랜섬웨어 공격 방어 SSD에서 선점적 랜섬웨어 감지를 통한 Foreground I/O 지연 시간 단축 기법 연구

박지윤[1], 민동현[1], 이중희[2], 김영재[1]
서강대학교 컴퓨터공학과, 고려대학교 정보보호학과
{owo1113, mdh38112, youkim}@sogang.ac.kr {j_lee}@korea.ac.kr

Reducing Foreground I/O Latency via Preemptive Ransomware Detection on Ransomware Attack Tolerant SSD

Jiyun Park[1], Donghyun Min[1], Junghee Lee[2], Youngjae Kim[1]
[1]Department of Computer Science and Engineering, Sogang University, Seoul, Republic of Korea
[2]School of Cybersecurity, Korea University, Seoul, Republic of Korea

요 약

Soild-State Drive (SSD)는 덮어쓰기 요청 수행 시 이미 쓰여진 페이지 대신 새 페이지에 데이터가 기록되는 다른자리 덮어쓰기 (Out-of-place update) 를 수행한다. SSD 는 이전 데이터가 쓰여진 페이지를 무효화시킨다. Ransomware-tolerant SSD는 이 무효화 페이지들을 백업으로 관리하여 추가적인 데이터 공간 비용 지불 없이 버저닝을 수행하여 랜섬웨어 공격에 대응한다. SSD 내부의 페이지 버져닝 기술은 빠른 속도의 높은 랜섬웨어 감지율와 높은 백업 공간 효율, 그리고 낮은 I/O 대기시간을 요구한다. 이 세 가지를 모두 만족하기 위해 최근 Content-based 인라인 감지를 수행하는 Direct Memory Access (DMA) 확장 하드웨어 모듈 설계가 제안되었다. 하지만 이 하드웨어적인 접근방법은 특수한 하드웨어 모듈을 요구한다. 따라서 SSD에 범용적으로 적용하기에는 제한적이다. 본 연구에서는 (i) Content-based Inline 감지 작업을 분석하여 합리적인 선점 정책을 설계하고 이를 바탕으로 (ii) 선점적 랜섬웨어 감지 기술을 Microsoft SSD 시뮬레이터에 구현하였다. 실험 결과, Erebus 리눅스 랜섬웨어 워크로드에서 기존의 SSD에 비해 포그라운드 I/O 응답 지연 시간이 거의 증가하지 하지 않음을 확인하였다.

1. 서 론

랜섬웨어는 멀웨어의 일종으로 사용자의 데이터를 암호화 시킨 후, 사용자가 데이터 복구를 위한 비용을 지불하도록 위협한다. 랜섬웨어로 인한 피해 금액은 지속적으로 증가하고 있기 때문에 해결책의 필요성이 증대되고있다 [1]. 랜섬웨어의 공격을 방어하기 위한 백업 기술이 활발히 연구되어 왔다 [2]. 전통적으로 백업은 호스트 단에서 수행되어 왔다. 하지만 복사본을 위한 명시적인 백업 수행은 결과적으로 스토리지에 추가적인 공간을 요구하고 OS가 손상될 경우에 복사본 역시 파괴될 수 있다는 단점이 있다.

이와 달리 SSD (Solid State Drive) 스토리지 단에서의 백업 기술은 공간 효율적이고 OS가 손상된 경우에도 안전하다 [3, 4, 5]. SSD에서의 백업 기술은 기본적으로 SSD의 다른자리 덮어쓰기로 인해 무효화되는 데이터 페이지를 없애지않고 백업으로써 유지하기 때문에 백업본을 위한 명시적인 공간이 필요없다. 또한 SSD 펌웨어는 OS와 분리되어 있다. 따라서 권한을 가진 랜섬웨어로 인해 호스트 OS가 손상된 경우에도 복사본은 안전하게 지켜질 수 있다.

SSD 단에서의 백업 기술들은 공통적으로 SSD 안에서 랜섬웨어의 I/O 를 감지하고 이 경우에만 백업을 수행한다. 만약 모든 I/O에 대해 백업 복사본을 유지하면 공간 오버헤드가 매우 커지기 때문이다 [4]. Amoeba [5]는 SSD에서 매 쓰기 I/O 마다 Content-based 랜섬웨어 감지 알고리즘을 수행하여 높은 랜섬웨어 I/O 감지 정확도를 확보하였고 결과적으로 백업 공간 효율을 높힌 최신 연구이다. 하지만 이러한 매 쓰기 I/O 마다의 Content-based 감지는 데이터가 저장된 기존 페이지를 플래시 메모리로부터 읽는 과정, 알고리즘 수행을 위한 많은 양의 바이트 단위 계산이 필요하기 때문에 I/O 처리가 늦어진다. 따라서 I/O 대기시간이 커지는 문제가 발생한다. 이를 해결하기 위해 Amoeba는 SSD 내부에 DMA (Direct Memory Access) 하드웨어 가속기를 부착하였다. 이 내부 DMA는 플래시 메모리와 DRAM 간의 데이터 전송 과정 중에 랜섬웨어 감지 알고리즘을 고속으로 수행한다. 따라서 I/O 대기시간이 길어지는 문제를 해결하였다.

하지만 이 기술은 대부분의 벤더 (vendor) 에게 DMA 전문 기술과 추가적인 하드웨어 비용을 요구한다. 위 문제를 해결하기 위해, 우리는 펌웨어 수정으로 Content-based 감지를 기존 SSD에 비해 성능저하 없이 수행하는 것을 목표로 한다. 소프트웨어적으로 Content-based 인라인 감지를 구현하기 위해, 우리는 기존 연구에서 해결하지 못한 두 가지 문제점을 해결해야 한다.

첫째, 하나의 쓰기 I/O를 처리하는 시간 자체가 길어진다. 이를 확인하기 위해 소프트웨어 방식의 Content-based 인라인 감지 기법을 Cosmos+ OpenSSD [6]에 구현하고 감지로 인한 시간 딜레이를 측정하였다. 우리의 실험에 따르면 감지 딜레이는 총 약 1.15 ms 로 나타났다 (섹션 2.2). 이 수치는 SLC NAND Flash 기준으로 보고된 Erase latency에 가까울 정도로 큰 수치이다 [7]. 따라서 기존의 쓰기 과정에 Content-based 인라인 감지를 펌웨어에서 수행하는 것은 많은 시간을 소요한다. 둘째, 많은 시간을 필요로하는 인라인 감지 수행 도중 I/O 요청이 들어올 경우 요청에 곧 바로 응답할 수 없으므로 I/O 응답 시간이 길어지는 문제가 발생한다.

이 문제들을 해결하기 위해 본 연구에서는 랜섬웨어를 즉시 감지하는 기존의 인라인 감지 도중 I/O 선점을 허용하는 선점적 랜섬웨어 감지 기술을 제안한다. 이를 위해 인라인 감지 작업을 분석하여 선점 지점을 선정한다. 또한 선점을 위해 감지 상태를 저장하여 선점 이후 감지를 이어수행할 수 있도록한다. 따라서 감지 도중 들어오는 I/O를 우선적으로 수행한 후 다시 감지로 돌아오기때문에, I/O 응답이 크게 지연되는 것을 막을 수 있다. 우리는 이와 같은 시스템을 Microsoft의 Disksim 시뮬레이터 [8]에 구현하고 실험하였다. 실험결과, 인라인 감지에 비해 약 23.6 배의 성능 향상을 확인하였으며 이는 기존 SSD의 성능과 유사한 것을 확인하였다.

이 논문은 삼성전자 반도체의 연구 지원과 2017년도 정부(과학기술정보통신부)의 재원으로 한국연구재단-차세대정보 • 컴퓨팅기술개발사업의 지원을 받아 수행된 연구임 (No. 2017M3C4A7080243)

2. Background

2.1 랜섬웨어 감지 및 방어 SSD

현재 연구된 랜섬웨어 공격을 방어하는 SSD는 페이지에 대한 쓰기 요청이 SSD에 올때마다 랜섬웨어 감지를 곧 바로 수행한다. 우리는 이를 인라인 감지라 부른다. 쓰기 요청은 새 페이지에 쓰고 예전 페이지는 백업 상태로 바꾸어 예전 데이터를 유지한다. Amoeba [5]는 similarity 와 entropy 같은 Content-based 랜섬웨어 감지 기법을 활용하여 정확한 인라인 감지를 수행하는 최신 연구이다. Similarity는 이전 데이터와 새 데이터의 byte-level에서의 차이를 의미한다. Entropy는 Shannon entropy이며, 이는 새로운 데이터의 무질서도를 의미한다 [9].

매 쓰기 I/O 마다의 Content-based 랜섬웨어 감지 알고리즘 계산으로 인한 성능 딜레이를 방지하기 위해, Amoeba는 SSD 내부에 새로운 DMA (Direct Memory Access) 하드웨어 가속기를 제안하였다. Amoeba는 DMA 하드웨어 가속기의 도움을 받아 SSD 내부에 similarity와 entropy가 통합된 감지기를 만들어서 정확하게 페이지의 감염 여부를 확인을 한다. Amoeba 는 감염된 페이지에 대해서만 백업을 수행하여 백업 공간 효율을 높혔다. 또한 DMA 전용 하드웨어를 사용하기 때문에 성능에 실질적으로 미치는 계산 지연시간을 최소화하여 낮은 I/O 대기 시간을 조건도 만족시켰다.

2.2 Content-based 인라인 감지 작업 분석

우리는 Amoeba와 달리 별도의 DMA를 필요로 하지 않고, 펌웨어 수정만으로 Content-based 인라인 감지를 수행하고자 한다. 펌웨어상에서 Content-based 인라인 감지가 구현되었을 경우의 성능 저하의 원인을 파악하고 분석하기 위해, 우리는 인라인 감지의 세부 작업들을 분류했고 각 작업들이 야기하는 latency를 측정하였다. 각 세부 작업은 구체적으로 표 1과 같이 4단계를 거쳐 수행된다. 우리는 D_{old} 를 이미 쓰여진 페이지 내용, D_{new} 는 새 페이지 쓰여질 내용, 덮어쓰기 요청을 W라 가정한다.

표 1: SSD 내부에서 인라인 랜섬웨어 감지 단계

Steps (in order)	Description
Step 1	Wait until the new data (D_{new}) from the host is uploaded to DRAM.
Step 2	Wait until the previous data (D_{old}) from existing page in NAND is uploaded to DRAM.
Step 3	Byte by byte similarity comparison between D_{new} and D_{old}.
Step 4	Shannon entropy calculation for D_{new}.

Step 1과 Step 2 의 경우, similarity와 entropy 계산을 위해 CPU의 연산 능력이 필요하므로 데이터를 플래시 메모리에서 DRAM 으로 읽는 작업이 필요하다. 특히, Step 2의 경우는 similarity 계산을 위해 D_{old} 에 대한 읽기도 필요하다. Step 3와 Step 4의 경우는 공통적으로 페이지내 데이터를 바이트 단위로 접근한다. 왜냐하면 similarity와 entropy가 byte-level 알고리즘이기 때문이다. 한 번의 바이트 단위의 접근이 이루어지면 similarity 계산을 위해 바이트 마다의 비교를 수행하고 entropy 를 위해 byte occurrence 를 센다. 우리는 위와 같은 인라인 감지의 세부 작업들을 Cosmos+ OpenSSD [6]에 구현하였다. 인라인 감지 각 단계가 순차적으로 수행되지만 바이트 단위의 접근을 하고 메모리에 저장해두면 similarity 뿐만 아니라 entropy 계산에서 다시 사용할 수 있다. 이와 같은 방식의 감지 작업들의 총 latency 는 약 1.15 ms 로 측정되었다.

3. 선점적 랜섬웨어 감지 기술

Content-based 인라인 감지로 인해 생기는 시간 딜레이는 포그라운드 I/O 에 대한 간섭을 일으켜 결국 I/O 요청 응답

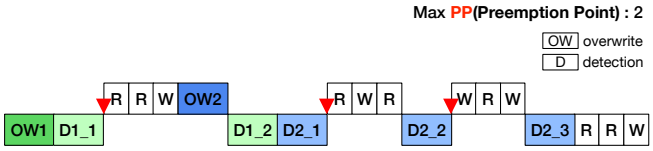


그림 1: 인라인 감지 도중 I/O 선점

시간이 길어지고 성능 저하 문제를 야기한다. 우리는 인라인 감지 수행 중에도 I/O 요청에 대한 처리가 가능하게 만들기 위해 선점을 허용하는 랜섬웨어 감지 기술을 제안한다. 선점은 수행하던 작업을 멈추고 도중에 포그라운드 I/O 요청을 수행하는 것을 말한다. I/O의 선점을 허용할 경우 랜섬웨어 감지 작업보다 우선적으로 I/O 요청을 즉각적으로 수행하므로 I/O 응답 시간을 줄일 수 있는 대안이다.

선점적 랜섬웨어 감지 기술은 감지를 수행하는 시간동안 선점 지점에서 I/O 요청을 수행한다. 선점 지점이란 탐지를 수행하는 도중 선점해야할 I/O 요청이 들어왔는지 검사하는 시점을 의미한다. 따라서 인라인 감지 수행 도중, 우리는 매 선점 지점마다 서비스해야할 외부 I/O 요청이 요청 큐에 존재하는지 검사한다. 만약 서비스해야할 I/O 요청이 한 개 이상 있을 때, 감지 작업은 일시정지되고 현재 감지 상태를 저장한다. 이후, I/O 에게 CPU 자원을 넘겨주는 선점이 수행된다. 반면, 수행해야할 I/O 요청이 없으면 감지를 계속 수행한다. 인라인 감지 도중에 선점을 하는 방법은 3가지 경우가 있을 수 있다.

Case 1: 선점 하지 않는 경우로 인라인 감지을 그대로 수행한다.

Case 2: Step 사이에서만 선점을 허용하면, 99 %를 차지하는 Step 3와 Step 4 latency 만큼 포그라운드 I/O 성능이 저하된다.

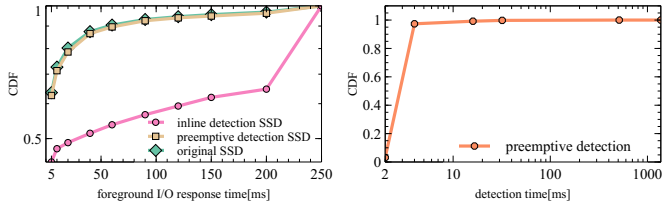
Case 3: 선점 지점마다 선점을 허용하여 I/O요청 있는 경우 이를 수행한 후, 다시 감지을 이어 수행한다.

따라서 가장 합당한 Case를 선정하기위해서 인라인 감지 과정을 분석해야한다.

선점 지점 설정: 선점 지점을 정하기위해 2가지가 분석되어야 한다. 첫째, 감지 과정 중 가능한 전체 선점 영역을 확인해야 한다. 인라인 감지 중 Step 1과 Step 2 과정 도중 I/O 의 선점을 허용하고 I/O 수행을 시킨다면 이후에 다시 처음부터 감지 작업이 재수행된다. 왜냐하면 Step 1과 Step 2는 선점 친화적이지 않은 작업이기 때문이다. 반면, Step 3와 Step4는 도중에 작업이 멈추고 I/O 수행을 허용해주더라도 선점 이후에 감지 작업으로 돌아와 중단 시점부터 이어수행할 수 있다. 우리는 이러한 Step 3와 Step 4 를 선점 친화적 작업이라 한다. 따라서 우리는 선점 친화적 작업이 선점할 수 있는 영역으로 설정한다. 둘째, 가능한 전체 선점 친화적 지점 (Step 3 와 Step 4) 중 실제로 선점 지점을 얼만큼 둘지를 정해야 한다. 선점을 위해선 요청 큐에 쌓인 I/O 요청을 확인해야 하고 이 작업은 선점 instruction 수행과 메모리 접근의 오버헤드가 부과된다. 따라서 우리는 선점 오버헤드가 선점 친화적 작업의 딜레이를 넘지 않도록 선점 지점의 개수를 정한다. PGC논문 [10]에 따르면, 폴링 방식으로 요청 큐를 보는 작업은 600 ns의 시간이 소요된다. 이 선점 오버헤드는 600 ns로, 실제 우리가 관찰한 감지 latency의 0.0005 %로 매우 적은 비율을 차지한다. 따라서 우리는 선점 친화적 작업 (Step 3, Step 4)에 한하여 작업 도중 선점 지점의 최대 갯수를 $\frac{Latency\ Of\ Step3\ And\ Step4}{\text{선점오버헤드}}$ 로 둘 수 있다.

감지 상태 보존: 제안된 시스템은 인라인 감지를 수행하다가 선점 지점이 오면, 현재 수행되고 있는 감지의 상태를 기억해두어야한다.

- 현재 감지가 수행되고 있는 페이지의 PPN (Physical Page Number)을 저장해야한다. 선점이 끝나고 다시 감지로 돌



(a) I/O 응답 시간 (b) 감지 수행 시간

그림 2: 선점적 랜섬웨어 감지 기술 성능

아왔을 때, PPN을 보고 감지 수행중이던 페이지를 알 수 있다.

- 현재 감지에서 수행중인 Step을 저장해야한다. Step 1와 Step 2의 경우에는 PPN만 알면 재수행 가능하지만, Step 3와 Step 4의 경우에는 그렇지 못하다. 왜냐하면, 페이지의 데이터에 대해 바이트 단위의 접근을 수행하기 때문이다.
- Step 3와 Step 4의 경우를 대비해 페이지에서 접근중이던 데이터의 offset을 저장해야한다.

그림 1는 최대 선점 지점이 2개인 경우이다. 선점은 인라인 감지 도중 선점 지점를 만나면 시작된다. 선점이 시작되면, 현재 수행 중인 감지 (**D1_1**)에 대한 선점 상태를 저장한다. 그 다음, 요청 큐에 쌓여있는 I/O 요청이 있는지 확인한다. 쌓여있는 요청이 있다면, 모두 수행하고 감지 상태를 확인하여 감지로 돌아온다. 만약 이때, 새로운 덮어쓰기 요청이 들어와도 I/O 수행을 멈추지 않는다면, 감지 작업이 한없이 미루어질 수 있다. 따라서, 감지를 하지 못하기 때문에 GC(Garbage collection)을 수행할 수 없다. 따라서, SSD 공간 오버헤드를 증가시킨다. 또한, 감지 상태를 메모리에 저장하여 두고 있어야 하기때문에 메모리 오버헤드도 증가시키는 문제가 발생한다. 따라서, 그림 1과 같이 새로운 덮어쓰기 요청 (**OW2**)이 수행되는 경우에는 I/O 수행을 멈추고 다시 감지 작업 (**D1_2**)으로 돌아온다. 반면, 기다리고 있는 요청이 하나도 없다면 I/O수행 없이 바로 감지를 이어 수행 한다.

4. 실험 평가

제안 된 선점적 랜섬웨어 감지 기술을 구현하고 평가하기 위해 Microsoft Research에서 개발 한 DiskSim 시뮬레이터 [8]를 변형시켰다. 실험의 작업량을 위해 총 크기가 약 5 GB 인 pdf, image, doc 등 다양한 형식의 파일을 미리 생성한다. 그 후, Erebus 랜섬웨어 [11]를 통해 실험을 수행했다.

제안된 설계에 대한 성능을 평가하기 위해 감지 시스템이 없는 기존 SSD, Content-based 인라인 감지를 수행하는 SSD 그리고 제안하는 선점적 감지를 수행하는 SSD를 비교군으로 두었다. 선점적 감지 SSD 는 선점 친화적 작업 영역안에서 50개의 선점 지점의 개수를 두고 I/O 응답 시간의 변화를 측정하였다. 그림 2(a) 를 보면 우선 기존 SSD에 비해 인라인 감지의 I/O 응답시간의 평균이 약 23.7 ms 에서 622.6 ms 로 26.3배 정도로 심각하게 지연되는 것을 확인할 수 있다. 왜냐하면, 인라인 감지로 인해 overwrite과정이 길어지고, 따라서 다음에 들어오는 I/O에 응답하는 시간들이 길어지기 때문이다. 반면, 선점적 감지를 수행하는 시스템의 경우 인라인 감지에 비해 약 622.6 ms 에서 26.4 ms 로 23.6배 I/O 응답 시간을 낮추었다. 또한, 기존 SSD에 비해선 오직 약 0.9배의 적은 성능 저하를 보인다. 선점적 감지 기술은 감지 도중에 I/O를 수행하여 I/O의 응답시간을 줄이기 때문이다.

그림 2(b)에서는 섹션 3에서 서술한 감지 정책의 효용성을 증명했다. 제안한 선점적 랜섬웨어 감지 기술에서는 선점 도중 I/O 중에 overwrite 요청이 들어올 경우, 선점을 중단하고 기존의 감지 작업으로 돌아간다. 따라서, 감지 작업이 한없이 미루어지는 문제가 발생하지 않기 때문에 감지 수행 시간이 전체적으로 짧아진다. 실험 결과에 따르면 총 감지 작업 중 99%가 약 16 ms이내에 수행이 완료되어 감지 작업이 길어지는 문제를 해결하였다.

5. 결 론

본 연구에서는 감지 정확도가 높다고 알려진 content-based 감지 작업을 SSD에서 소프트웨어적으로 수행할경우 발생하는 포그라운드 I/O 성능저하를 해결하였다. 문제를 해결하기위해서, 인라인 감지 도중 포그라운드 I/O의 선점을 허용 하는 선점적 랜섬웨어 감지 기술을 제안한다. 또한, 인라인 감지 과정을 분석하여 합리적인 선점 정책 설계하고, 이를 바탕으로 선점적 랜섬웨어 감지 기술을 Microsoft SSD 시뮬레이터에 구현하였다. 실험 결과로, Erebus 리눅스 랜섬웨어 워크로드에서 기존의 SSD에 비해 포그라운드 I/O 응답 지연 시간이 1.1배로 거의 증가하지 하지 않음을 확인하였다.

참고 문헌

[1] V. Wilson, "Recent ransomware attacks in 2019." https://spinbackup.com/blog/24-biggest-ransomware-attacks-in-2019/.

[2] A. Continella, A. Guagnelli, G. Zingaro, G. De Pasquale, A. Barenghi, S. Zanero, and F. Maggi, "ShieldFS: a self-healing, ransomware-aware filesystem," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pp. 336–347, ACM, 2016.

[3] S. Baek, Y. Jung, A. Mohaisen, S. Lee, and D. Nyang, "SSD-Insider: Internal defense of solid-state drive against ransomware with perfect data recovery," in *Proceedings of the 38th International Conference on Distributed Computing Systems (ICDCS)*, pp. 875–884, IEEE, 2018.

[4] J. Huang, J. Xu, X. Xing, P. Liu, and M. K. Qureshi, "FlashGuard: Leveraging intrinsic flash properties to defend against encryption ransomware," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 2231–2244, ACM, 2017.

[5] D. Min, D. Park, J. Ahn, R. Walker, J. Lee, S. Park, and Y. Kim, "Amoeba: An autonomous backup and recovery ssd for ransomware attack defense," *IEEE Computer Architecture Letters*, vol. 17, no. 2, pp. 245–248, 2018.

[6] Y. H. Song, "Cosmos+ OpenSSD Platform." http://www.openssd.io/, 2017.

[7] J. Hruska, "How Do SSDs Work?." https://www.extremetech.com/extreme/210492-extremetech-explains-how-do-ssds-work/.

[8] V. Prabhakaran and T. Wobber, "SSD extension for DiskSim simulation environment," *Microsoft Reseach*, 2009.

[9] N. Scaife, H. Carter, P. Traynor, and K. R. Butler, "Cryptolock (and drop it): stopping ransomware attacks on user data," in *Proceedings of the 36th International Conference on Distributed Computing Systems (ICDCS)*, pp. 303–312, IEEE, 2016.

[10] J. Lee, Y. Kim, G. M. Shipman, S. Oral, F. Wang, and J. Kim, "A semi-preemptive garbage collector for solid state drives," in *Proceedings of the International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 12–21, IEEE, 2011.

[11] T. MICRO, "Erebus linux ransomware: Impact to servers and countermeasures." https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/erebus-linux-ransomware-impact-to-servers-and-countermeasures, 2017.