KIPS Trans. Comp. and Comm. Sys.
Vol.8, No.6 pp.151~158 pISSN: 2287-5891

https://doi.org/10.3745/KTCCS.2019.8.6.151

Real-Time Detection on FLUSH+RELOAD Attack Using Performance Counter Monitor

Jonghyeon Cho† · Taehyun Kim†† · Youngjoo Shin†††

ABSTRACT

FLUSH+RELOAD attack exposes the most serious security threat among cache side channel attacks due to its high resolution and low noise. This attack is exploited by a variety of malicious programs that attempt to leak sensitive information. In order to prevent such information leakage, it is necessary to detect FLUSH+RELOAD attack in real time. In this paper, we propose a novel run-time detection technique for FLUSH+RELOAD attack by utilizing PCM (Performance Counter Monitor) of processors. For this, we conducted four kinds of experiments to observe the variation of each counter value of PCM during the execution of the attack. As a result, we found that it is possible to detect the attack by exploiting three kinds of important factors. Then, we constructed a detection algorithm based on the experimental results. Our algorithm utilizes machine learning techniques including a logistic regression and ANN(Artificial Neural Network) to learn from different execution environments. Evaluation shows that the algorithm successfully detects all kinds of attacks with relatively low false rate.

Keywords : Cache-Side Channel Attack, FLUSH+RELOAD Attack, Performance Counter Monitor, Attack Detection

Performance Counter Monitor를 이용한 FLUSH+RELOAD 공격 실시간 탐지 기법

조 종 현† · 김 태 현†† · 신 영 주†††

요 약

캐시 부채널 공격 중 하나인 FLUSH+RELOAD 공격은 높은 해상도와 적은 노이즈로 여러 악성 프로그램에서도 활용되는 등 비밀 정보의 유출에 대한 위험성이 높은 공격이다. 따라서 이 공격을 막기 위해 실시간으로 공격을 탐지하는 기술을 개발할 필요가 있다. 본 논문에서는 프로세서의 PCM (Performance Counter Monitor)를 이용한 실시간 FLUSH+RELOAD 공격 탐지 기법을 제안한다. 탐지 방법의 개발을 위해 우선 공격이 발생하는 동안 PCM의 여러 카운터들의 값들의 변화를 4가지 실험을 통해 관찰하였다. 그 결과, 3가지 중요한 요인에 의해 공격 탐지를 할 수 있다는 것을 발견하였다. 이를 바탕으로 머신 러닝의 logistic regression과 ANN(Artificial Neural Network)를 사용해 결과에 대한 각각 학습을 시킨 뒤 실시간으로 공격에 대한 탐지를 할 수 있는 알고리즘을 개발하였다. 이 탐지 알고리즘은 일정한 시간동안 공격을 진행하여 모든 공격을 감지하는데 성공하였고 상대적으로 적은 오탐률을 보여주었다.

키워드 : 캐시 부채널 공격, FLUSH+RELOAD 공격, Performance Counter Monitor, 공격 탐지

1. 서 론

최근 Meltdown이나 Spectre 등 CPU 마이크로아키텍처의 설계 결함을 이용한 새로운 종류의 공격들이 발견되었다. 이러한 마이크로아키텍처 공격들은 CPU 캐시 메모리를 매개체로 하여 기밀정보를 유출하는데, 이때 사용되는 방법이 바로 캐시 부 채널을 이용한 FLUSH+RELOAD[1, 2] 공격 기법이다. FLUSH+RELOAD 부 채널 공격은 다른 종류의 캐시 부 채널 공격들에 비해 높은 해상도와 낮은 잡음을 가지고 있어 공격 성능(Signal-to-noise)이 매우 우수하여 대부분의 CPU 마이크로아키텍처 공격에서 기본적으로 사용되고 있다. CPU 마이크로아키텍처 공격은 최근에 발견되었기 때문에 그 심각성에 비해 공격을 원천적으로 차단할 마땅한 대응방법이 많지 않은 문제가 있다. 이에 따라 공격 발생 시 이를 실시간으로 탐지하는 방법에 관한 연구가 절실히 필요하다고 할 수 있다.

※ 이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(2019-0-00533, 컴퓨터 프로세서의 구조적 보안 취약점 검증 및 공격 탐지 대응).
※ 이 논문은 2018년도 한국정보처리학회 추계학술발표대회에서 'Performance Counter Monitor을 이용한 FLUSH+RELOAD 공격 실시간 탐지 기술'의 제목으로 발표된 논문을 확장한 것임.
† 준 회 원 : 광운대학교 컴퓨터공학과 학사과정
†† 준 회 원 : 광운대학교 컴퓨터공학과 석사과정
††† 정 회 원 : 광운대학교 컴퓨터정보공학부 조교수
Manuscript Received : December 18, 2018
First Revision : February 12, 2019
Accepted : March 13, 2019
* Corresponding Author : Youngjoo Shin(yjshin@kw.ac.kr)

본 논문에서는 CPU 마이크로아키텍처 공격에서 비밀 정보를 유출하는데 핵심이 되는 FLUSH+RELOAD 부 채널 공격에 대한 실시간 탐지 기법을 제안한다. FLUSH+RELOAD 공격은 기본적으로 공격자와 공격대상 간에 공유하는 CPU 캐시 메모리를 이용하여 공격이 일어난다. CPU 캐시는 데이터 접근을 빠르게 하려고 CPU 내부와 인접한 곳에 탑재하는 작은 메모리이다. FLUSH+RELOAD 공격은 여러 계층의 캐시 메모리 중에서 LLC(Last Level Cache) 를 이용하여 공격을 진행한다. 공격이 진행되면 LLC 내 캐시 라인이 비워지게 되어 캐시 적중 실패 (Cache miss)가 비정상적으로 많이 발생하게 된다.

본 논문에서 제안하는 공격 탐지 기법의 기본 아이디어는 공격 발생 시 cache miss가 높은 비율로 증가하는 등 프로세서가 비정상적인 활동 상태를 보이는 것을 이용하는 것이다. CPU 캐시의 사용 상태에 대한 실시간 정보를 얻기 위해 인텔 프로세서에서 제공하는 PCM(Performance Counter Monitor)을 사용하였다. PCM은 캐시 사용 상태나 명령어 실행 횟수 등 프로세서 동작과 관련한 카운터를 실시간으로 제공하는 CPU 하드웨어 유닛이다.

공격 탐지 기법을 설계하기 위하여 우선 cache miss 뿐만 아니라 PCM 이 제공하는 다양한 종류의 카운터 값 중에 탐지에 사용될 수 있는 카운터를 식별하였다. 그다음 머신러닝의 logistic regression과 ANN (Artificial Neural Network) 기법을 이용하여 앞서 식별된 PCM 카운터의 값에 따라 정상적인 상태인지 공격이 진행 중인 상태인지를 판별하는 알고리즘을 개발하였다. 다양한 컴퓨터 사용 환경을 반영한 탐지 모델을 얻기 위해 동영상 재생, 오피스 및 웹 브라우저 사용 등 여러 애플리케이션을 동시에 실행하는 실험 환경들을 구성하여 학습을 진행하였다. 이를 통해 얻은 모델을 바탕으로 본 논문에서 제안하는 탐지 기법은 실제 FLUSH+RELOAD 공격이 일어날 때 99% 이상의 정확도로 탐지에 성공하였다.

본 논문의 구성은 다음과 같다. 2장에서는 FLUSH+RELOAD 캐시 부 채널 공격, PCM 및 logistic regression 등에 대한 배경지식을 설명한다. 3장에서는 탐지에 필요한 PCM 카운터를 식별하기 위한 실험 설계와 실험 수행 결과를 기술한다. 4장에서는 3장의 결과를 이용한 공격 탐지 기법을 설계한 내용을 기술한다. 5장에서는 FLUSH+FLUSH 와 PRIME+PROBE 등 다른 종류의 캐시 부 채널 공격에 대한 탐지 효용성에 대해 논의하며 마지막 6장에서는 결론 및 향후 연구계획을 기술한다.

2. 배경 지식

2.1 FLUSH+RELOAD 공격

FLUSH+RELOAD 공격은 공격자(spy)와 공격 대상(victim)이 같은 CPU 캐시를 공유하여 동작하는 환경에서 상호 간의 캐시 간섭을 부 채널로 활용하여 비밀 정보를 획득하는 공격이다. 이 공격은 spy와 victim 간에 메모리 페이지를 공유하고 있는 조건에서 공유 캐시 라인을 clflush명령어(L3 캐시를 포함한 모든 레벨의 캐시에 들어있는 특정 메모리 라인을 제거하는 명령어)를 이용하여 간섭을 유발한다.

FLUSH+RELOAD 공격은 크게 세 단계로 나누어지게 된다. 공격의 첫 단계는 spy가 모니터링 하고자 하는 대상 메모리 라인을 clflush 명령어를 이용하여 모든 레벨의 캐시에서 제거한다. 두 번째 단계에서 spy는 victim이 가지고 있는 비밀 정보(e.g., 비밀 키 등)를 바탕으로 프로그램(e.g., 암호 알고리즘 등)을 실행하는 동안 대기한다. 이때 victim 의 비밀 정보 값에 따라 대상 메모리 라인의 엑세스 여부가 달라진다. 세 번째 단계에서는 spy는 대상 메모리 라인을 다시 엑세스(reload)하고 엑세스에 걸린 시간을 측정한다. 만약 기다리는 시간 동안 victim이 대상 메모리 라인에 접근했다면 데이터는 캐시에 적재되고 따라서 엑세스 시간은 짧아진다. 반대로 victim이 메모리 라인에 접근을 안 했다면 데이터는 메인 메모리로부터 가져오므로 오랜 시간이 걸리게 된다. 이 시차를 이용하여 해당 비밀 정보의 비트 값을 알아낼 수 있다. 이 공격을 이용하여 암호 알고리즘의 개인키 값과 응용 프로그램에서의 활동 내용 등 다양한 정보를 추출할 수 있다.

FLUSH+RELOAD 공격은 주로 RSA[1]나 AES[2, 12] 등 암호 알고리즘을 대상으로 하여 비밀 키를 추출하는 데 활용되었다. 또한, 이 공격은 victim이 키보드 이용 시 키 입력 값[3]에 대한 정보도 추출할 수 있음을 보여줌으로써 그 응용 범위가 매우 광범위하다.

2.2 FLUSH+FLUSH 공격

FLUSH+FLUSH [4] 부 채널 공격은 앞서 설명한 FLUSH+RELOAD 공격의 변종 형태이다. 이 공격은 세 번째 단계에서 메모리 엑세스(reload)를 수행하는 대신에 clflush 명령을 실행하여 캐시 라인을 flush 하는 방식으로 공격을 진행한다. Flush 동작을 다시 할 때 clflush명령어 시간을 측정해서 victim의 메모리 엑세스 여부를 판단한다. 만약 데이터가 대상 캐시 라인에 적재된 상태라면(즉, victim 이 메모리에 엑세스 하였다면) clflush 명령어의 실행 시간은 길게 측정이 된다. 반대로 메모리 엑세스가 일어나지 않았다면 캐시 라인에 데이터가 존재하지 않기 때문에 명령어의 실행시간이 짧아지게 된다. 이러한 clflush명령어의 실행시간의 차이를 이용해서 victim 의 메모리 엑세스 여부를 알 수 있게 된다.

본 공격은 FLUSH+RELOAD 공격과 달리 공격 과정에서 메모리 엑세스(reload)가 일어나지 않기 때문에 프로세서의 캐시 사용 빈도가 정상적인 경우와 크게 차이가 나지 않는다. 따라서 PCM을 이용한 공격 탐지 시 쉽게 탐지가 되지 않는 장점이 존재한다.

2.3 PRIME+PROBE 공격

PRIME+PROBE 공격[5]은 앞서 기술한 두 개의 캐시 부 채널 공격과 다른 형태로 진행된다. Spy와 victim은 같은 캐시 집합을 공유하는 상태에서 공격을 진행한다. 캐시 집합을

공유하기 위해서는 우선 메모리 주소가 캐시에 어떻게 매핑되는지를 알아야 한다. 이를 위해 인텔 프로세서의 슬라이스 구조를 역공학 분석하거나[6] 전체 캐시 라인 중 대상 캐시 집합과 충돌을 일으키는 라인들의 충돌 집합을 구성하는 과정이 요구된다[7].

Victim이 사용하는 대상 캐시 집합을 식별하였다면 공격이 시작된다. PRIME+PROBE 공격의 첫 번째 단계는 대상 캐시 집합에 spy의 데이터를 먼저 채워 넣는 것이다. 그다음, 두 번째 단계에서 victim이 프로그램을 실행하는 동안 대기한다. Victim은 가지고 있는 비밀 정보의 값에 따라 프로그램 실행 과정에서 대상 캐시 집합의 엑세스 여부가 결정된다. 이후에 세 번째 단계에서 spy 는 다시 자신의 데이터를 엑세스하여 그 시간을 측정한다. 만약 victim이 메모리를 엑세스하였다면 대상 캐시 집합이 victim의 데이터들로 채워지면서 동시에 spy의 데이터는 evict 되므로 victim의 메모리 엑세스 여부를 알 수 있게 된다.

이 공격은 캐시 라인을 이용하지 않기 때문에 좀 더 광범위 하게 사용이 되며 클라우드 환경에서 은닉 채널로 많이 사용된다[4]. 클라우드 환경에서는 서로 다른 매체가 캐시라인을 공유하는 것이 힘들기 때문에 캐시 집합을 공유한 상태에서 은닉채널로써 데이터를 전달하는데 본 공격이 활용될 수 있다.

2.4 Performance Counter Monitor (PCM)

Performance Counter Monitor(PCM) [8]는 인텔 프로세서의 퍼포먼스 카운터 값을 보여주기 위한 도구로써 PAPI (Performance application programming interface) 와 유사하다. PAPI는 프로세서 아키텍처에서 제공하는 성능 카운터에 접근하기 편한 인터페이스이며 CPU 카운터에 제한되지 않고 CUDA나 네트워크와 같은 다른 요소에서도 동작하는 특징이 있다. PCM과 PAPI의 차이점은 PCM은 오직 프로세서 코어에 대한 카운터 값만 지원을 하지만 PAPI는 QPI(Quick Path Interconnect)와 같은 인텔 프로세서의 언코어(Uncore) 부분의 카운터도 접근할 수 있다. 본 논문에서 제안하는 탐지 기법은 프로세서 코어에서 일어나는 동작들을 모니터링하는데 국한되므로 PCM에서 제공하는 카운터로도 충분하다.

2.5 Logistic Regression

단일 데이터(x)에 대한 Logistic Regression[9]은 분류하고자 하는 데이터에 대한 가중치(W)를 곱한 뒤 바이어스(b) 더한 값을 Equation (1)의 linear H(x)이라 하면, logistic regression의 특성인 binary classification을 위해 linear H(x)는 모든 값을 0~1 사이로 만들어져야 한다. Equation (1)에서 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 0~1이다. 만약 이 가정 값이 0.5보다 크면 예측 값은 1 이며, 0.5보다 작으면 예측 값이 0이다. Equation (1)에서 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 cost의 값은 작아지고, 예측 값이 틀리면 cost 값이 커지는 것을 의미한다.

$$\begin{aligned} \text{linear } H(x) &= Wx + b \\ g(z) &= \frac{1}{e^{-z}} \\ H(x) &= \frac{1}{1+e^{-(Wx+b)}} \\ C(H(x),y) &= \begin{cases} -\log(H(x)) & (y=1) \\ -\log(1-H(x)) & (y=0). \end{cases} \end{aligned} \tag{1}$$

2.6 ANN(Artificial Neural Network)

인공 신경망(ANN)[10]은 생물학의 신경망에서 영감을 얻은 학습 알고리즘이다. 일반적으로 사용되는 인공신경망은 입력층과 은닉층 그리고 출력층 이렇게 세 가지 층으로 구분된다. 그리고 각 층은 여러 개의 노드들로 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 2개 사용하여 훈련을 진행하도록 했다.

3. 공격 탐지를 위한 PCM 카운터 식별

PCM은 캐시 적중률이나 명령어 실행 횟수 등 프로세서 동작 상태를 모니터링하기 위한 여러 성능 카운터 값을 제공한다. PCM이 제공하는 다양한 카운터들 중에서 FLUSH+RELOAD 캐시 부 채널 공격 탐지에 사용될 카운터를 찾기 위하여 실험을 진행하였다. 본 장에서는 실험을 통해 공격 탐지를 위한 PCM 카운터를 식별하는 내용을 기술한다.

3.1 실험 환경

PCM의 어떤 카운터가 FLUSH+RELOAD 공격을 받을 때 영향을 받는지 몇 가지 실험을 위해 컴퓨터를 spy와 victim으로 나누어 실험을 진행했다. Spy 컴퓨터는 Intel® Core™ i5-5250U 1.6GHz 프로세서를 탑재하고 8GB DDR3 메모리를 사용하는 MacBook Air(2015년도 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 LTS로 같은 Unix계열 운영체제를 사용하고 있다.

공격 실험을 위해 Mastik [11]의 FLUSH+RELOAD 공격

프로그램을 사용하였다. Mastik은 여러 가지 캐시 부채널 공격들을 한곳에 모아 놓은 도구로 이 프로그램을 사용한다면 간단한 환경설정만으로 쉽게 FLUSH+RELOAD 공격을 실행할 수 있다.

3.2 실험 결과 및 관찰

FLUSH+RELOAD 공격은 같은 컴퓨터 위에서 공격이 이루어져야 하므로 spy의 컴퓨터에서 SSH 접속을 통해 victim 컴퓨터의 계정으로 접속하여 공격을 진행하였다.

어떤 PCM 카운터가 공격 탐지에 효과적인지 파악하기 위해서는 실험 환경을 최대한 실제 컴퓨터의 다양한 사용 환경과 유사하게 맞추어서 실험을 진행할 필요가 있다 따라서 victim 컴퓨터에 비디오 재생, 음악 재생, LibreOffice Calc 와 Firefox 등 총 네 가지 응용프로그램을 각각 실행시켜 실험을 진행했다. 각 응용프로그램 마다 관찰되는 카운터들의 양상이 모두 달랐기 때문에 3분의 시간을 두고 총 3번의 공격을 하여 공통으로 어떠한 변화가 있는지 관찰했다.

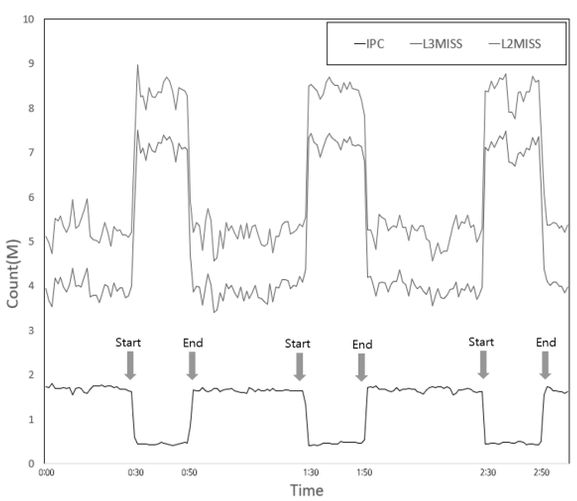

Fig. 1. IPC, L3 Miss, L2 Miss Count Values when Playing Video
(Start: The Beginning of Attack, End: The End of Attack)

첫 번째 실험에서는 비디오 재생 중 FLUSH+RELOAD 공격을 진행하여 카운터 값들을 관찰하였다. 시작을 기준으로 3분간 PCM을 이용하여 측정하였는데, 분마다 30초부터 50초 동안 공격을 했을 때 큰 변화를 보인 카운터 값은 L3 Miss, L2 Miss, IPC로 3가지를 발견할 수 있었다. L3 Miss 와 L2 Miss 는 각각 프로그램 실행 중 L3 캐시와 L2 캐시에서 발생한 적중 실패 횟수를 나타낸다. IPC 는 단위 사이클 당 처리된 명령어의 평균 횟수를 나타낸다. Fig. 1을 보면 공격을 하지 않을 때 L3 Miss와 L2 Miss는 각각 4M와 5~6M의 값을 가지며 IPC는 2M가 조금 안 되는 것을 볼 수 있다. 한편, 공격이 진행되는 30초부터 50초에서는 L3 Miss의 카운터는 7M, L2 Miss는 8.5M, IPC는 0.5M로 변하여 카운터 값이 확연히 달라지는 것을 확인할 수 있다.

두 번째 실험에서는 음악 재생 중 FLUSH+RELOAD 공격을 진행하였다. 첫 번째 실험과 같은 조건에서 실험을 진행했을 때 역시 세 가지 카운터가 크게 변하는 것을 볼 수 있었다. Fig. 2를 보면 공격을 하지 않을 때 L3 Miss의 카운터는 0.1M, L2 Miss의 카운터는 0.2M, IPC의 카운터는 1M 정도 나타나는 것을 볼 수 있다. 한편 공격이 진행되는 구간에는 첫 번째 실험과 같이 L3 Miss는 3M, L2 Miss는 3.5M, IPC는 0.2M로 두 카운터는 증가하고 IPC는 감소하는 것을 볼 수 있었다.

세 번째와 네 번째 실험은 각각 LibreOffice와 Firefox를 사용 중인 환경에서 진행하였으며, 모두 L3 Miss와 L2 Miss의 크기만 다를 뿐 L3, L2의 급격한 증가와 IPC 감소의 공통적인 결과를 나타냈다. 네 가지 실험에서 공통으로 나타난 것은 프로그램을 실행할 때 해당 프로그램에 대한 정보를 가져오는 과정에서 캐시 miss가 급증하는 모습을 볼 수 있었다.

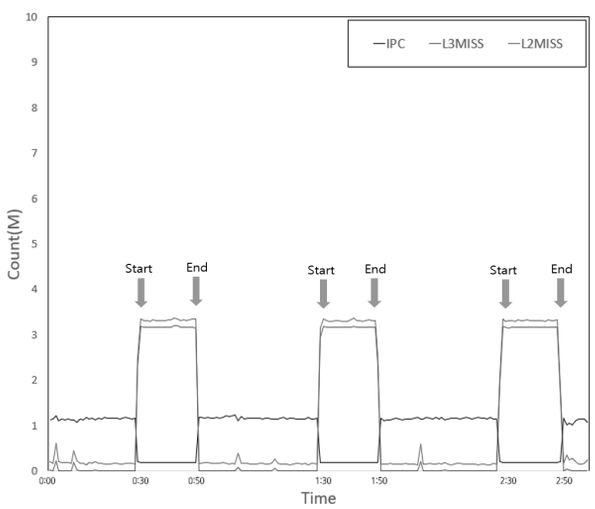

Fig. 2. IPC, L3 Miss, L2 Miss Count Values when Playing Music
(Start: The Beginning of Attack, End: The End of Attack)

이 네 가지 실험을 통해 확인할 수 있는 것은 FLUSH+RELOAD 공격을 진행할 경우 여러 가지 요인이 변하는 것을 PCM을 통해 관찰할 수 있으며 그중 가장 큰 변화는 L3 Miss, L2 Miss, IPC라는 것을 볼 수 있었다. 이러한 결과가 나타나는 이유를 설명하면, FLUSH+RELOAD 공격에서 clflush명령어를 사용하여 L3 캐시에 있는 데이터를 제거하는 환경에서, victim이 해당 캐시라인을 접근하는 경우(공격성공), 같은 환경에서 victim이 해당 캐시라인을 접근하지 않고 spy가 reload하는 경우(공격실패) 모두 L3 캐시 Miss가 발생한다. 따라서 FLUSH+RELOAD 공격이 진행될 때 victim의 접근 유무와 관계없이 L3 캐시 Miss가 발생하며, 이 때문에 하나의 명령어를 사용할 때 걸리는 사이클이 매우 증가하여 IPC가 감소한다고 판단할 수 있었다. 이 실험을 통해 다음 장에 진행할 논리 회귀 머신러닝을 통한 공격예측 프로그램 제작을 진행하였다.

4. PCM을 이용한 실시간 공격 탐지 기법

앞서 얻은 실험의 정보를 통해 PCM이 제공하는 세 가지 카운터의 변화를 관찰하여 FLUSH+RELOAD 공격을 탐지할 수 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 등 PCM 이 제공하는 다른 카운터를 추가로 입력값으로 사용했다. 앞서 4가지 실험에서 사용된 응용프로그램들을 실행할 때 카운터들이 각각 다르므로 일정 시간을 두고 동시에 여러 작업을 진행하며 PCM을 통해 카운터 값들을 CSV파일로 추출하였다. 총 10분의 시간 동안 4가지 프로그램들을 실행 및 종료하며 3번의 공격을 진행하였다. 이때 공격을 진행한 시간을 모두 표시하여 학습 입력값의 label로 사용할 수 있도록 했다. Label은 FLUSH+RELOAD 공격이 진행된 경우를 1, 아무 공격이 가해지지 않은 경우를 0으로 하여 트레이닝 집합을 제작했다.

머신 러닝은 오픈소스인 Tensorflow를 사용하여 진행했는데, 트레이닝 집합을 이용하여 logistic regression, ANN두 가지 기법을 활용하여 학습모델을 제작했다. CSV파일을 사용하여 훈련을 진행한 결과, 초기 학습모델의 정확도는 logistic regression은 96.07%, ANN은 99.05%로 ANN이 좀 더 높은 정확도를 나타내는 것을 확인할 수 있었다.

실시간으로 감지하기 위해서는 항상 훈련할 수 없으므로 학습된 모델은 메타파일로 저장하여 입력 값만 넣으면 바로 결과가 나올 수 있도록 했다. Fig. 3은 파이썬을 기반으로 하여 작성한 해당 공격 탐지 프로그램의 의사코드이다. 프로그램의 기본언어로 파이썬으로 선택한 이유는 tensorflow를 사용하여 학습한 모델을 이용하기 때문에 이 모델을 이용하기 위해서는 파이썬 언어를 사용할 필요가 있었다. 또한, 파이썬을 이용한다면 더 짧은 코드로 프로그램의 작성이 가능하다.

이 프로그램은 PCM을 사용하여 나오는 결과를 학습된 모델의 입력 값으로 넣어 탐지하므로 두 개의 프로세스를 이용해야 한다. 따라서 첫 번째 줄(line 1)에서 fork() 함수를 사용하여 프로세스를 생성하고 하나의 프로세스는 PCM을, 하나의 프로세스는 탐지를 진행하도록 했다. fork() 함수가 실행된 이후에는 부모와 자식 두 개의 프로세스로 나뉘게 된다. PCM을 실행시키는 자식 프로세스는 CSV파일로 카운터 값을 출력하게 되고(line 3), 탐지를 진행하는 부모 프로세스에서는 다른 프로세스에서 내보낸 CSV파일을 읽어 탐지를 진행한다(lines 5-14). PCM이 CSV파일을 생성할 때 한 줄씩 누적하여 출력하기 때문에 skiprow를 활용하여 최신의 정보를 받아올 수 있도록 했다(line 6). 그 다음 line 7에서 파일에서 읽은 카운터 값들을 array로 만들어 detection함수의 입력값으로 넣어 학습된 모델에 대입하고, 해당 카운터들의 값들이 공격을 받을 때의 값들이라면 1을, 아니라면 0을 결과 값으로 출력하여 사용자에게 공격의 감지를 알린다(line 8). CSV 파일의 데이터가 많아지면 해당 줄을 찾는데 시간이 오래 걸리기 때문에 20개의 정보가 생성될 때마다 로그 파일을 초기화해주어 줄을 찾는데 시간을 줄일 수 있도록 했다(lines 12-14).

```
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)
```

Fig. 3. Pseudo Code for Real-time Detection Program

4.1 성능 평가

우리가 개발한 방법의 성능을 평가하기 위하여 실제 공격 발생 시 얼마나 정확하게 탐지를 하는지 실험을 통해 측정하였다. PCM을 이용하기 위해서는 root 권한이 필요하므로 root 권한으로 학습된 두 모델로 실시간 탐지를 진행하였다. FLUSH+RELOAD 공격은 spy가 LLC 캐시 라인을 flush하고 reload하는 과정 중 victim 프로그램의 실행이 있어야 공격에 성공하므로[1], spy는 victim 프로그램의 실행을 기다리며 flush와 reload를 계속 진행하게 된다. 따라서 탐지프로그램은 이러한 반복에서 변화되는 L3 Miss, L2 Miss 등의 카운터 값을 탐지하기 위해 CSV의 값을 읽어 판단하도록 했다. 탐지 시간 간격은 탐지 시간을 너무 빠르게 하면 전체적인 시스템 성능 저하가 발생하여 카운터 값들에도 영향을 미칠 수 있다고 판단하여 0.2초로 설정했다. PCM의 옵션으로 0.2초마다 CSV 파일에 기록하도록 하고 프로그램 역시 0.2초의 시간 지연을 주어 기록과 탐지의 시간을 맞춰 주었다. 각각 5분의 시간을 두고 victim 컴퓨터로 탐지 프로그램을 실행하고 spy 컴퓨터로 3번의 공격을 진행하여 올바른 탐지 여부를 실험해 보았다.

실험 결과, logistic regression을 이용한 탐지는 3번의 공격을 성공적으로 탐지하였고, 공격 이외의 탐지를 1초를 기준으로 총 8초 정도 감지하였다. ANN을 이용한 탐지는 공격 감지에는 성공하였으나, 앞선 결과에 비해 탐지결과가 너무 늦게 나오는 것을 확인할 수 있었다. 이에 대한 이유를 추측해 보았는데, logistic regression보다 ANN의 연산 횟수가 매우 많으므로 이러한 결과가 나오는 것으로 볼 수 있었다. 따

라서 정확도는 ANN을 이용한 모델이 좀 더 높지만, 실시간 탐지에는 부적합하다는 것을 발견할 수 있었다.

공격 탐지 간격에 대한 논의. FLUSH+RELOAD 공격을 활용하여 암호키를 탈취하는 공격에 대한 기존의 연구 결과를 보면 AES의 경우 128 비트 비밀키를 완전히 추출하는데 약 25초(Single OS 환경)에서 1분(Cross-VM 환경)정도 시간이 소요되며[12] 이보다 개선된 후속 연구 결과에서도 10초 내외의 시간이 소요되는 것으로 알려져 있다[2]. RSA 알고리즘의 경우 AES 보다 더 긴 길이의 개인키(2048 비트)를 추출해야 하므로 공격에 소요되는 시간은 훨씬 더 길다[1]. 한편 FLUSH+RELOAD를 이용하여 victim의 키보드 입력을 가로채는 공격의 경우 원활한 공격을 위해 victim 이 최소 0.8 초마다 키 입력을 하는 상황을 가정하고 있다[3]. 즉, 공격자는 최소 0.8초 이상 동안 FLUSH+RELOAD 공격을 지속해야만 한다. 따라서 본 논문에서 제시한 탐지 간격인 0.2초는 FLUSH+RELOAD 공격을 탐지하여 암호키나 키보드 입력 값과 같은 비밀정보의 유출을 실시간으로 차단하는 데 충분하다고 볼 수 있다.

4.2 다양한 컴퓨팅 환경에서의 공격 탐지

앞서 기술한 공격 탐지 기법이 보다 다양한 컴퓨팅 환경에서도 적용될 수 있는지를 확인하기 위하여 두 가지 조건을 달리하여 실험을 진행했다. 첫 번째는 동일한 OS를 갖는 서로 다른 컴퓨터 환경에서의 실험을 진행하였고 두 번째는 모두 다른 OS를 갖는 컴퓨터 환경에서의 실험을 진행했다. 실험에 사용한 victim 컴퓨터는 Intel® CoreTM i5-7400 3.0GHz 프로세서를 탑재하고 16GB DDR RAM3를 사용하는 Lenovo ideacentre 720를 이용하여 실험을 진행하였다. 두 조건에서 사용한 공격 프로그램은 동일하며 탐지 프로그램은 기존에 학습된 모델을 그대로 이용했다.

첫 번째 실험은 앞선 OS환경과 동일한 Ubuntu Linux를 사용하여 진행했다. 가상환경은 VMware를 사용하였고, 프로세서는 2개로 설정하고 RAM은 1GB의 환경을 설정했다. 앞선 조건과 동일하게 탐지 프로그램을 실행한 상태에서 여러 가지 프로그램들을 실행시키고 공격을 진행하였는데 동일한 OS에서는 공격을 성공적으로 감지하는 것을 확인할 수 있었다.

두 번째 실험은 가상환경에서의 여러 가지 다른 OS에서의 실험을 진행했다. 가장 먼저 CentOS 7 Linux에서 동일하게 진행했는데, CentOS에서는 PCM이 작동되지 않는 문제점을 발견하여 프로그램 실행을 진행하지 못하였다. 다음으로 진행한 운영체제는 Debian 10 Linux buster를 이용하여 진행했다. Debian OS는 Ubuntu와 비슷한 환경이었기 때문에 문제없이 PCM이 잘 작동하는 것을 확인하였고, 동일한 실험을 진행했다. 역시 여러 가지 응용프로그램들을 실행시킨 상태에서 공격을 진행한 결과, 탐지가 잘 이루어지는 것을 볼 수 있었다.

같은 컴퓨터에서의 같은 OS, 다른 OS에서의 실험은 CentOS를 제외하고는 모두 탐지프로그램이 잘 작동하는 것을 볼 수 있었다. 하지만, 두 가지 환경에서 모두 탐지를 하는 과정에서 몇 번의 잘못된 탐지를 발견했는데, 이에 대해 연구를 진행한 바로는 가상환경 아래의 OS와 함께 가상 CPU를 사용하며 실행되기 때문에 가상환경에서의 카운터의 값이 기존 환경보다 더 많은 값을 가지고 있는 것으로 이유를 찾을 수 있었다. 따라서 이에 대한 적응으로 가상환경에 대한 학습이 진행된다면, 좀 더 좋은 결과가 나올 것으로 예측해 보았다.

5. 다른 종류의 캐시 부채널 공격에 대한 탐지 성능 고찰

본 논문에서 제안한 방법은 여러 캐시 부채널 공격 중 가장 많이 사용되는 FLUSH+RELOAD 공격에 대한 탐지에 초점을 맞추어 설계되었으며, 실험을 통해 높은 공격 탐지성능을 제공함을 확인하였다. 그러나 본 제안 방법은 FLUSH+RELOAD 공격뿐만 아니라 다른 캐시 부 채널 공격들에 대해서도 탐지 효용성을 가지고 있다. 이 장에서는 다른 종류의 공격들에 대한 탐지 성능에 대해 논의한다.

5.1 FLUSH+FLUSH 공격의 탐지

본 논문에서 제안한 탐지 프로그램이 보다 은밀하게 공격을 수행하는 경우에도 대응할 수 있는지 확인하기 위하여 추가 실험을 진행하였다. 구체적으로 FLUSH+FLUSH 공격을 수행하였을 때 IPC, L3 Miss와 L2 Miss의 카운터 값의 변화 양상을 확인해 보았다. 이 실험에서도 마찬가지로 Mastik의 FLUSH+FLUSH 프로그램을 사용하여 어떻게 PCM의 카운터 값들이 변화하는지 지켜보았다. 앞서 실험들과 동일한 조건으로 3분의 시간을 두어 30초 동안 3번의 공격을 진행했다. 카운터 값을 측정하는 동안 Firefox와 Youtube를 실행하여 값의 변화를 확실히 볼 수 있도록 했다.

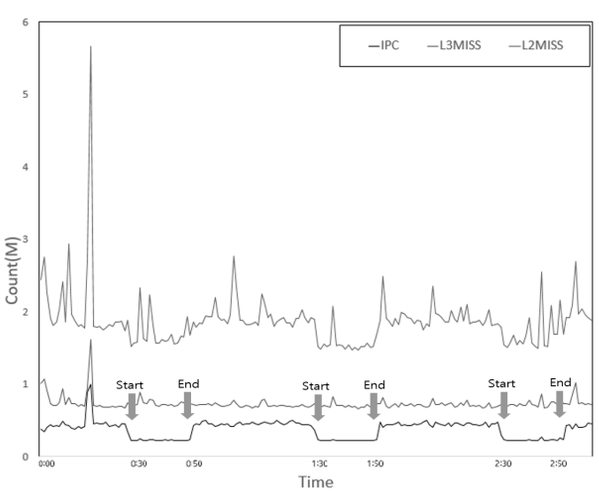

Fig. 4. IPC, L3 Miss, L2 Miss Count Values when Playing Youtube (Start: The Beginning of Attack, End: The End of Attack)

위 Fig. 4는 FLUSH+FLUSH 공격에 대한 탐지 실험 결과

를 보여준다. 그림의 값들을 보면 L3 Miss와 L2 Miss의 값들은 공격을 진행할 경우 별 차이가 없는 것을 볼 수 있었다. 이는 FLUSH+FLUSH 공격의 경우 메모리를 엑세스하는 reload 단계가 생략되었기 때문으로 본 공격을 제안한 저자들이 주장하는 바와 같이 높은 은닉성을 갖는 것을 확인하였다. 하지만, IPC의 값의 변화 양상을 보면 FLUSH+ RELOAD 공격에 비해 많은 변화량을 보여주지는 않았으나 공격이 실행되는 동안 어느 정도 감소하는 것을 볼 수 있었다. 따라서 IPC 카운터를 이용하여 여전히 FLUSH+FLUSH 공격에 대한 탐지가 가능함을 알 수 있다.

5.2 PRIME+PROBE 공격의 탐지

PRIME+PROBE 공격은 spy와 victim 간에 메모리 공유가 일어나지 않은 조건에서 수행할 수 있는 공격 기법이다. 따라서 캐시 집합 내의 모든 캐시 라인들에 대해서 victim 의 캐시 사용 형태를 파악해야 하므로 필연적으로 FLUSH+RELOAD 공격에 비해 훨씬 많은 메모리 엑세스가 발생하게 된다.

본 논문의 제안 방법은 PCM 카운터 중 L2 Miss 와 L3 Miss의 값을 기준으로 공격 여부를 탐지하므로 메모리 엑세스가 많이 일어나는 PRIME+PROBE 공격에 대해서도 쉽게 탐지가 가능하리라고 예상할 수 있다.

6. 결 론

본 논문에서는 PCM을 사용하여 캐시 부 채널 공격 기법 중에 하나인 FLUSH+RELOAD 공격에 대한 실시간 탐지 기법을 제안하였다. 이를 위해 여러 가지 실험을 통해 중요한 요인을 알아내고 해당 값들을 머신 러닝을 통해 변화에 대한 label을 학습시켜 실시간으로 공격에 대한 탐지를 할 수 있도록 했다. 또한 여러 가지 환경에서의 탐지 프로그램 실행을 통해 연구를 통해 만들어낸 프로그램으로 성공적으로 FLUSH+FLUSH나 PRIME+PROBE 등 모든 공격에 대한 탐지를 할 수 있는 결과를 보여주었다.

이 결과를 토대로 프로그램 성능 최적화와 공격 탐지율 극대화 방안, 공격 대응 등 추가적인 연구를 진행하려 한다.

References

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

[2] B. Gulmezoglu, M. Inci, G. Irazoki, T. Eisenbarth, and B. Sunar, “Cross-VM Cache Attacks on AES,” *IEEE Trans. Multi-Scale Comput. Syst.*, Vol.2, No.3, pp.211-222, 2016.

[3] Daniel Gruss, Raphael Spreitzer, and Stefan Mangard, “Cache Template Attacks : Automating Attacks on Inclusive Last-Level Cache,” *USENIX Security Symposium 2015.*

[4] Daniel Gruss, Raphael Spreitzer, and Stefan Mangard, “Flush+Flush : A Fast and Stealthy Cache Attack,” *DIMVA 2016 Detection of Instruction and Malware, and Vulnerability Assessment.*

[5] Manuel Weber, Michael Schwarz, Lukas Giner, and Daniel Gruss, “Hello from the Other Side : SSH over Robust Cache Covert Channels in the Cloud.” *Network and Distributed System Security Symposium 2017 (NDSS'17).*

[6] Gorka Irazoqui, Thomas Eisenbarth, and Berk Sunar, “Systematic Reverse Engineering of Cache Slice Selection in Intel Processors,” *2015 Euromicro Conference on Digital System Design (DSD).*

[7] Fangfei Liu, Yuval Yarom, Qian Ge, Gernot Heiser, and Ruby B Lee, “Last-Level Cache Side-Channel Attacks are Practical,” *2015 IEEE Symposium on Security and Privacy.*

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

[9] Machine Learning and Deep Learning for Everyone [Online], https://hunkim.github. io/ml/.

[10] What is Artificial Neural Network? [Online], http://blog.lgcns.com/1359.

[11] Y. Yarom, “Mastik: A Micro-Architectural Side-Channel Toolkit,” [Online] https://cs.adelaide.edu.au/~yval/Mastik/.

[12] Irazoqui, Gorka, et al. “Wait a minute! A fast, Cross-VM attack on AES,” *Research in Attacks, Intrusions and Defenses. Springer International Publishing,* pp.299-319, 2014.

조 종 현

https://orcid.org/0000-0001-8688-2160

e-mail :whwhdgus94@naver.com

2013년～현 재 광운대학교 컴퓨터공학과 학사과정

관심분야 : Cache Side-Channel Attack & Machine Learning

김 태 현

http://orcid.org/0000-0001-7590-4168

e-mail : taehyun9203@gmail.com

2018년 광운대학교 전자공학과(학사)

2018년～현 재 광운대학교 컴퓨터공학과 석사과정

관심분야 : CPU Micro-Architectural Security

신 영 주

http://orcid.org/0000-0003-4831-7392
e-mail : yjshin@kw.ac.kr
2006년 고려대학교 컴퓨터학과(학사)
2008년 KAIST 전산학과(석사)
2014년 KAIST 전산학과(박사)
2008년~2017년 국가보안기술연구소 선임연구원
2017년~현 재 광운대학교 컴퓨터정보공학부 조교수
관심분야 : Applied Cryptography, CPU Micro-Architectural Security, Cloud Computing, SDN/NFV Security