Scalar lab 발표 시작하겠습니다.

저희 연구실은 uncore 부분에서의 side attack에 대해 연구를 진행하게 되었습니다. 그 중 timing attack의 본질을 이해하기 위해 cache side attack 기법 중 하나인 flush reload에 대해 공부하고 재현한 과정에 대해 발표 드리겠습니다.

시간차 분석 부채널 공격(timing analysis side channel attack) 중 2013년에 제안된 암호화 알고리즘에 대한 FLUSH+RELOAD 공격을 재현해보고 기존 인텔 하드웨어의 보안 취약점을 검증한다. 특히 가상화된 환경에 확장하여 OpenSSL의 AES-128를 타깃으로 Rijndael 암호 알고리즘의 128-비트 대칭키를 복구해보고 공격 결과를 분석한다.

사실 flush reload는 다른 연구실에서도 다뤄본 주제로 알고 있고, 실제로 광운대 교수님께서는 flush reload에 관련한 국내 논문도 내셨다고 알고 있어서 저희보다 더 잘 아시겠지만,, 마음으로 진행하겠습니다.

목차는 flush reload attack에 대한 설명, threshold 설정 과정, flush reload를 이용한 open s니 공격 구현, 마지막으로 future work으로 진행할 계획입니다.

Flush reload란 intel cpu의 구조적 특징에 의해 발생할 수 있는 side attack 공격 방법입니다. intel cpu의 구조적인 특징인 shared cache, inclusive cache라는 점과 cache에서 데이터를 가져올 때의 시간과 memory에서 data를 가져올 때 걸리는 시간의 차이점을 이용하여 공격을 진행합니다.

진행과정은 다음과 같습니다.

먼저 공격자가 공유 주소를 cache로부터 flush 시킨 후, 피해자의 행동을 주시하며 기다립니다. 그리고 적당한 시간이 지난 후 flush 시킨 memory line을 reload 할 때의 시간을 측정합니다. 만약 reload 했을때의 시간이 특정 임계값보다 빠르다면 피해자가 해당 memory line을 접근했다는 의미이고, 임계값보다 느리다면 피해자가 memory line을 접근하지 않았음을 의미하게 됩니다. 이 방식으로 공격자는 피해자가 어떠한 주소를 접근했는지 유추할 수 있으며, 이를 기반으로 피해자의 정보를 추출할 수 있게 됩니다.

기존의 공격방식과 flush reload의 차이점을 들자면 크게 두 가지가 있습니다. 첫번째는 기존 공격이 cache set 단위의 공격을 진행하였다면, flush reload는 특정 memory line 단위로 접근을 하기 때문에 더 높은 정확도를 얻을 수 있게 됩니다. 두번째는 L1 cache에 집중된 반면, LLC에 집중을 하여 다른 코어 환경에서의 공격이 가능하다는 점입니다.

저희는 두 가지 환경에서 flush reload 공격을 재현해보았습니다. Cpu는 다음과 같고, OS는 우분투 16.04로 동일하며, 가상머신 상에서 공격이 가능한지를 보기 위해 AEWS 서비스로 제온의 경우 cpu 환경을 실행해 보았습니다.

먼저 시스템환경에 따라 각각의 임계값을 구하기 위해 임계값을 구하는 코드를 따로 돌려보았습니다. 진행과정은 다음과 같습니다. 먼저 cache에 값을 넣어 채운 후, load 명령어를 사용하여 cache hit 일때의 load time을 추출하였습니다. Load time들은 hit histogram이라는 배열에 저장을 합니다.

그 다음 cache를 모두 flush 시킨 후 cache 다시 load 명령어를 실행하여 cache miss일 때의 시간을 추출하였습니다. 마찬가지로 load time들은 miss histogram이라는 배열에 저장하였습니다.

Hit histogram에 저장되어있는 load time 중 가장 높은 빈도수의 time과, miss histogram 중 가장 짧게 걸린 load time을 각각 구합니다. 즉, cache로부터 가져올 때 가장 높은 확률의 시간과 memory로부터 가져올 때 걸렸던 시간 중 가장 짧았던 시간을 두 spot으로 잡습니다.

그리고 두 지점의 사이 영역 중 hit과 miss가 가장 적게 일어난 영역을 trheshold로 잡습니다. 간단히 설명드리면 cache hit일때의 시간과 chache miss일 때의 시간 사이 부분에서 영역을 잡는데, hit와 miss가 둘 다 안일어나서 가장 양분화가 잘 일어날 시간을 임계값으로 정하는 것입니다.

사진 예시를 보면 왼쪽이 hit histogram, 오른쪽이 miss histogram을 의미하는데, cahce hit일때와 cache miss일때의 값 사이에서 가장 최소 시간을 threshold로 지정하였습니다. 굳이 최소 시간으로 지정한 이유에 대해서는 threshold가 낮을수록, 컴파일러의 추측실행에 의한 1종 오류, false positive를 줄일 수 있기 때문입니다.

이제 open ssl에서의 대칭키를 복구하는 방식에 대해 설명드리겠습니다. 먼저 open ssl은 암호화 프로토콜로, AES 알고리즘을 사용합니다. AES 암호 방식은 128bit의 플레인 텍스트를 암호화하여 128비트의 비밀 텍스트로 바꾸는 방식입니다. 암호화 과정에서는 1round라는 연산 단위가 들어가며, 방식에 따라 다른 라운드 횟수를 거칩니다. 저희가 사용한 aes 128 알고리즘은 10 round를 실행합니다.

연산을 위해서는 비밀키인 사이퍼 키와 값을 랜덤으로 변환하기 위한 lookup table인 t-table이 필요합니다. 저희는 4개의 t-table에 대해 flush reload 공격을 수행합니다. 즉 , 피해자가 t-table에 접근한 값을 추출하여 개인키를 복원하는 실험을 진행하였습니다. 각 round에서 필요한 라운드키는 이전 라운드 키에 의해 생성되므로, 마지막 round key를 추출하여 aes 알고리즘을 역 연산하면 원래의 비밀키를 추출할 수 있습니다.

진행방식은 open ssl에서 제공한 t-table의 offset을 설정합니다.

그 다음 AES 암호화 키를 생성한 후, 각 table마다 암호화하는데 걸린 시간을 추출하여 임계값과 비교 후, 피해자가 접근한 t-table 값을 추출합니다. 이를 기반으로 알고리즘 역연산을 하여, 2. 에서 만든 개인키를 복원할 수 있습니다.

두환경에서 정확도를 나타내는 그래프입니다. 저희가 앞에서 언급한 추천 임계값을 기준으로, threshold를 기준보다 작게 했을 떄, 그리고 크게 했을 떄의 키 복원률을 나타내었습니다. 이처럼 다른 threshold를 적용했을 때 각각의 정확도가 다름을 알 수 있습니다.

여기까지 저희가 알아본 내용과 실제 재현한 실험에 대해 설명을 했습니다. 사실 이 실험에는 한계점이 존재합니다. Flush reload attack의 장점은 다른 코어 상에서의 공격이 가능하다는 점이었는데, 저희는 우선적으로 t-table에서의 timing difference에 의한 키 복원에 초점을 맞추었기 때문에 같은 코드 상에서 어쩌면 공격자만 존재하는 방식으로 진행을 하게 되었습니다. 추후에는 cross-core 환경에서의 flush reload attack에 대하여 공격방식을 연구할 예정이며, threshold를 좀 더 정확히 알아내기 위한 방법에 대해 연구할 예정입니다. 또한 새로운 방식의 cache timing attack을 알아보고 교수님께서 언급해주신 cache set 방식을 랜덤화하였을 때의 생기는 영향에 대해서 연구할 계획입니다.