPU E5-2620 v4 2.1GHz 프로세서를 탑재하고 128GB DDR3 메모리를 사용하는 Asus X99-E WS 서버 컴퓨터를 사용하였다. 두 컴퓨터의 운영체제는 각각 OS X High Sierra, Ubuntu 16.04 LTS 으로 같은 Unix 계열 운영체제를 사용하고 있다. 공격을 위해 Mastik 의 FLUSH+RELOAD 공격 프로그램을 사용했는데, Mastik 은 여러가지 캐시 부채널 공격들을 한곳에 모아 놓은 도구이다. 이 프로그램을 사용한다면 경로설정만으로 FLUSH+RELOAD 공격을 진행할 수 있다. 해당 프로그램은 GnuPG-1.4.13 에서 개인 키를 가져오기 위한 것이기 때문에, victim 의 경로에 해당 프로그램을 설치하여 공격이 올바르게 진행될 수 있도록 했다.

3.2 공격 및 관찰

FLUSH+RELOAD 공격은 같은 프로세서 위에서 공격이 이루어져야 하므로 spy 의 컴퓨터에서 SSH 접속을 통해 victim 컴퓨터의 계정으로 접속하여 공격을 진행할 필요가 있었다. 따라서 공격용 계정을 따로 만들어 공격을 진행하였다. 공격에 대한 PCM 의 변화 관찰을 위해 비디오재생, 음악재생, LibreOffice Calc, Firefox 네 가지 응용프로그램을 각각 실행시켜 실험을 진행했다. 각 응용프로그램의 카운터들이 모두 달랐기 때문에 3 분의 시간을 두고 총 3 번의 공격을 하여 공통으로 어떠한 변화가 있는지 관찰했다.

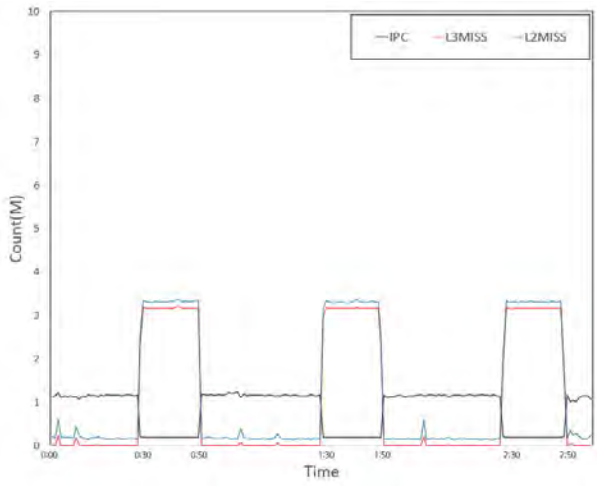
<Fig. 1> 비디오 재생 중 공격 시 IPC, L3 Miss, L2 Miss count 값



첫 번째 실험은 비디오 재생 중 공격을 진행하였다. 시작을 기준으로 3 분간 PCM 을 이용하여 측정하였는데, 분마다 30 초부터 50 초 동안 공격을 했을 때 큰 변화를 보인 카운터 값은 L3 Miss, L2 Miss, IPC 로 3 가지를 발견할 수 있었다. 그림 <Fig. 1>을 확인하면 공격을 하지 않을 때 L3 Miss 와 L2 Miss 는 각각 4M 와 5~6M 의 값을 가지며 IPC 는 2M 가 조금 안 되는 것을 볼 수 있다. 공격이 진행되는 30 초부터 50 초에서는 L3 Miss 의 카운터는 7M, L2 Miss 는 8.5M, IPC 는 0.5M 로 변하는 것을 확인할 수 있다.

두 번째 실험은 음악 재생 중 공격을 진행하였다. 첫 번째 실험과 같은 조건에서 실험을 진행했을 때 역시 3 가지 카운터가 크게 변하는 것을 볼 수 있었다. 그림 <Fig. 2>를 확인하면 공격을 하지 않을 때 L3 Miss 의 카운터는 0.1M, L2 Miss 의 카운터는 0.2M, IPC 의 카운터는 1M 정도 나타나는 것을 볼 수 있다. 공격이 진행되는 구간에는 첫 번째 실험과 같이 L3 Miss 는 3M, L2 Miss 는 3.5M, IPC 는 0.2M 로 두 카운터는 증가하고 IPC 는 감소하는 것을 볼 수 있었다.

<Fig. 2> 음악 재생 중 공격 시 IPC, L3 Miss, L2 Miss 카운터 값



세 번째, 네 번째 실험은 첫 번째 실험과 L3 Miss, L2 Miss 의 크기만 다를 뿐 L3, L2 의 급격한 증가와 IPC 감소의 공통적인 결과를 나타냈다. 네 가지 실험에서 공통으로 나타난 것은 프로그램을 실행할 때 해당 프로그램에 대한 정보를 가져오는 과정에서 캐시 miss 가 급증하는 모습을 볼 수 있었다.

이 네 가지 실험을 통해 확인할 수 있는 것은 FLUSH+RELOAD 공격을 진행할 경우 여러 가지 요인이 변하는 것을 PCM 을 통해 관찰할 수 있으며 그 중 가장 큰 변화는 L3 Miss, L2 Miss, IPC 라는 것을 볼 수 있었다. 이러한 결과가 나타나는 이유를 설명하면, FLUSH+RELOAD 공격으로 캐시 메모리가 flush 되어 캐시 miss 가 많이 발생하게 되고, 이 때문에 하나의 명령어를 사용할 때 걸리는 사이클이 매우 증가하여 IPC 가 감소한다고 판단할 수 있었다. 이 실험을 통해 다음장에 진행할 논리 회귀 머신러닝을 통한 공격예측 프로그램 제작을 진행하였다.

4. FLUSH+RELOAD 실시간 탐지

앞서 얻은 실험의 정보를 통해 PCM 의 세 가지 변화로 FLUSH+RELOAD 공격을 받고 있다는 것을 알 수 있다. 따라서 우리는 이점을 이용하여 실시간으로 해당 공격을 감지할 수 있는 프로그램을 제작하였다. 또한, 세 가지 요인 이외에 작은 변화들이 있을 것으로 판단하여 EXEC, FREQ, AFREQ, L3 Hit, L2 Hit, L3 MPI, L2 MPI 를 추가로 입력값으로 사용했다. 4 가지 응용프로그램들을 실행할 때 카운터들이 각각 다르므로 일정 시간을 두고 동시에 여러 작업을 진행하며 PCM 을 통해 카운터 값들을 CSV 파일로 추출하였다. 총 10 분의 시간 동안 4 가지 프로그램들을 실행 및 종료하며 3 번의 공격을 진행하였다. 이때 공격을 진행한 시간을 모두 표시하여 학습 입력값의 label 로 사용할 수 있도록 했다. Label 은 FLUSH+RELOAD 공격이 진행된 경우를 1, 아무 공격이 가해지지 않은 경우를 0 으로 하여 트레이닝 집합을 제작했다. 머신 러닝은 오픈소스인 Tensorflow 를 사용하여 진행했는데, 트레이닝 집합을 이용하여 logistic regression, ANN 두 가지 기법을 활용하여 학습모델을 제작했다. CSV 파일을 사용하여 훈련을 진행한 결과, 초기 학습모델의 정확도는 logistic regression 은 96.07%, ANN 은 99.05%로 ANN 이 좀 더 높은 정확도를 나타내는 것을 확인할 수 있었다. 실시간으로 감지하기 위해서는 항상 훈련할 수 없으므로 학습된 모델은 메타파일로 저장하여 입력값만 넣으면 바로 결과가 나올 수 있도록 했다. <Fig. 3>은 파이썬을 기반으로 하여 작성한 해당 프로그램의 의사코드이다. PCM 과 탐지를 동시에 실행시키기 위하여 fork 를 이용하여 서부 프로세스에서 PCM 이 실행되게 하고 다른 프로세스에서는 PCM 을 통해 생성되는 CSV 파일을 읽어 공격 여부를 판단하도록 했다. PCM 이 CSV 파일을 생성할 때 한 줄씩 누적하여 출력하기 때문에 skiprow 를 활용하여 최신의 정보를 받아올 수 있도록 했다. 줄 수가 많아지면 해당 줄을 찾는 데 시간이 걸리기 때문에 20 개의 정보가 생성될 때 마다 해당 로그 파일을 초기화해주어 줄을 찾는데 시간을 줄일 수 있도록 했다.

<Fig. 3> 실시간 탐지 프로그램 의사코드

```
1:  pid = fork()
2:  if pid == 0 then
3:      execute(PCM)
4:  else
5:      While true do
6:          f = open(Log.csv, skiprow = s)
7:          array = f.readline()
8:          result = Detection(array)
9:          if result == 1 then
10:             print(Detection message)
11:         f.close()
12:         if s == 20 then
13:             s = 0
14:             truncate(Log.csv)
```

PCM 을 이용하기 위해서 root 권한이 필요하므로 root 권한으로 학습된 두 모델로 실시간 탐지를 진행하였다. PCM 의 옵션으로 0.5 초마다 CSV 파일에 기록하도록 하고 프로그램 역시 0.5 초의 delay 를 주어 기록과 탐지의 시간을 맞춰 주었다. 각각 5 분의 시간을 두고 victim 컴퓨터로 탐지 프로그램을 실행하고 spy 컴퓨터로 3 번의 공격을 진행하여 올바른 탐지 여부를 실험해 보았다. Logistic regression 을 이용한 탐지는 3 번의 공격을 성공적으로 탐지하였고, 공격 이외의 탐지를 1 초를 기준으로 총 8 초 정도 감지하였다. ANN 을 이용한 탐지는 공격 감지에는 성공하였으나, 앞선 결과에 비해 탐지결과가 너무 늦게 나오는 것을 확인할 수 있었다. 이에 대한 이유를 추측해 보았는데, logistic regression 보다 ANN 의 연산 횟수가 매우 많으므로 이러한 결과가 나오는 것으로 볼 수 있었다. 따라서 정확도는 ANN 을 이용한 모델이 좀 더 높지만, 실시간 탐지에는 부적합하다는 것을 발견할 수 있었다.

5. 결론

본 논문에서는 PCM 을 사용하여 FLUSH+RELOAD 공격에 대한 실시간 탐지 기술을 설명하였다. 이를 위해 여러가지 실험을 통해 중요한 요인을 알아내고 해당 값들을 머신 러닝을 통해 변화에 대한 label 을 학습시켜 실시간으로 공격에 대한 탐지를 할 수 있도록 했다. 연구를 통해 만들어낸 프로그램으로 성공적으로 모든 공격에 대한 탐지를 할 수 있는 결과를 보여주었다.

하지만, 이 프로그램은 동시에 두 개의 프로세스가 작동함으로 인해 전체적인 성능이 저하되고, 감지 후 대응이 없다는 문제가 있다. 이를 위해 프로그램 성능 최적화와 공격 탐지 시 대응을 위한 추가적인 연구를 진행하려 한다.

6. Acknowledgement

본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 SW 중심대학지원사업의 연구결과로 수행되었음(2017-0-00096). 이 성과는 2018 년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No.NRF-2017R1C1B5015045).

참고문헌

[1] Yarom Yuval, and Katrina E. Falkner. Flush+ Reload: a High Resolution, Low Noise, L3 Cache Side-Channel Attack. USENIX Security, 2014.

[2] Taylor Hornby. Side-Channel Attacks on Everyday Applications: Distinguishing Inputs with FLUSH+RELOAD. Black Hat 2016.

[3] https://docs.it4i.cz/software/debuggers/intel-performance-counter-monitor/ -Intel Performance Counter Monitor

[4] https://hunkim.github.io/ml/ -모두를 위한 머신러닝/딥러닝 강의.

[5] http://blog.lgcns.com/1359 -LG CNS 인공신경망이란 무엇인가?