Internet **붕괴의 기술적 원인 분석**

===

Internet 마비에 대한 기술적인 원인 분석 (공격기법 분석 갱신. Ver. 0.2)

2003년 1월 26일

(revision 2003년.1월 27일)

작성자: 전 상훈 (nickname: 바다란)
(winsnort@hotmail.com , winsnort@securityindepth.net)

Internet 붕괴의 기술적 원인 분석

==

@ 개요

revision 개요: IDS 로그 및 기타 로그를 확인해 보았을 때 1433번 포트로의 접속은 거의 없었으며 1434번 포트로의 UDP 접속이 거의 대부분이라 공격 패턴 판단을 잘못 하고 있다는 생각 하에 재갱신 하게 되었습니다. 공격패턴을 수정합니다. 시나리오도 수정 했습니다.

이전 문서는 : http://my.netian.com/~mil21/internetcrisis0.pdf

안녕하세요. 바다란 입니다. 긴말 하지 않고 시간이 많지 않으니 바로 본론으로 들어 가겠습니다. 또 문제가 된 상황에 대해서 제가 파악한 내용대로 해석 하겠습니다.

공격 코드에 대한 분석과 취약점을 면밀히 검토하여 현재 일어난 사건들에 대해 하나하나 규명해 보겠습니다.

문제는 SQL 웜이였다. SQL 서버에서 공격 대상이 된 서비스는 Resolution 서비스로서 이 서비스에 대해서 알아 둘 필요가 있다. MS SQL 서버에서 Data를 저장하는 영역을 여러 개로 나눌 수 있는데 이 하나의 영역 자체를 instance라 칭하며 SQL 서버와의 통신에 사용하는 포트는 1433번 포트로만 사용이 가능하다. 하나의 SQL 서버에 여러 개의 Instance가 존재 한다는 말은 한대의 SQL 서버로 여러 개의 Database를 나누어서 사용한다는 말과 동일하다 할 수 있다. 그러면 여러 개의 Database가 하나의 서버에 존재할 때 통신을 할때 어떻게 통신을 할 수 있을까?. 1433번 포트에 Listening하는 데이터의 Instance는 단 하나이며 여러 개가 있을 경우에는 특정 포트에 접속하여 찾는 방식이 아닌 1434번 포트를 이용하여 다른 Instance를 찾게 된다. 이 1434번 포트가 Resolution 서비스가 된다.

만약 한대의 서버에 a 라는 쇼핑몰 a 사의 DB와 B라는 쇼핑몰 B사의 DB가 동시에 존재할때 먼저 설치된 a 사의 DB만 1433번에서 listening 하고 있는 것이 된다. 그럼 다른 DB는 어떻게 될까?

1433번 포트가 아닌 다른 임의의 포트에서 listening 하도록 세팅이 가능하도록 되어 있다. 하나 하나의 별개의 DB를 instance라고 하며 이 임의의 포트에 배정된 instance를 찾기 위해 Resolution 서비스가 이용이 된다. 이 instance는 로컬에 있는 것만 해당되지 않으며 원격의 instance도 동일하게 해당이 된다.

✍✍추가정보 추가: 1434번 포트는 두 가지 용도로 사용이 되고 있음을 추가로 확인 했다. 먼저 들어오는 트래픽에서 첫번째 바이트가 0x04일 경우에 Monitoring thread가 실행이 되며 Keep –alive 매커니즘의 경우 단일 바이트로 0x0A 시그널을 받았을 경우에 동일한 0x0A 의 single byte packet을 반응함으로써 매커니즘이 구현 되어 있다.

Internet 붕괴의 기술적 원인 분석

==

자 설명이 길었다. 이정도면 Resolution 서비스에 대한 설명은 충분히 되었으리라 보고 이제 문제의 상황들에 대해서 하나하나 짚어 보자.

I. Problem Solving

1. DNS 서버는 왜 죽었는가?

실제 문제해결에 투입된 인력의 언급이 아니였다면 단순한 도스공격에 죽었다고 생각 했을 것이다. 의문점을 풀기 위해 제게 보내준 메일 한통으로 인해 문제 해결이 쉽게 되었다고 생각한다.

왜 죽었는가?.. UNIX BIND를 쓰는 KT DNS에 초당 Query가 10배 증가한 것을 질문 했었는데 처음에는 단순한 DDos 공격에 의해 죽은 것으로 판단을 했었다. 단순 DDos 공격이 아닌 DNS 쿼리가 순식간에 10배 가량 증가해서 서버가 처리 업무를 다 처리하지 못하여서 과부하에 의해 죽었다는 것을 알게 되었다. 그러면 이것은 왜 죽었는가?

Resolution 서비스에 그 해답이 있다. SQL 서버의 메뉴에 보면 새 Instance를 추가 하는 부분이 있다. 여기를 누르게 되면 1434 Port로 Resolution 서비스를 위한 udp packet을 보내게 되고 여기에 대해 반응을 하게 됩니다. 즉 instance가 존재하는 것을 알려주는 패킷을 보내주게 되겠죠. (정확한 반응에 대해서는 유추 한 것입니다. 내부 로직을 알 수 없기 때문에 보이는 현상으로 또 경험으로 유추한 것입니다.)

자 SQL 서버 사용 해 보신 분은 아실 것입니다.
Add New instance 메뉴를 누르게 되면 한참동안의 서버가 구동 된후 instance 네임으로 나오는 여러 DB 목록을 볼 수 있습니다. 여기에 문제가 있는 것입니다.

비단 자신의 서브넷에 존재하는 DB 뿐 아니라 다른 여러 DB 목록도 보셨을 겁니다.
이런 Resolution Service에 의해 서브넷의 범위와 클래스 대역의 범위를 벗어나는 문제가 생긴 것이죠.

Internet 붕괴의 기술적 원인 분석

===

이 부분은 MS의 SQL 개발팀에서 DDos에 관련된 문제를 광범위 하게 일으킬 수 있다는 것에 대해 책임으로 부터 자유로울 수 없을 것입니다. 핫픽스의 경우도 예측 하건데 이런 Instance를 찾아오는 부분에 대해서는 동일하며 다만 감염되는 부분 (1434 Monitoring thread의 함수 인자를 체크하지 못한 두 함수의 버퍼오버런 버그)에 대한 조치만 취해져 있을 것입니다. (물론 이부분은 MS의 핫픽스 코드를 볼 수 없기 때문에 제가 느끼는 예상입니다.)

1434번 port로 첫번째 바이트가 0x04 이면 Monitoring thread가 가동이 되며 이 Monitoring thread에 감염을 시도하는 트래픽이고 1434포트는 감염이 된 후에 이루어 지는 트래픽이라 판단 됩니다. DNS 쿼리의 증가는 1434번을 통해 감염된 SQL 서버가 공격 코드내에서 계산된 임의의 주소로 single packet 인 0x0A 를 보내게 되면 SQL Server 자체가 DB 서버의 instance를 쿼리하는 과정에서 DNS쿼리가 급격하게 증가하여 죽은 것으로 확신 합니다.

* DNS 쿼리가 왜 발생하는지에 대한 질문이 많아서 답변으로 보낸 글을 발췌 하였습니다.
[만약 두대의 서버에 똑같은 instance 명을 가지고 있는 DB 가 존재하면 어떻게 됩니까?
그냥 Instance 명만 읽어 오게 되면 이 두개의 Instance 를 구분할 수가 없게 됩니다.

그래서 하나의 Instance 로 부터 Keep alive 패킷이 오게 되면

패킷을 받은 Resolution 서비스는 자신의 Instance가 살아 있음을 알려주게 됩니다.

이 응답을 받은 서버에서는 SQL 서버 자체에서 Instance를 구분하거나 접속하기 위한 정보들을 가져오게 되겠죠. 여기에는 도메인명이나 IP등도 기본적으로 포함이 될 것입니다.
이렇게 생각하시면 DNS 쿼리는 일어 납니다. 그리고 무한루프를 돌고 있으니 계속 적인 DNS 쿼리가 발생하며 쿼리를 수행하는 자체는 SQL Server가 되는 것이죠.

즉 DNS 를 다운시킨 직접적인 시스템은 공격 코드가 아닌 SQL Server가 되는 것입니다.]

2. 왜 그렇게 폭발적으로 트래픽이 늘었는가?

이 문제에 대해서는 우선 Resolution service의 취약점에 대해서 알고 있어야 합니다.
SQL 의 instance를 찾아 내는 로직에는 Keep-alive 로직이 있다고 한다. MS02-039

Internet 붕괴의 기술적 원인 분석

===

Article 참고 .

이 Keep-alive 로직이라는 것은 DB 서버내의 instance나 주위의 DB instance 에서 활성화 되어 있고 현재 활성화 되어 있지 않은 instance의 구분을 위해 사용이 됩니다. Instance가 계속 활성화 되어 있는지 또는 등록된 instance가 서비스가 죽어 있는지 확인을 지속적으로 수행하는 매커니즘이라고 한다.

악의적인 사용자가 keep-alive 패킷을 만들 수 있으며 이것을 Resolution 서비스(UDP 1434 port)에 임의의 IP Address를 조작하고 시작 byte로 0x0A single packet을 준비한 후 임의의 IP Address에 해당하는 SQL Server로 keep-alive 패킷(0x0A)을 전송하게 된다. 만약 위조된 IP Address에 SQL Server의 Resolution (UDP 1434 port) 포트가 활성화 되어 있다면 서로간의 Keep-alive packet(0x0A) 을 교환하는 무한루프에 빠질 수 있다. 이 루프가 계속 되게 되면 데이터를 교환하는 시스템 자체의 Resource를 계속 높이게 되는 것이다.

즉 정리해 보면 트래픽에 흘러간 패턴은 행위에 따라 네 가지로 볼 수 있다.

1). 1434 Monitoring port를 직접 공격한 패턴 (주된 감염 시도 패턴)
(버퍼 오버런을 일으켜서 SQL Server가 운영되는 System 레벨의 권한에서 임의의 명령을 실행할 수 있는 능력을 가지게 된다.)
First byte : 0x04로 시작하는 패턴

2). 공격 당한 후 임의의 IP 주소 및 자신의 IP 로 keep-alive packet을 보내는 패턴 1434 UDP port로 통신 (DDos 패턴)
단일 byte: 0x0A

3). keep-alive packet을 받은 서버에 SQL 서버가 존재할 경우 keep-alive packet을 받은 서버와의 never ending cycle (DDos 및 상호 서버간의 Resource 소비를 급격하게 증가 시킴)
단일 byte: 0x0A의 상호 교환

4). Keep-alive packet을 받은 SQL 서버에서 instance를 찾기 위한 프로세스 (DNS query 해당 가능성 높음)
SQL Server 자체에서 instance를 찾기 위한 프로세스에서의 DNS(53) Query packet

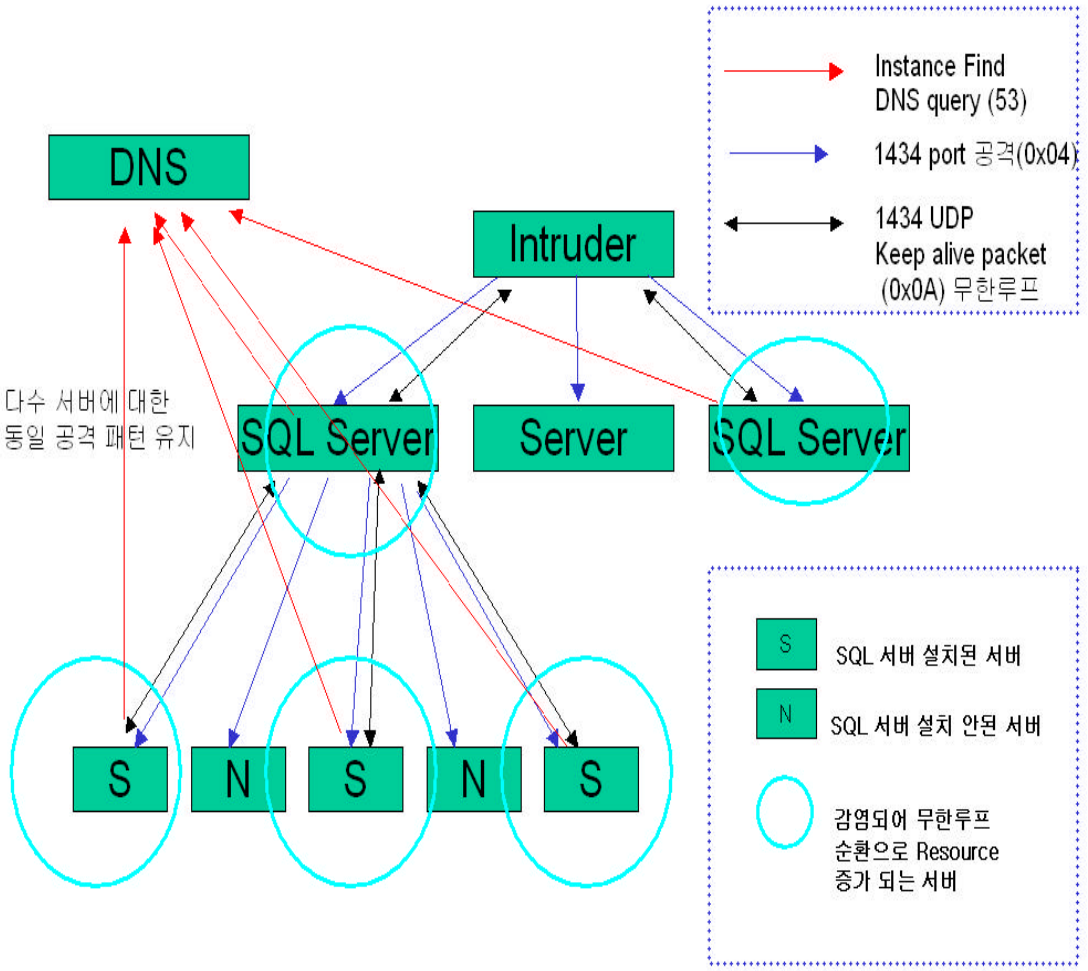
이런 트래픽의 복합적인 문제로 인하여 기반망의 붕괴가 이루어 졌음을 개인적인 조사 및 예상에 의해 판단하는 바이다.

Internet 붕괴의 기술적 원인 분석

===

앞의 두 가지 경우와 분석을 그림으로 표시하면 다음과 같다.

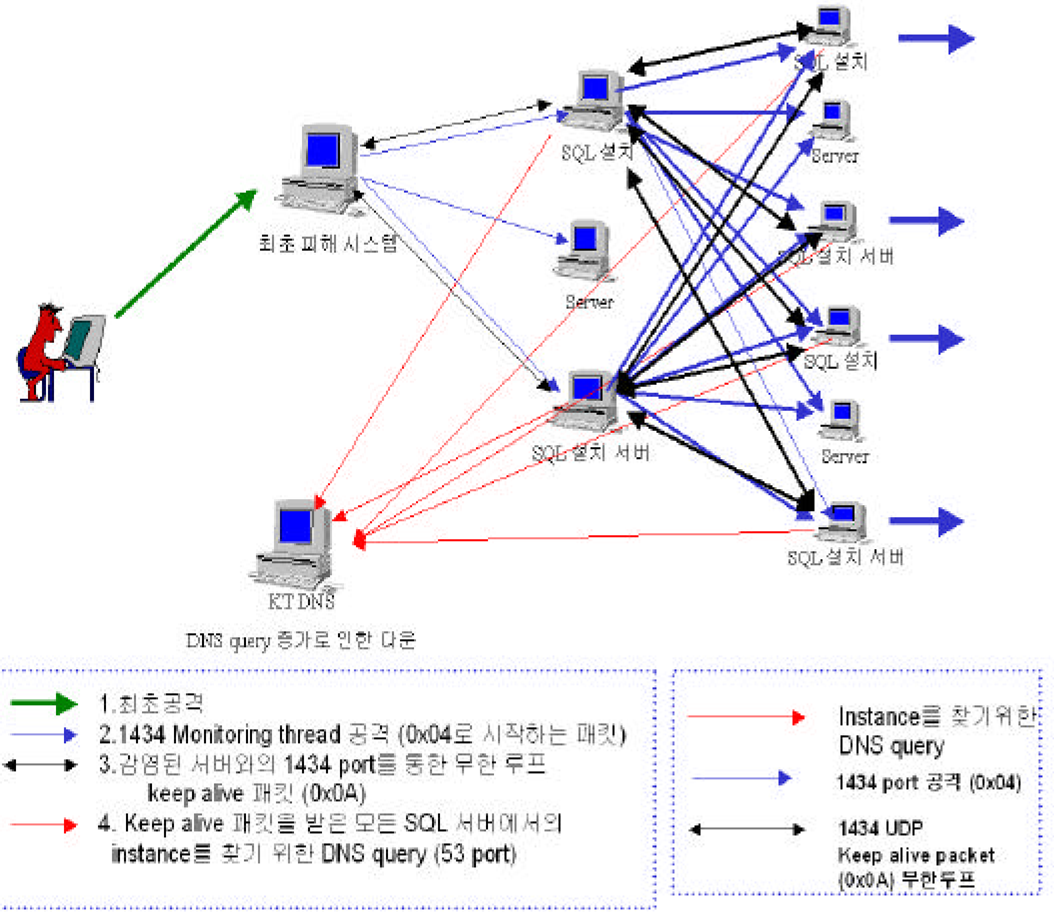
하나의 감염 된 서버는 256개 이상의 쓰레드 생성 후 재 공격을 가한다고 생각하면 트래픽의 양 및 감염 시도를 지속적으로 시도하는 서버의 규모는 대단히 많을 것으로 쉽게 예측이 가능하다.



이해하기 쉽도록 하나의 그림을 더 보도록 하자.

Internet 붕괴의 기술적 원인 분석

===



→ 1.최초공격 시도

→ 2.1434 Monitoring Port에 대한 버퍼오버런 공격 (0x04)로 시작

↔ 3.감염된 서버와의 1434 port를 통한 무한 루프

keep alive 패킷 교환 (0x0A)

→ 4. Keep alive 패킷을 받은 모든 SQL 서버에서의

instance를 찾기 위한 DNS query (DNS port 53)

1. 0x04로 시작하는 packet을 1434 포트에서 받으면 Monitoring thread가 가동되는데 이 기능에서 특정 문자열 길이를 넘겼을 경우 버퍼오버런 현상이 일어나며 여기에서

Internet 붕괴의 기술적 원인 분석

===

임의의 코드를 실행시킴 (임의의 코드는 keep alive packet 전송 및 공격코드 전송)

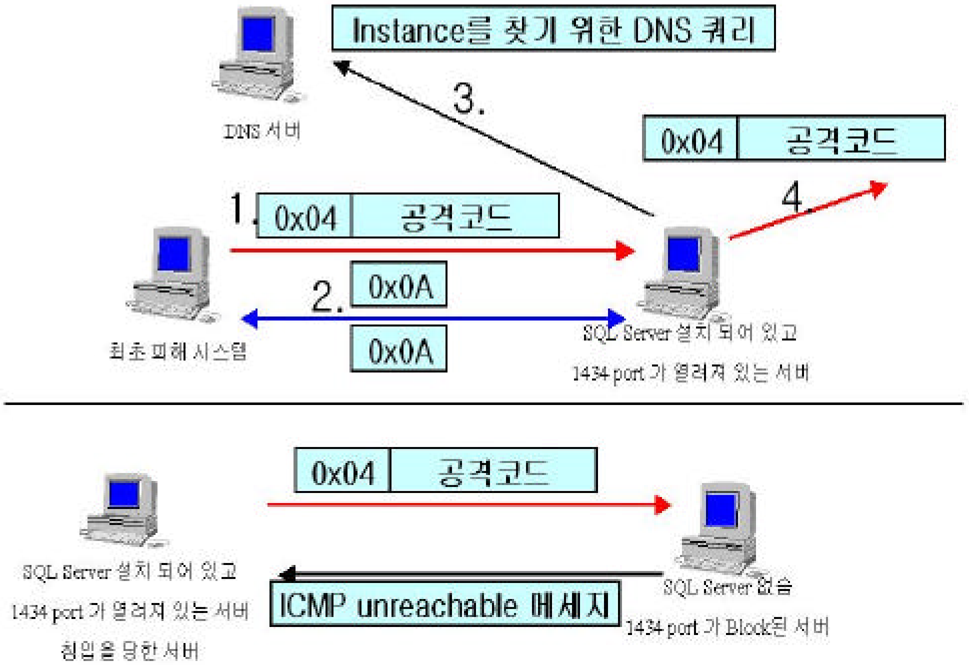
2. 침해 시스템에서 1434포트가 활성화 되어 있으면 first byte 값으로 0x04를 가지는 버퍼오버런 코드와 이진 실행코드를 임의의 IP에 발송한다. 임의의 IP에 1434 port가 활성화 되어 있을 경우 감염시키게 된다.

3. 웜이 전체가 분석된 것이 아니라 전송부분만 disassemble 되어 있어서 내부 내용을 유추할 수 없으나 SQL Server 두 대만으로 구성해 놓았을 때 각 서버의 프로세스 점유율이 80프로 까지 올라간다는 관제업체 직원의 이야기를 미루어 보건대 내부 웜의 내용에 0x0A 단일 바이트 (Keep alive packet)전송 매커니즘이 들어 있는 것으로 판단 됩니다. 즉 침해 시스템에서는 공격 시스템과 0x0A keep alive packet을 무한으로 주고받으며 Resource를 잡아먹도록 되어 있다고 예측 합니다.

<u>아래 이미지의 순서는 실행 되는 순서를 예측한 것임</u>

Internet 붕괴의 기술적 원인 분석

==


Instance를 찾기 위한 DNS 쿼리
3.
DNS 서버
0x04
공격코드
4.
1.
0x04
공격코드
2.
0x0A
0x0A
최초 피해 시스템
SQL Server 설치 되어 있고
1434 port 가 열려져 있는 서버
0x04
공격코드
SQL Server 설치 되어 있고
1434 port 가 열려져 있는 서버
침입을 당한 서버
ICMP unreachable 메세지
SQL Server 없음
1434 port 가 Block된 서버

현재까지 국내외로 조사된 결과로는 웜의 전파방식과 코드에 대한 disassemble 루틴만 알려져 있으며 정확한 내부 코드에 대해서는 알려져 있지 않다. 전송 코드에 대한 자세한 디스어셈블 루틴은 http://www.techie.hopto.org/sqlworm.html 여기를 참고 하기 바란다.

II. Technical Analysis

1월 26일 올라온 eeye사의 공격코드에 대한 disassembly 루틴과 버그트랙 메일링 리스트에 올라온 여러 정보들을 종합해 보도록 하자.

```
=====================================================
생략..
...
entrypoint
                xor      ecx, ecx
                push     ecx
                push     ecx
                push     eax
                xor      ecx, 9B040103h
                xor      ecx, 1010101h
                push     ecx              ; 9A050002 = port 1434 / AF_INET
                lea      eax, [ebp-34h]  ; (socket)
                push     eax
                mov      eax, [ebp-40h]  ; ws2_32 base address
                push     eax
                call     dword ptr [esi] ; GetProcAddress
                push     11h
....
```

이 부분은 1434 포트에 Keep alive packet을 보내는 과정으로 추정이 된다. 1434 port에 keep alive packet을 전송하는 코드의 일부분으로 보여진다.

```
PRND:
                mov      eax, [ebp-4Ch]  ; Pseudo Random Algorithm Start
                lea      ecx, [eax+eax*2]
                lea      edx, [eax+ecx*4]
                shl      edx, 4
                add      edx, eax
                shl      edx, 8
                sub      edx, eax
                lea      eax, [eax+edx*4]
                add      eax, ebx         ; Pseudo Random Algorithm End
                mov      [ebp-4Ch], eax
                push     10h
                lea      eax, [ebp-50h]
                push     eax
                xor      ecx, ecx
```

```
==================================================
            push    ecx
           xor     cx, 178h
            push    ecx
           lea     eax, [ebp+3]
            push    eax
           mov      eax, [ebp-54h]
            push    eax
           call    esi            ; sendto
          jmp     short PRND     ; Jump back to Pseudo Random Algorithm Start
```

이 루틴은 임의의 주소루틴을 생성하고 전송하는 루틴이다. 앞부분의 생략된 부분에서는 Reverse Packet 방식에 따라 Worm과 EIP 그리고 Garbage 데이터의 Address를 변경하는 루틴이 들어 있다.

좀 더 많은 시간과 여유가 주어 진다면 정밀하게 분석해 보고 싶지만 허락 하지 않는 환경이 있다.

간단히 코드에 대한 분석을 했었고 여러 정황에 대한 판단 근거를 가지고 있다는 사실만으로 Technical 부분은 마치고자 한다. 더욱 더 이끌고 가기에는 시간의 여유가 없다. (본인은 황금 같은 주말+월요일은 이 X 같은 사건으로 인하여 엉망이 되어 버렸다.)

III. 결론

이 웜의 특징을 보자면 아주 짧은 코드와 SQL 서버 자체의 기능을 이용하여 공격의 효과를 배가 하였다고 판단 된다. SQL Server의 고유기능과 최적화된 공격코드를 이용한 아주 효과적인 공격 형태를 취하여 피해가 막심하였던 것으로 판단된다. 위의 기술적인 설명으로 정확하지는 않으나 개인적인 판단과 그 동안의 경험으로 미루어 보건대 충분히 이번 사태의 문제점에 대해서 설명이 되었다고 생각한다.

1월 27일 오후 3시 현재까지의 결론은 1434 UDP port 한 곳만으로 전파 및 무한 루프에 의한 리소스 소모가 일어난다고 판단한다. 1434 UDP port에 대해 들어오는 패킷에 따라 가변적으로 작동하는 SQL Server의 매커니즘으로 인하여 적지않은 혼동이 있었다. Packet의 첫번째 byte가 0x04로 시작할 경우에 NGSSoftware사의 분석 문서를 보게 되면

전송 문자열: \ 0x04 \ 0x41 \ 0x41 \ 0x41 \ 0x41 (0x41=A) 이런 패킷이 전송 되었을 때

Internet 붕괴의 기술적 원인 분석

==

첫번째 패킷이 0x04로 시작 되므로 Monitoring thread가 실행이 되며 이 thread는 레지스트리의 값을 읽는다고 한다.
HKLM \ Software \ Microsoft \ Microsoft SQL Server \ AAAA \ MSSQLSERVER \ CurrentVersion (등록된 SQL Server의 instance를 찾는 과정이라 예상됨)
이 레지스트리를 열도록 되어 있는 명령인데 AAAA 대신에 좀 더 긴 문자열을 삽입하게 되면 stack overflow가 발생하게 되고 이에 따라 Return address가 반환이 되는데 여기에서 공격코드를 실행할 수가 있다.
Ref: http://www.ngssoftware.com/advisories/mssql-udp.txt
참고 하시길.

그리고 첫번째 단일 패킷이 0x0A일 경우에만 instance에 대해 keep alive 로직을 따라 반응을 보이는 것으로 되어 있다.
결국 SQL Server 자체가 UDP port 1434번에 대해 두 가지 형태의 옵션을 패킷 형태로 실행함으로 인해 하나의 포트를 통해 두 가지 형태의 공격이 동시에 발생하고 부수적으로 다른 Traffic에 대한 공격도 일어남으로써 효과가 극대화 되었다고 할 수 있다. 위의 그림에는 언급되어 있지 않지만 다른 IP의 1434 port로 연결 문자열을 전송 하였을 때 1434번 포트가 열려져 있지 않을 경우에는 또 ICMP Unreachable 과 같은 다양한 형태의 에러 메시지가 전송 시스템으로 되돌아 오는 효과도 있을 수 있다. 모든 가능성에 대해 자료가 보강되는 대로 문서를 계속 갱신하게 될 것이다.

당장 1월 27일 오전이 문제가 될 수 있는데 각 기업 망의 SQL 서버가 원격에 있는 Instance를 이용할 경우 연결이 되지 않는 데이터의 문제가 발생할 수 있다.
또한 백신회사에서 개발된 백신제품을 보건대 정상적인 패킷을 차단하는 부분은 앞으로 SQL Server의 운용에 많은 문제가 될 수 있음을 이야기 하며 해결책은 패치 이외에는 없음을 명확하게 하고자 한다.
백신을 만들지 못하는 이유는 SQL Server의 기능을 변경해야 하는데 SQL Server가 Open source도 아니지 않는가? 정상적인 SQL Server의 운영에 지장이 있을 수 있으므로 염두에 두기 바란다. 또한 일시적인 방편으로 라우터와 방화벽에서 1433, 1434 포트에 대해 모두 Block을 했을 것이다. 이 문제를 해결하기 위해서는 오직 만든 회사의 패치만이 대안임을 명확하게 하고자 한다. 당장 Instance 경우는 Instance의 포트를 변경함으로써 정상적으로 이루어 질 수 있지만 이런 Instance를 찾아내는 것이 힘든 과정이 될 것이므로 기업의 서버 관리자들 및 데이터베이스 관리자 들은 명심해야 할 것이다.

제시한 문제점과 지적이 틀렸을 경우엔 지체하지 말고 연락 주시기 바란다. 그래야 명랑 사회가 유지 될 수 있으니 말이다. 추가로 그림 그리느라 시간이 걸렸음을 양해하시길..

Internet 붕괴의 기술적 원인 분석

===

즐겁고 신나는 한해의 액땜으로 생각하고 문제점을 정비하길 기원해 본다.

연락처: winsnort@hotmail.com (MSN)
Email: winsnort@securityindepth.net , winsnort@skinfosec.co.kr

---

소심하고 안정적인 상태의 성공을 바라기 보다는
거대하고 야심찬 꿈의 실현에서의 실패를
나는 바라노라. - 절망의 바다에서
