안랩 온라인 보안 매거진

월간 安

2013 여름호 (Jul. & Aug.)

다차원 분석 플랫폼

CONTENTS

다차원 분석 플랫폼, 악성코드 대응의 새 판을 짜다

악성코드가 나날이 고도화, 다양화되면서 사후 대처 위주의 대응 방식은 한계를 맞고 있다. 제로데이 공격을 비롯해 최근 화두가 되고 있는 APT 공격에 신·변종 악성코드를 이용한 사례가 증가하면서 기존과 같은 시그니처 기반의 악성코드 탐지만으로는 더 이상 만족할 만한 수준의 위협 대응을 담보하지 못하는 실정이다.

이에 안랩은 최근 독자적인 다차원 분석 플랫폼 구축을 완료했으며, V3 제품군을 필두로 APT 대응 솔루션 트러스와처(AhnLab TrusWatcher), 네트워크 보안 솔루션 트러스가드(AhnLab TrusGuard) 시리즈까지 거의 모든 제품에 순차적으로 적용할 예정이다. 이를 통해 사전 방역을 근간으로 한 혁신적인 대응 체계를 구축한다는 전략이다. 안랩의 새로운 위협 대응 프레임워크이자 악성코드 대응 기술인 다차원 분석 기술을 미리 살펴본다.

매일같이 엄청난 수의 악성코드가 제작·유포되고 있다는 것은 더 이상 새로운 사실이 아니다. 안랩 시큐리티대응센터(ASEC)가 집계한 바에 따르면, 지난 4월 감염이 보고된 악성코드는 모두 550만 8895건이었다. 이보다 앞선 지난 3월에는 약 760만 건에 달하는 악성코드 감염이 보고됐다. 국내만 해도 매월 평균 500~700만 건에 달하는 악성코드 감염이 발생하고 있는 것이다.

이 뿐만 아니라, 전세계적으로 알려지지 않은(unknown) 신·변종 악성코드의 유포 속도는 더욱 가속화되고 있다. 또한 안티바이러스의 탐지를 우회하는 악성코드 등 동작 방식도 훨씬 치밀화, 고도화되고 있다.

이와 함께 최근에는 지능형 지속 보안 위협 APT(Advanced Persistent Threat)와 같이 타깃 기업이나 기관, 혹은 특정 지역에서 주로 사용하는 프로그램의 취약점을 찾아내 악용하는 '맞춤형 악성코드'까지 등장하고 있어 악성코드 대응은 '백사장에서 바늘을 찾는 일'처럼 어려워지고만 있는 추세다. 이 때문에 시그니처 기반의 전통적인 대응 방식의 한계론도 등장했다.

다차원적인 분석이 해법이다

이 같은 이유로 많은 보안 업체들은 기존의 시그니처 방식이 아닌 각기 다른 방향에서의 악성코드 대응 방식을 도입하고 있다. 안랩 또한 다각도에서 위협을 분석하고 입체적으로 대응할 수 있는 방안을 모색한 결과, 올해 '다차원 분석 플랫폼'을 구현하고 자사 대부분의 제품에 적용을 진행하고 있다.

다차원 분석은 행위 기반, 평판 기반 등 다양한 분석 기술을 복합적으로 적용함으로써 위협의 유입 단계부터 사전적인 대응을 가능케 하는 신개념 종합 위협 대응 기술이다.

실시간 위협 대응은 물론, 보다 능동적인 위협 대응이 가능한 다차원 분석은 총 6개의 주요 탐지 및 대응 기술로 구성되어 있다. ▲URL/ IP 탐지 기술 ▲클라우드 기반 탐지 ▲시그니처 기반 탐지 ▲평판 기반 탐지 ▲행위 기반 탐지 ▲액티브 디펜스(Active Defense)가 그것이다.

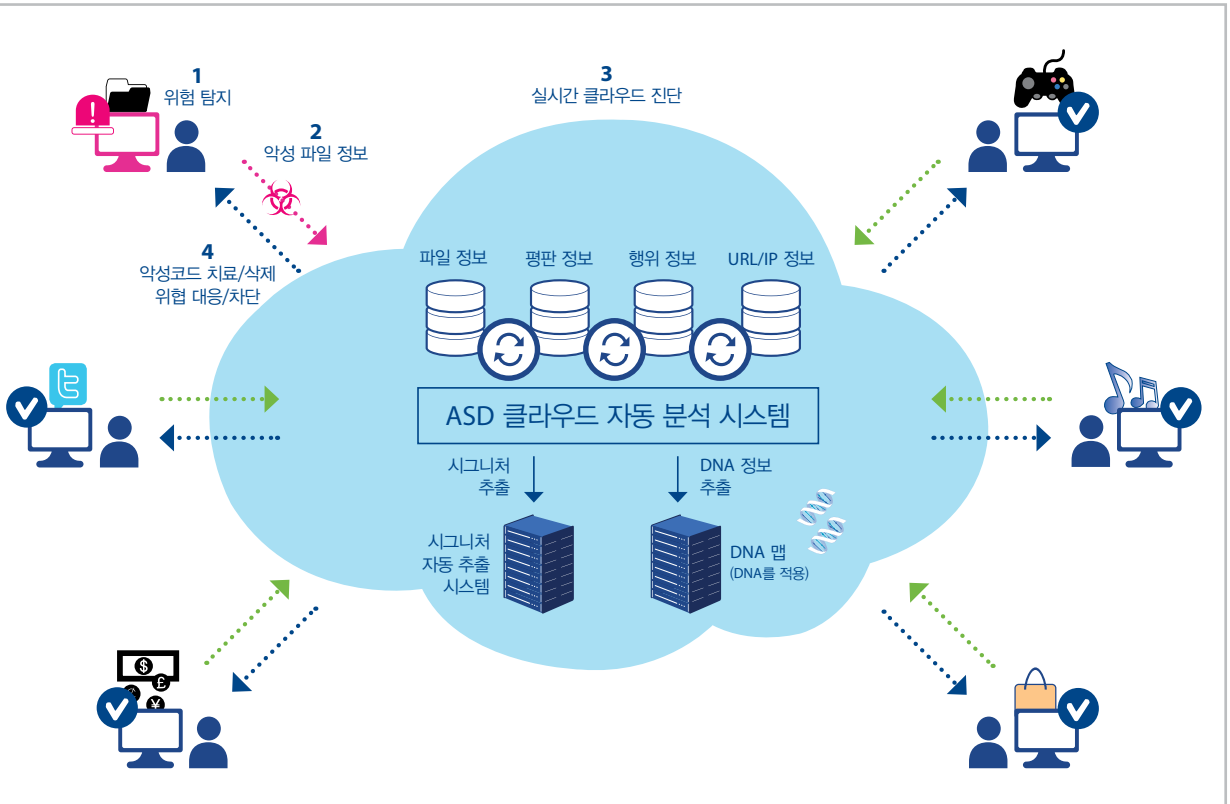
[그림 1] 다차원 분석 플랫폼 개념도

1. URL/IP 기반 탐지 – 악성 웹사이트 접속 차단, C&C 서버 등 외부와의 연결 차단

최근 대부분의 악성코드는 인터넷을 통해 네트워크 내부로 유입되는 양상을 보이고 있다. 감쪽같이 정상적인 것으로 위장해 악성코드를 유포하는 교묘한 웹 사이트들이 기승을 부리고 있을 뿐만 아니라 웹 사이트에 접속만해도 악성코드에 감염되는 사례도 빈번하게 발생하고 있다.

이 때문에 악성코드의 네트워크 유입을 방지하기 위해서는 악성 파일이 유포된 바 있는, 또는 다수의 경험을 통해 위험하다고 판단되는 URL에 대해서는 사전에 차단할 필요가 있다. 이것이 바로 URL 및 IP 기반 탐지 기술의 핵심이다. URL/IP 기반 탐지 기술은 악성코드를 유입 단계부터 막는, 그야말로 원천 차단하기 위한 기술로 사전 방역의 관점에서 매우 중요한 가치를 지닌다.

특히 C&C 서버와의 통신 등 악의적인 외부 네트워크와의 연결도 탐지 혹은 차단해 중요 정보 탈취나 공격자의 명령에 의한 추가적인 공격을 방지할 수 있다. 별도의 시그니처 없이 실시간으로 위험한 URL 또는 IP를 탐지하고 대응할 수 있다는 점도 최근의 위협 대응에 유용한 기술로 인정받고 있다.



[그림 2] ASD 네트워크 개념도

2. 클라우드 기반 탐지 – 실시간 위협 대응, 사전 방역

클라우드 기반 탐지 기술은 안랩의 클라우드 컴퓨팅 기반의 안랩 스마트 디펜스(AhnLab Smart Defense, 이하 ASD) 기술을 이용해 실시간으로 파일의 악성 여부를 다각도로 분석, 진단하는 기술이다. 대규모 파일 DB를 중앙서버에서 관리하고 PC(또는 시스템)에 설치되어 있는 ASD 엔진에서 파일의 악성여부에 대해 문의하면 이에 대해 응답을 해주는 방식이다. 시그니처 엔진이 업데이트되기도 전에 위협 대응이 가능한, 사전 대응 관점의 진단 기술로 특히 신•변종 악성코드 진단에 효과적이다.

안랩의 클라우드 기반 탐지 기술은 2000만 명 이상으로 구성된 ASD 네트워크를 통해 실제 발생 또는 유포되고 있는 위협, 즉 샘플을 보다 신속하게 수집해 위협이 발생하는 시점에 실시간으로 대응이 가능하다. ASD 네트워크를 통해 수집된 샘플에 대한 즉각적인 분석 후 V3를 비롯한 안랩의 전 제품에 해당 위협 정보를 공유함으로써 악성코드 확산 방지에 기여한다.

3. 시그니처 기반 탐지 – 오탐 최소화, 진단 속도 개선 및 엔진 경량화

안랩의 다차원 분석에 적용된 시그니처 기반 탐지 기술 역시 혁신적인 변화를 거듭해 기존의 시그니처 방식과는 상당한 차이를 보인다.

우선, 안랩의 시그니처 기반 탐지의 중심에는 7억여 개의 ASD DB가 있다. ASD DB에는 악성 파일에 대한 정보뿐만 아니라 정상 파일에 대한 정보도 축적되어 있어 보다 신속하고 효율적인 진단이 가능하며 오탐이 거의 없는 정확한 진단이 가능하다.

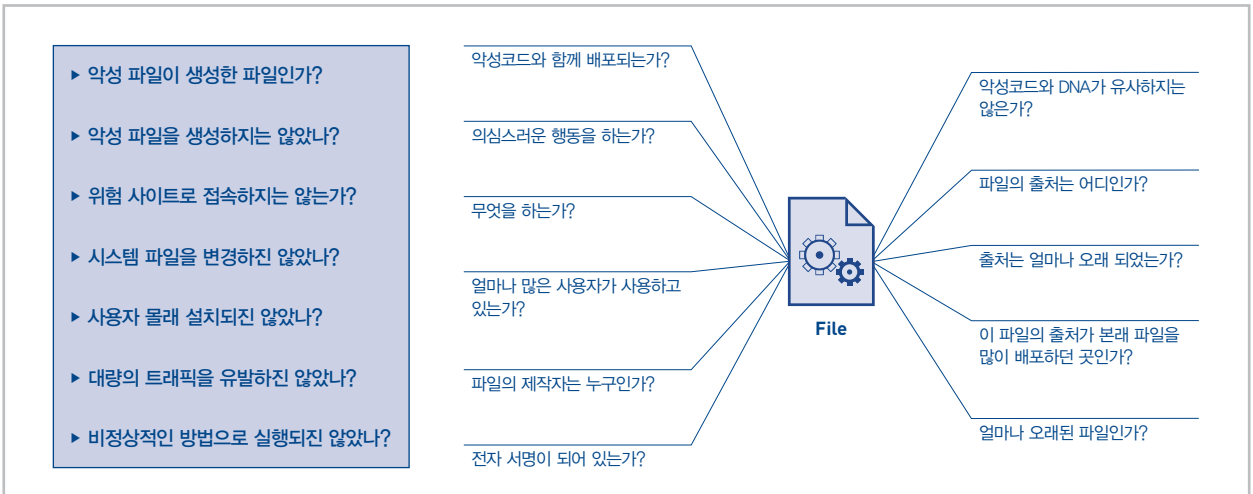
또한 안랩이 독자 개발한 DNA 스캔(Scan) 기술을 적용해 시그니처 기반 탐지에 혁신을 더했다. DNA 스캔은 한 마디로 '악성 프로그램의 DNA 정보를 이용해 탐지하는 기술'이라 할 수 있다. 악성코드가 갖고 있는 고유한 특성, 즉 파일의 DNA를 추출해 악성코드 진단에 이용하는 것으로, 기존의 파일 시그니처 대신 파일의 특성을 기반으로 진단하는 방식이다. 이는 샘플 수집 후 분석 및 대응하는 방식인 즉 사후 대응 방식과는 확연한 차이를 보인다. 시그니처가 적용되어 있지 않더라도 사전 방역이 가능해서다. 수백여 개의 DNA 룰을 기반으로 다양한 변종 악성코드에 대한 진단까지 가능해 사전 방역 효과를 제공하며 오탐 및 미탐의 가능성이 최소화된다.

다종다양한 악성코드의 시그니처를 끊임없이 축적해 대응하는 방식은 현실에 부합하는 효과적인 대응책이라 하기 어렵다. 엔진에 적용되는 시그니처 DB가 늘어날수록 엔진의 크기도 늘어나야 하기 때문이다. 반면에 방대한 샘플 DB와 분석 노하우를 통해 압축적으로 재편한 DNA 룰을 적용하면 시그니처를 끊임없이 축적하는 기존 방식에 비해 백신의 엔진 크기를 혁신적으로 줄일 수 있어 사전 방역과 사용자 편의성이라는 1석 2조의 효과가 있다.

4. 평판 기반 탐지 – 우회 공격 및 잠재적인 위협 대응

안랩의 다차원 분석 플랫폼의 핵심은 평판 기반 탐지와 행위 기반 탐지라 할 수 있다. 앞서 살펴본 클라우드 기반 기술과 시그니처 기반 기술이 기존의 기술을 혁신적으로 개선한 것이라면, 평판 기반 기술과 행위 기반 기술의 적용은 악성코드 대응의 패러다임 자체를 혁신한 것으로 평가할 만하다.

평판 기반 탐지 기술은 안전 여부가 확인되지 않은(unknown), 이른바 '미진단' 파일 및 프로그램에 대한 잠재적인 위험성을 사전 차단하는 기술이다. 이 기술은 최근의 고도화된 공격 기법 대응에 유용한데, 바로 백신을 우회하는 악성코드 탐지다. 최근 악성코드 제작자들이 악성코드를 유포하기 전 백신의 탐지 여부를 테스트 해보는 경우가 늘고 있다. 이 역시 시그니처 기반의 백신이 갖는 한계로 지적되기도 한다. 이와 관련해 안랩의 다차원 분석 기술은 사용자 수와 평가, 출처, 다른 파일들과의 관계 등 다양한 평판 요소를 진단 기준으로 추가함으로써 시그니처 기반 백신의 한계를 넘어 선제적 대응을 구현하고 있다.



[그림 3] 평판 정보의 예

이 기술 역시 신•변종 악성코드와 같이 악성 여부가 결론 내려진 경우가 아닐 때 유용하다. 클라우드를 통해 수집/분석된 다양한 평판 정보를 바탕으로 다른 사용자들의 평가를 참조해 사용자가 실제적으로 판단하고 대응할 수 있도록 한다.

미확인 파일(혹은 프로그램)임에도 불구하고 다수의 사용자들이 악성 또는 정상으로 분류한 정보나 실제 사용자 수 등에 대한 정보를 제공함으

로써 사용자의 환경 등 여러 요소를 고려하여 차단 또는 허용 등을 판단할 수 있다. 다수의 사용자 '평가'를 적용함으로써 사용자 측면에서의 실질적인 보안 수준 설정과 대응이 가능하다는 의미다.

5. 행위 기반 탐지 – 제로데이 공격, 익스플로이트 대응

평판 기반 탐지 못지 않게 유용한 기술이 행위 기반 탐지 기술이다. 행위 기반 탐지 기술은 파일의 행위가 악성인지의 여부를 진단하는 기술로, 탐지를 우회하는 고도화된 악성코드를 비롯해 제로데이 취약점을 이용한 악성코드 대응에 효과적이다. 앞서 설명한 평판 기반 기술과 마찬가지로 안랩의 축적된 노하우와 인프라를 기반으로 구현된 이 기술은 악성 파일이 궁극적으로 수행하는 행위를 100여 개의 패턴으로 상세하게 구분해 진단의 근거로 삼는다. 예를 들어 사용자의 동의 없이 또 다른 악성코드를 다운로드 및 실행하는 등의 행위를 하는 것만으로도 악성 여부를 진단하고 대응할 수 있다. 이로써 제로데이 공격은 물론, 비정상적인 익스플로이트(exploit)를 원천 차단할 수 있다.

6. 액티브 디펜스(Active Defense) – 상세 분석 보고서, 능동적인 대응 구현

클라우드 기반, 시그니처 기반, 평판 기반, 행위 기반 기술의 결합으로도 현재의 악성코드는 대부분 대응이 가능하다. 그러나 악성코드 및 공격 기법은 진화하는 속도도 상상을 초월하며 그 방향 또한 예측하기가 쉽지 않다. 고도의 분석 기술의 결합마저도 우회하는 위협의 발생 가능성을 완전히 배제할 수는 없다는 의미다.

이 같은 만약의 경우마저 대비하기 위한 기술이 바로 액티브 디펜스(Active Defense)이다. 안랩의 독자 기술로 개발된 액티브 디펜스는 시스템 내부의 위협에 대한 가시성을 제공하는 기술이자 기능이다. 현재 시스템 내에서 이상 행위를 보이는 파일이 있다면 그 파일이 구체적으로 어떤 행위를 하는지 등에 대해 실시간으로 상세한 분석 정보를 제공함으로써 가시성을 확보함과 동시에 사용자의 선제적인 대응이 가능하다. 사용자는 각종 분석 정보를 활용해 악성코드 삭제 등 위협에 대해 보다 능동적으로 대처하고 보안 수준을 향상시킬 수 있다.

다차원 분석 기술의 기대 효과

이처럼 다양한 고도의 기술이 유기적으로 결합된 안랩의 다차원 분석 기술은 지난달 출시된 개인용 백신 V3 Lite, V3 365 클리닉에 적용됐으며 곧 출시될 'V3 인터넷 시큐리티 9.0(AhnLab V3 Internet Security 9.0, 이하 V3 IS 9.0)'을 비롯한 V3 제품군을 필두로 트러스가드 등 네트워크 제품군, 모바일 제품군, APT 대응 솔루션 트러스와처나 산업 보안 솔루션 트러스라인 등 대부분의 안랩 제품군과 서비스에 모두 적용될 예정이다.

[그림 4] 다차원 분석 기술이 최초 적용된 V3 IS 9.0의 화면

그렇다면 이 같은 다차원 분석 기술이 적용된 제품을 통해 기대되는 효과는 무엇일까.

결론부터 말하자면, 당연히 위협에 대한 정확한 진단이 가능하다. 미탐·오탐을 최소화하고 행위 기반 및 평판 기반의 탐지를 통해서 신·변종 악성코드까지 진단함으로써 보다 정교한 진단이 가능한 것이다.

특히 사전 방역 관점에서의 보안을 구현하고 있다는 점은 주목할 만하다. 특정 기업/기관을 노린 타깃 공격의 증가와 더불어 맞춤형 악성코드가 속속 등장하고 있는 현 상황을 고려할 때, 사후 대응을 넘어 사전적인 대응을 위한 기술과 기능을 제공함으로써 다양한 보안 위협에 대해 효과적인 대처가 가능할 것으로 기대된다. 이것은 시그니처 기반 기술의 한계를 뛰어넘었다는 점에서도 의미가 크다 할 것이다.

복잡다단한 오늘날의 위협 환경에선 다양한 기술의 융합이 필수불가결하다. 그러나 이를 유기적으로 엮어내 실질적인 효과를 내는 것은 결코 쉬운 일이 아니다. 바로 이 부분이 안랩의 기술과 노하우, 인프라의 가치가 드러나는 지점이다. 안랩은 20여 년간 축적해온 기술 노하우와 인프라를 바탕으로 각각의 고도화된 분석 기술이 동시에 시너지 효과를 낼 수 있도록 다차원 분석 플랫폼을 구현해냈다.

김정훈 안랩 ASD실 실장은 "다양한 분석 기술들은 모두 장단이 있고, 이 때문에 하나의 분석 기술로 악성코드를 모두 가려낸다는 게 쉬운 일이 아니다"라며 "다차원 분석 플랫폼의 경우, 각 기술의 장점은 취하고 단점은 다른 기술로 보완함으로써 기술의 완성도와 진단의 정확성을 높였다"고 말했다.

다차원 분석 플랫폼을 기반으로 한 안랩의 폭넓은 제품 포트폴리오를 통해 기존과는 차원이 다른 보안 수준을 구현할 것으로 기대되는 이유다. 여러 인증 기관의 비공식 테스트에서 다차원 분석 플랫폼 기반의 새로운 V3 IS 9.0 제품이 뛰어난 진단율을 보인 것은 물론, 엔진의 무게감과 검사 속도 또한 체감할 수 있을 정도로 개선됐다는 것도 김정훈 실장의 전언이다.

6.25 사이버테러 분석 보고서 1부_DDoS 공격 분석

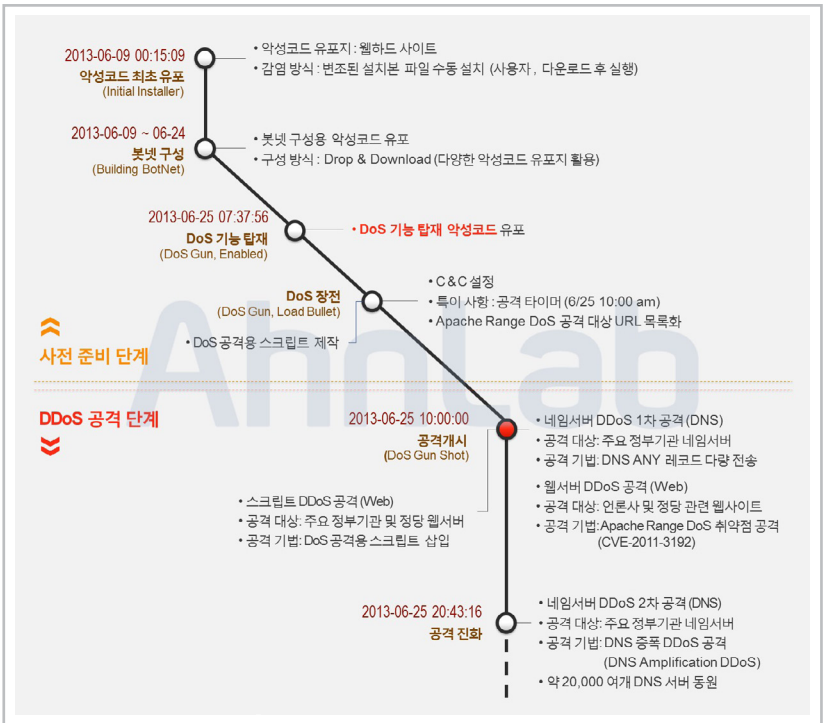
다시 돌아온 DDoS, 더욱 고도화된 양상으로 진화

2013년 6월 25일 오전 10시, 주요 정부 기관 및 언론사에 DDoS(Distributed Denial of Service, 분산 서비스 거부) 공격이 발생했다. 이번 DDoS 공격은 악성코드와 좀비 PC를 이용한 전통적인 DDoS 공격 방식 외에 일부 타깃 웹사이트에 대한 공격에는 웹사이트 접속만으로 DDoS 공격을 발생시키는 악성 스크립트 방식이 이용돼 주목을 끌었다. 이 같은 방식이 국가적 대형 DDoS 공격에 이용된 것은 처음이다. 또한 특정 언론사, 특정 정당의 시도당 및 관련 도메인에 대한 아파치 레인지(Apache Range) DoS 취약점을 이용한 공격도 나타났다.

다음은 이른바 '6.25 사이버테러' 관련 DDoS 공격 유발 악성코드 및 악성 스크립트를 이용한 공격 기법에 대한 안랩 시큐리티대응센터(ASEC)의 상세 분석 보고서를 발췌한 내용이다.

[그림 1]은 6.25 사이버테러에서 나타난 DDoS 공격 양상을 타임라인 별로 정리한 내용이다. 오전 10시부터 주요 정부 기관을 대상으로 벌어진 이번 DDoS 공격의 방식은 크게 두 가지로 구분된다. ▲좀비 PC를 통한 DDoS 공격 방식과 ▲악성 스크립트를 이용한 DDoS 공격 방식이 그것이다.

우선, 악성코드 유포를 통한 좀비 PC를 이용한 DDoS 공격 방식은 지난 2011년 3.4 DDoS 공격 당시와 같이 웹하드를 통해 DDoS 공격 관련 악성코드가 배포된 것으로 확인됐다. 공격자는 특정 웹하드의 설치 파일과 업데이트 파일을 통해 개인 사용자 PC를 악성코드에 감염시켜 좀비 PC로 만들었다. 이후 C&C 서버를 통해 6월 25일 오전 10시에 좀비 PC들이 특정 DNS(Domain Name Service) 서버에 DDoS 공격을 수행하도록 명령을 내린 것으로 확인됐다. DNS 서버는 웹 사이트 이용자들이 정부 기관의 주소를 입력하면 이를 실제 웹사이트로 연결시켜주는 기능을 하는데, 이 DNS 서버가 공격을 받아 일부 정부기관 웹사이트들의 접속이 원활하지 못했던 것이다.



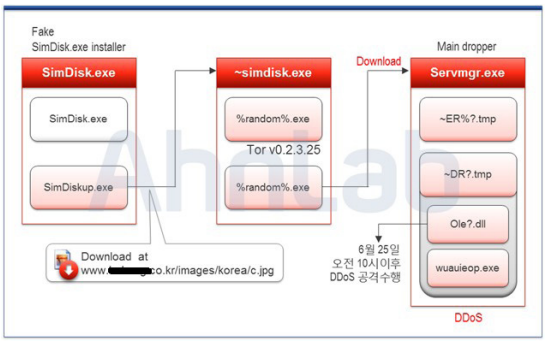
[그림 1] 6.25 DDoS 공격 주요 타임라인

한편 안랩이 최초로 확인한 '악성 스크립트를 이용한 DDoS 공격 방식'은 좀비 PC를 이용한 공격과 달리, 공격자가 특정 웹사이트에 악성 스크립트를 삽입해 놓음으로써 사용자들이 해당 사이트를 방문하면 미리 설정해 놓은 웹사이트로 공격 트래픽을 발생시키는 방식이다. 사용자가 악성 스크립트가 설치된 해당 웹사이트에 접속(방문)하고 있는 동안에 사용자의 PC에서 공격 트래픽이 발생한다.

6.25 DDoS 공격 유발 악성코드 분석

1. 주요 파일 분석 정보

특정 웹하드 업체의 설치파일(RARSFX)을 변조하여 해당 파일이 압축 해제되면서 SimDiskup.exe가 실행되도록 설정돼 있다. 다음은 6.25 DDoS 공격 방식 중 악성코드를 기반으로 한 공격에 이용된 파일의 상세 분석 내용이다.



[그림 2] 파일 관계 및 공격 시나리오

(1) servmgr.exe - 드롭퍼

드롭퍼인 servmgr.exe는 [그림 3]과 같이 리소스 영역에 4개의 PE 파일을 갖고 있다. servmgr.exe는 실행되면 다음과 같이 동작한다.

① 시스템 버전 정보 체크

② OpenFileMappingA를 호출해 감염 여부 확인

```
MappingName = "Global\MicrosoftUpgradeObject9.6.4"
```

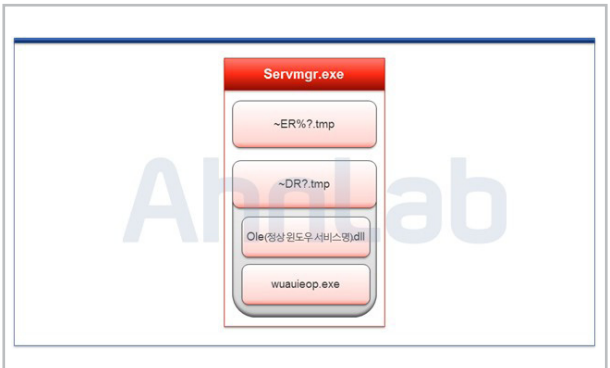
③ 시스템 버전에 따라 동작

servmgr.exe는 [표 1]과 같이 타깃 시스템 환경에 따라 기능은 동일하지만 시스템 환경에 맞는 각기 다른 파일을 생성한다는 점이 특징적이다. 또한 시스템 버전 정보가 6.0(Vista) 이상일 경우, GetVersionExA로 UAC 우회를 위한 파일을 추가로 생성한다.

시스템 환경	주요 생성 파일 및 동작 내용
32bit	- %Temp%\~DR(숫자).tmp 생성 후 별도의 인자값 없이 LoadLibaray 호출로 자기 자신을 로드한다. - 이후 'Ole(정상 윈도우 서비스명).dll' 파일을 생성하고 서비스에 등록한다. 파일명은 Ole로 시작되며 감염된 시스템의 윈도우 서비스명을 가져와 파일명을 만든다. - 'Ole(정상 윈도우 서비스명).dll' 파일이 특정 파일을 내려 받고, 내려 받은 파일에서 조건이 확인되면 DDoS 공격을 수행하는 파일을 생성 및 실행하는 등의 주요 기능을 수행한다.
64bit	- 다음과 같은 2개의 파일을 생성한 후, CreateProcessA로 실행한다. %Temp%\~ER6.tmp (64bit UAC 우회를 위한 파일) %Temp%\~DR7.tmp (32bit에서 로드 한 파일과 동일한 기능하는 파일로 Ole(정상 윈도우 서비스명).dll 파일을 서비스에 등록 해둔다.)

[표 1] 동일한 기능을 위해 시스템 환경에 따라 생성되는 파일

④ 배치 파일 %Temp%\ud.bat 생성 후 자기 자신 삭제



[그림 3] servmgr.exe의 리소스 섹션에 존재하는 PE 파일

(2) ole(정상 윈도우 서비스명).dll

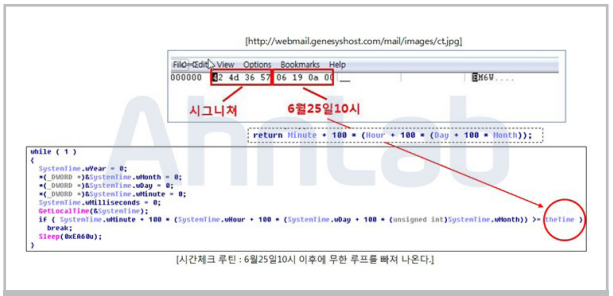
① 해당 악성코드는 'system32\ole(정상 윈도우 서비스명).dll' 경로에 저장된다. 단, 감염될 때마다 파일명이 달라지는데 일정하게 ole라는 문자로 파일명이 시작되며, 감염된 시스템의 정상 윈도우 DLL 파일명을 조합해 파일명이 만들어지는 것으로 추정된다. 이 DLL 파일은 서비스로 동작하면서 조건이 일치하면 다음과 같은 경로에 파일을 생성하고 실행한다.

```
'%system32%\wuauieop.exe'
```

② 단, 위의 경로에 파일을 생성하고 실행하기 전 다음의 URL에 접속하여 파일을 다운로드한다.

```
http://webmail.genesyshost.com/mail/images/ct.jpg
http://www.hostmypic.net/pictures/e02947e8573918c1d887e04e2e0b1570.jpg
```

③ 다운로드한 파일을 %temp%\~MR숫자2자리.tmp로 저장한 후, 시그니처('BM6W')와 시간 정보(6월 25일 10:00)를 확인하여 조건에 부합하면 '%system32%\wuauieop.exe'라는 경로에 파일을 생성하고 실행한다.



[그림 4] DDoS 공격 수행을 위한 시간 정보 확인

(3) wuauieop.exe

DDoS 공격 관련 기능을 갖고 있는 파일은 wuauieop.exe이다. 위의 과정을 거쳐 %system% 폴더에 생성된 wuauieop.exe라는 이름의 파일이 실질적인 DDoS 공격을 수행한다. 파일 내부에 2개의 공격 대상 IP가 고정되어 있다. 다음과 같이 각각 쓰레드(thread)가 구동되어 무한 루프로 동작하여 공격이 이루어진다.

```
Thread Function(1) (=401110)
- 공격 대상 IP: "152.99.1.10"

Thread Function(2) (=4012A0)
- 공격 대상 IP: "152.99.200.6"
```

또한 2개의 쓰레드 루틴에서 랜덤하게 도메인 네임을 생성하는 다음과 같은 코드가 존재한다. 이를 통해 도메인 네임을 랜덤화하여 DDoS 공격 효과를 증대시킨 것으로 추정된다.

```
"%s.gcc.go.kr"
- %s로 된 부분은 랜덤하게 a~z까지의 ASCII로 채워진다.
```

(4) ~SimDisk.exe

~SimDisk.exe는 네트워크 접속 정보의 감시 및 추적, 분석을 까다롭

게 하기 위해 이용하는 tor 프로그램의 소스코드를 사용한 자체 빌드 버전의 설치본 파일이다. ~SimDisk.exe가 실행되면 다음과 같은 경로에 랜덤한 이름으로 파일을 생성한다.

C:\Documents and Settings\(사용자명)\Application Data\Identities\{e38632c0-f9a0-11de-b643-806d6172696f}

- cmd.exe: 아래의 conime.exe(tor.exe)를 구동하는 런처 프로그램
- config.ini: cmd.exe가 참조하는 구성 설정 파일(정상 파일)
- conime.exe: tor v0.2.3.25 소스코드를 자체 빌드한 버전
- V3에서 Win-Trojan/Agent.3233806이라는 진단명으로 기 진단된 파일

2. 악성코드 기반 DDoS 공격 트래픽 분석

이번 6.25 DDoS 공격 중 악성코드를 기반으로 한 공격 방식도 크게 2가지로 구분된다. ▲DNS Query 및 DNS 증폭 DDoS 공격과 ▲아파치 레인지(Apache Range) 취약점을 이용한 공격이 그것이다.

(1) DNS DDoS 트래픽 분석

■ DNS ANY Query DDoS 공격

DNS 서버에 대한 서비스 거부 공격을 수행하는 DNS DDoS 트래픽은 아래와 같은 2개의 DNS 서버를 공격했다. 주요 정부 기관에서 이들 도메인 네임 서버를 사용하고 있었기 때문에 이번 공격의 대상이 된 것으로 보인다.

<DNS ANY Query DDoS 공격 >
ns.gcc.go.kr(152.99.1.10)
ns2.gcc.go.kr(152.99.200.6)

다음은 DNS ANY Query DDoS 공격 트래픽의 주요 특징이다.

① 일반적으로 정상 트래픽의 경우 A Query인데 반해 모두 ANY Query를 전송했다.
gcc.go.kr의 주 네임 서버를 효과적으로 공격하기 위해 'ANY' 레코드 정보를 전송했다.

② 랜덤 호스트명을 이용한 DNS Query를 두 개의 NS(152.99.1.10, 152.99.200.6)로 동시에 요청한다.
랜덤 호스트명 사용 시, 캐시(Cached)된 정보의 재사용이 불가하기 때문에 네임 서버에 부하를 가할 수 있으므로 {랜덤}.gcc.go.kr을 쿼리로 사용했다.

③ DNS 서버 부하를 위해 데이터 크기를 1300~1400 byte로 조정한다. 일반적인 DNS Query는 소량의 트래픽이나, 공격 네임 서버로의 대역폭(Bandwidth) 소모를 유발하기 위해 약 1400 byte의 데이터를 사용했다.

한편, 수집된 패킷은 초당 약 712회의 DNS Query를 전송하는 것으로 확인됐다.

Traffic	Captured	Displayed	Displayed %	Marked	Marked %
Packets	71	71	100.000%	0	0.000%
Between first and last packet	0.100 sec				
Avg. packets/sec	712.392				
Avg. packet size	1364.155 bytes				
Bytes	96855	96855	100.000%	0	0.000%
Avg. bytes/sec	971813.239				
Avg. MBit/sec	7.775				

[그림 5] DNS ANY Query 패킷 요약

■ DNS 증폭 DDoS 공격

공격 대상 NS에 직접 ANY Query를 전송하는 방식과 달리, 인터넷 상의 NS(reflector)를 동원하는 DNS 증폭 DDoS 공격을 수행하는 악성코드가 발견됐다. 해당 악성코드는 6월 25일 20:43:16(GMT+9) 경에 제작된 것으로 파악되며, DNS 증폭 분산 서비스 공격을 목적으로 제작됐다.

DNS 증폭 DDoS 공격이란 공개적으로 접근 가능한 DNS 서버를 이용해 대량의 DNS 응답 패킷을 공격 대상에 전송하는 공격 형태이다.

<DNS 증폭 DDoS 공격 대상 서버>
ns.gcc.go.kr(152.99.1.10)
ns2.gcc.go.kr(152.99.200.6)

해당 악성코드에 감염된 다수의 PC에서 IP 스푸핑(Spoofing)을 이용해 DNS ANY 레코드 쿼리의 출발지 주소를 타깃 DNS 서버로 위장한다. 이후 해당 결과를 Open Resolver(Reflector)가 적용된 DNS 서버를 통해 타깃으로 유도하여, 대역폭 및 네임 서버 자원을 고갈시키는 등의 DNS 증폭 공격(DNS Amplification DDoS)을 수행한다. 이 공격에는 2만여 개의 DNS 서버가 동원되었으나, 해당 DNS 서버들의 일부는 Open DNS Resolver 서버가 아닌 것으로 확인됐다.

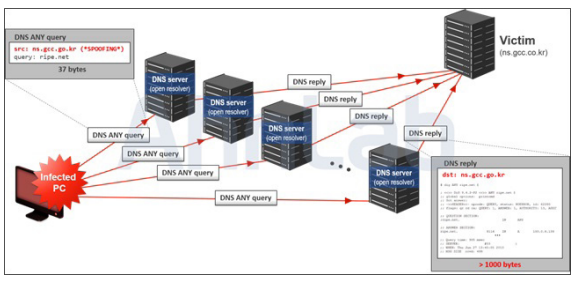
[그림 6]은 DNS 증폭 공격 과정으로, 그 내용은 다음과 같다.

① ripe.net에 대한 ANY 레코드 요청
Open DNS Resolver가 DNSSEC을 지원하는 ripe.net ANY 레코드를 요청해 큰 응답값을 수신한다.
ripe.net의 ANY Query의 결과는 1000byte 이상으로, 일반적인 요청의 수십 배에 달한다.

② 소스 IP 스푸핑
공격 대상 서버인 주요 정부기관 DNS 서버 IP로 위장한 DNS 쿼리 패킷을 발송한다.

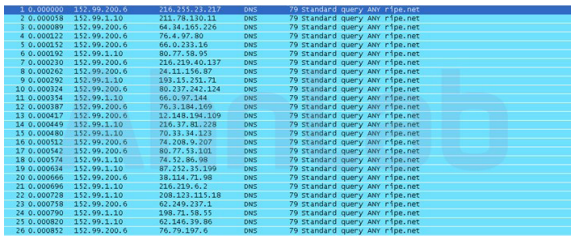
③ 패킷 분할(Fragmentation)
이더넷 1500byte를 넘는 큰 응답값은 패킷이 분할되는데, 응답받는 서버에서는 이를 재조립(reassemble)하는 과정에서 서버에 부하가 가중된다.

④ 추가적인 TCP 커넥션이 발생되어 부하가 가중된다.
응답이 512byte가 넘어서면 DNS에서는 Truncated flag 값을 설정하여 TCP 접속을 유도한다.
즉 ripe.net의 응답이 3000byte를 초과하면 Truncated flag를 설정하고 다시 TCP 접속을 통해 데이터를 수신한다.



[그림 6] DNS 증폭 공격 과정

DNS 증폭 DDoS 공격 패킷은 [그림 7]과 같이 DNS Query 출발지를 변조하는 등의 특징을 보인다.



[그림 7] 출발지가 변조된 DNS Query

또한 [그림 8]과 같이 ripe.net의 ANY 쿼리 결과를 사용한다. 일반적인 요청 결과에 비해 매우 큰 데이터 크기의 응답값으로, 증폭 공격의 효과를 극대화하기 위한 것이다.

[그림 8] ripe.net ANY 레코드 쿼리를 요청하도록 지정

[그림 9]는 해당 악성코드에 감염된 PC에서 수집한 패킷 통계로, 초당 약 4400개의 DNS ANY 쿼리가 전송되는 것으로 확인됐다 (단, 각 수치는 시스템의 사양에 따라 차이가 있을 수 있다).

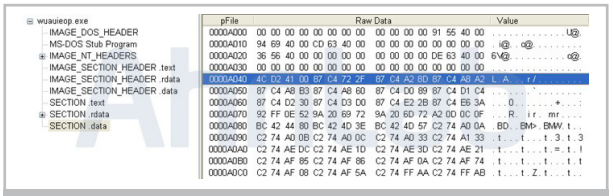
Traffic	Captured	Displayed	Marked
Packets	44167	44167	0
Between first and last packet	9.943 sec		
Avg. packets/sec	4442.145		
Avg. packet size	78.996 bytes		
Bytes	3489020		
Avg. bytes/sec	350912.026		
Avg. MBit/sec	2.807		

[그림 9] 공격 패킷 통계

ripe.net의 ANY 응답 결과를 전달할 때 사용하는 중계 DNS 서버(reflector) 목록은 악성코드의 '.data' 영역에 포함되어 있으며, [그림 10]의 VA값 0x40A044(FileOffset 0xA044)로부터 IP 목록을 가져온다.

[그림 10] 중계 DNS 서버 (reflector) 목록 획득 과정

[그림 11]은 FileOffset 0xA044 위치로, IP 목록이 저장돼 있다. 참조하기 위한 번호 EDX값을 구하기 위해 0x4C82로 나누는 점 등으로 미루어 공격에 사용하는 Resolver IP의 수는 1만 9586개로 추정할 수 있다.



[그림 11] 중계 DNS 서버(reflector) 정보(IP 주소)

(2) 아파치 레인지 취약점 공격

악성코드를 기반으로 한 6.25 DDoS 공격과 관련해 특정 정당 및 관련 도메인들에 대한 아파치 레인지 취약점 공격 시도가 파악됐다. 해당 악성코드는 외국인들을 대상으로 한 하숙/고시원 정보 제공 사이트(common/tpl/widget_layout.html)를 접속하고 C&C 서버로부터 공격 대상 URL을 전송받아 아파치 HTTPD 취약점을 공격하는 패킷을 전송한다.

<C&C 서버 정보>

mail.linkey.it:81

mail.pensys.com:32000

mail.intercity.com.my:80

이번 공격에 이용된 것은 아파치 HTTPD 취약점으로, 다수의 중복된 range 값을 헤더에 입력해 이를 처리하는 아파치 서버에 부하를 주는 방식이다. 이에 의해 메모리 자원이 고갈되고 CPU 점유율이 증가해 서비스 거부 상태에 이른다. 해당 취약점에 영향을 받은 아파치(Apache) 버전은 2.2.20 이전 버전으로 알려져 있으며, 1.3 버전에도 잠재적인 영향이 있을 수 있다.

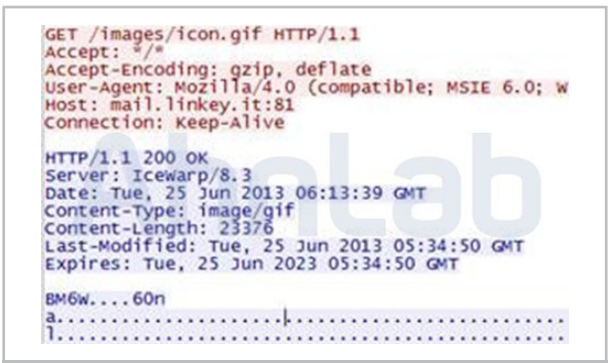
<아파치 HTTPD 취약점 개요>

- 취약점명: Apache Ranges DoS (CVE-2011-3192)
- 관련 사이트: http://httpd.apache.org/security/CVE-2011-3192.txt

공격 대상은 C&C 서버로부터 전송된다. C&C 서버의 응답은 다음과 같은 특징을 보인다.

<C&C 서버 응답 개요>

- 시그니처 : "BM6W"
- 4byte : 60na
- 공격 서버 + 대상 URI 정보
- 총 8개 서버 / 하위 URL 80개를 포함



[그림 12] C&C 서버로부터 공격 대상 URL 목록 수신

실제 공격 패킷은 [그림 13]과 같이 페이지를 요청(GET)할 때 중복되는 레인지(Range) 값을 대량으로 전송한다. 이로써 레인지 헤더에 6000 byte 이상의 값을 입력해 CPU 및 메모리 자원을 고갈시키며 서비스 거부 상태를 야기한다.

[그림 13] 페이지 요청(GET) 시 전송하는 중복된 레인지(Range) 값

악성 스크립트 기반의 DDoS 공격

6.25 DDoS 공격의 가장 큰 특징은 기존의 좀비 PC를 이용한 공격 외에 악성 스크립트를 이용한 공격 방식이 이용됐다는 점이다. 이 경우, 일반 사용자가 웹 브라우저를 통해 변조된 해당 웹사이트를 방문하는 것만으로 악성 트래픽이 발생된다.

1. 악성 스크립트 기반 DDoS 공격 시나리오

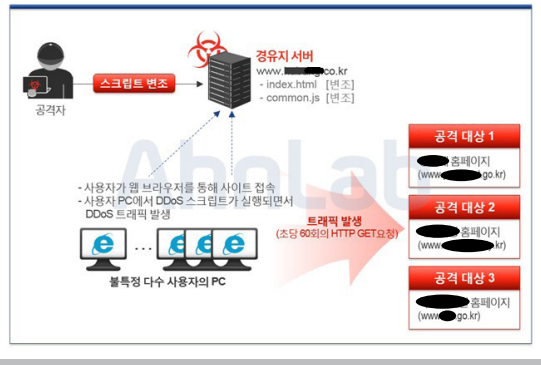
[그림 14]는 스크립트 기반 DDoS 공격의 진행 과정으로, 주요 내용은 다음과 같다.

① 공격자는 사전에 다수의 불특정 사용자들이 방문하는 경유지 서버를 해킹한다.

② 경유지 서버에서 사용하는 정상적인 페이지에 DDoS 공격 스크립트를 삽입한다.

③ 불특정 다수의 사용자들이 웹 브라우저를 통해 경유지 서버(변조된 웹사이트)를 방문하면, 사용자 PC 상에서 DDoS 공격 스크립트가 실행된다.

④ 스크립트에 설정된 공격 대상 서버에 다량의 HTTP GET을 요청해 악성 트래픽을 유발한다.



[그림 14] 스크립트 기반의 DDoS 공격 시나리오

악성 스크립트를 이용한 DDoS 공격 방식은 사용자가 해당 웹사이트를 방문한 상태일 때 트래픽이 발생하는 방식이다. 따라서 사용자가 접속한 브라우저를 닫거나 다른 사이트로 이동할 경우, 악성 스크립트 실행이 중단되기 때문에 브라우저를 통한 HTTP GET 요청도 함께 종료된다. 악성 스크립트를 기반의 DDoS 공격에 대한 피해를 최소화하기 위해 개인 사용자 및 기업 웹사이트 관리자는 다음과 같은 조치를 취할 수 있다.

〈스크립트 기반 DDoS 공격 조치 방안〉

1. 일반 사용자 조치 방안

경유지 사이트 접속(방문) 시 악성 스크립트가 실행되며 공격이 발생하므로,

– 해당 웹사이트(변조된 웹페이지)에 접속한 브라우저 창 종료, 또는 다른 사이트로 이동

– 백신 프로그램 최신 업데이트: 변조된 페이지나 악성 스크립트가 사용자 PC에서 실행되기 전에 백신을 통해 진단 가능(*V3 진단명: JS/Agent(엔진버전: 2013.06.27.01))

2. 웹사이트 관리자 조치 방안

웹 서버의 취약점을 이용해 악성 스크립트를 삽입하고 페이지를 변조하여 공격에 이용하므로,

– 웹 서버 취약점을 이용한 해킹 방지를 위해 서버 OS 및 응용 프로그램 최신 버전 적용

– 페이지 변조에 대한 모니터링 및 차단 강화

– 네트워크 상의 보안 시스템(IDS, IPS)을 통한 모니터링 및 차단 강화

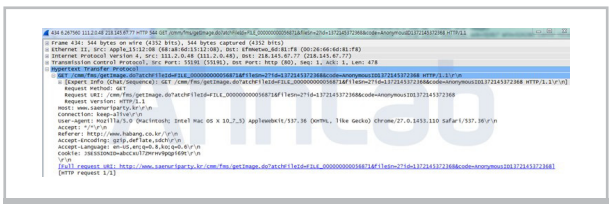
참고로, 안랩 V3 및 트러스가드(AhnLab TrusGuard)의 관련 진단명은 다음과 같다.

– V3 진단명: JS/Ddos

– 트러스가드 진단명: ddos_script_flood_0625(HTTP)

2. 악성 스크립트 기반 DDoS 공격 트래픽 분석

[그림 15]는 악성 스크립트 기반의 6.25 DDoS 공격과 관련된 HTTP GET 요청 패킷 정보다.


[그림 15] HTTP GET 요청 패킷 정보

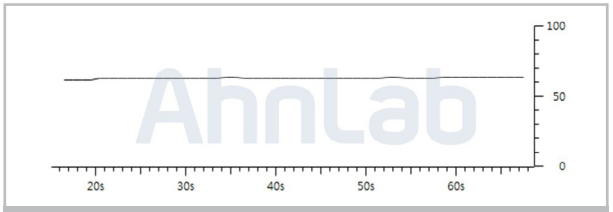
공격 스크립트는 [그림 16]과 같이 setInterval 주기가 '0'이 되도록 구현되어 있다. 단, HTTP GET 요청이 발생하는 주기는 브라우저마다 다소의 차이가 있을 수 있다.

```
var settingUrl = function(){
    var randomnumber=Math.floor(Math.random()*40);
    makeHttpRequest(urlList[randomnumber]);

var starting = setInterval(settingUrl, (2500 / 10000 | 0));
```

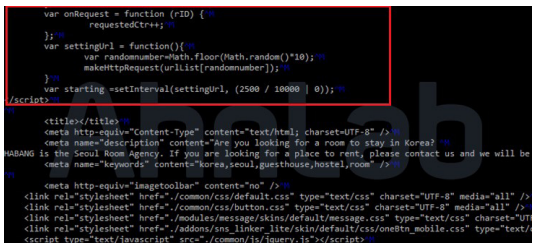
[그림 16] HTTP GET 요청 시도를 위한 스크립트 설정

분석 환경(IE 10)에서 재현한 결과, [그림 17]과 같이 대략 초당 60여 개 정도의 HTTP GET 요청을 시도하는 것으로 확인됐다. 한편, 구글 크롬 브라우저에서는 초당 40여 개 정도의 요청이 발생하는 것으로 분석됐다.



[그림 17] 초당 60회의 HTTP GET 요청

공격자는 다수의 사용자가 방문하는, 외국인을 대상으로 한 하숙/고시원 정보 제공 사이트 서버를 해킹해 공격의 경유지로 삼았다. 해당 웹사이트의 메인 페이지와 공통 자바 스크립트 파일에 각각의 타깃 공격용 스크립트를 삽입한 것이다. makeHTTPRequest 함수에 실제 공격 URL을 구성해 HTTP GET 요청 트래픽을 발생시킨다. 공격 대상인 3개 사이트의 하위 URL 총 60개를 대상으로 HTTP GET 요청을 반복적으로 수행한다.

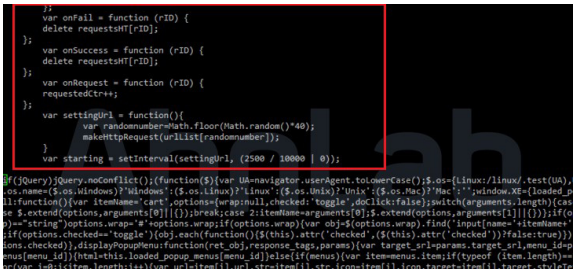


```
        var onRequest = function (rID) {
                requestedCtr++;
        };
        var settingUrl = function(){
                var randomnumber=Math.floor(Math.random()*10);
                makeHttpRequest(urlList[randomnumber]);
        }
        var starting =setInterval(settingUrl, (2500 / 10000 | 0));
</script>

        <title></title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <meta name="description" content="Are you looking for a room to stay in Korea?
HABANG is the Seoul Room Agency. If you are looking for a place to rent, please contact us and we will be
        <meta name="keywords" content="korea,seoul,guesthouse,hostel,room" />

        <meta http-equiv="imagetoolbar" content="no" />
    <link rel="stylesheet" href="./common/css/default.css" type="text/css" charset="UTF-8" media="all" />
    <link rel="stylesheet" href="./common/css/button.css" type="text/css" charset="UTF-8" media="all" />
    <link rel="stylesheet" href="./modules/message/skins/default/message.css" type="text/css" charset="UT
    <link rel="stylesheet" href="./addons/sns_linker_lite/skin/default/css/oneBtn_mobile.css" type="text/
    <script type="text/javascript" src="./common/js/jquery.js"></script>
```

[그림 18] 경유지 서버의 메인 페이지 변조 후



```
    };
    var onFail = function (rID) {
    delete requestsHT[rID];
};
    var onSuccess = function (rID) {
    delete requestsHT[rID];
};
    var onRequest = function (rID) {
    requestedCtr++;
};
    var settingUrl = function(){
            var randomnumber=Math.floor(Math.random()*40);
            makeHttpRequest(urlList[randomnumber]);
    }
    var starting = setInterval(settingUrl, (2500 / 10000 | 0));
```

[그림 19] 경유지의 자바 스크립트 파일 변조 후

유기적이고 고도화된 체계로 적극 대응

안랩은 공격 징후를 포착한 즉시 긴급 대응 체계를 가동해 DDoS 유발 악성코드를 분석하는 한편, 해당 악성코드 샘플과 유포지 정보를 관계기관에 공유했다. 또한 신속하게 전용 백신을 배포하고 상세 분석 보고서를 제공하는 등 피해 확산 방지 및 사전 대응에 나섰다.

6.25 DDoS 공격과 관련한 악성코드 및 악성 스크립트는 모두 V3로 진단이 가능하며, 공격 패킷에 대한 시그니처는 안랩 트러스가드(AhnLab TrusGuard)에 적용돼 있다. 이 밖에도 안랩 자체 테스트 결과, 안랩 트러스와처(AhnLab TrusWatcher)는 이번 DDoS 공격을 유발한 신종 악성코드를 행위 기반 분석 기술로 실시간 탐지했다.

6.25 사이버테러 분석 보고서 2부_하드디스크 파괴 공격 분석

하드디스크 파괴 악성코드, 기업 서버 노려

'6.25 사이버테러'에 DDoS와 더불어 하드디스크를 파괴하는 악성코드를 이용한 공격까지 나타나 관계자들을 긴장시켰다. 해당 악성코드에 감염되면 ▲파일 삭제 ▲사용자 PC 재부팅 시 하드디스크 파괴(MBR 삭제, 데이터 영역 삭제) ▲하드디스크 파괴 기능의 MBR 직접 삽입 등의 특징을 보인다. 이 밖에도 패스워드 및 바탕화면 변경 등의 증상이 나타난다. 이는 지난 3월 발생한 3.20 사이버테러 당시의 하드디스크 파괴와는 다른 양상이다.

다음은 안랩 시큐리티대응센터(ASEC)에서 분석한 6.25 사이버테러 관련 하드디스크 파괴 악성코드 분석 내용이다.

2013년 6월 25일 오전, 하드디스크 정보를 파괴하는 악성코드가 발견됐다. 6.25 사이버테러에서 발견된 하드디스크 파괴 유발 악성코드의 파일명은 다음과 같이 다양하나 그 기능은 동일하다. 이 글에서는 rdpshellex.exe를 중심으로 6.25 사이버테러의 하드디스크 파괴 공격을 살펴본다.

<6.25 사이버테러 하드디스크 파괴 악성코드명>
recdiscm.exe
taskhosts.exe
taskchg.exe
rdpshellex.exe
mobsynclm.exe
comon32.exe
diskpartmg.exe
dpnsvr32.exe
expandmn.exe
hwrcompsvc.exe

하드디스크 파괴 악성코드 동작 분석

6.25 사이버테러의 하드디스크 파괴를 유발한 rdpshellex.exe(245,760 바이트)는 다음과 같은 동작을 수행하는 것으로 확인됐다.

① 실행 조건 확인
해당 악성코드는 감염된 시스템에서 다음과 같이 윈도우 시스템 폴더에 lsass.exe 파일이 존재하는지 확인한다.

```
%System32%\icfg\lsass.exe
```

만약 감염된 시스템에 lsass.exe 파일이 존재하는 경우에는 바로 동작을 종료하고, 더 이상의 악의적인 기능을 수행하지 않는다. 또한 중복 실행을 방지하기 위해 다음의 Mutex를 확인한다.

```
Mutex = Microsoft-Windows-LDAP32-Client
```

② 서비스 등록
해당 악성코드는 서비스로 동작하며, 다음과 같은 정상 서비스 이름 중 하나를 사용한다.

- Removable Device Helper
- Windows Workstation Manager
- Windows Security Policy Service
- Windows Media Center Cloud Service
- Windows Fax Extension
- DCOM Event Filter
- Network Access Point Manager

③ 악성코드 인자별 기능
해당 악성코드는 다음과 같은 13개의 인자값을 갖고 있다.

```
r, i, p, m, b, z, d, f, n, w, s, t, a
```

인자값 –r 과 –I에 대한 분석 결과는 [그림 1]과 같으며, 그 외 각각의 인자값에 대해서는 추가 분석을 진행 중에 있다.

[그림 1] 특정 인자별 악성코드 기능

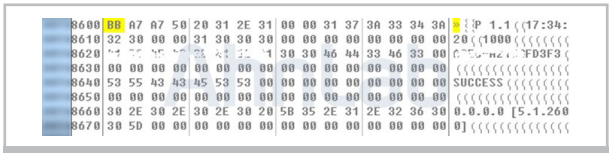
④ 감염 정보 전송

다음과 같은 IP의 시스템에 접속하여 감염된 시스템의 정보를 전송한다.

112.217.190.218: 8080
20.20.9.21:443
210.127.39.29:80

이때 전송되는 정보들은 다음과 같다.

- 시간 정보(hh:mm:ss)
- 컴퓨터 이름
- Message(EXIST, SUCCESS 등)
- OS 버전 정보(Major, Minor, Build No)



[그림 2] 감염 정보 전송

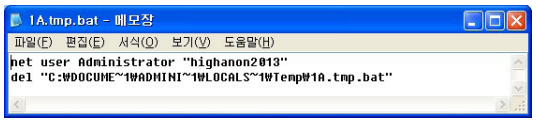
⑤ 특정 윈도우 서비스 강제 종료

동일한 %System32%\icfg\ 폴더의 i18n.nms 파일에서 0x1200 바이트 가량을 읽어 추가 작업을 수행하며, 윈도우 시스템에서 실행 중인 다음의 서비스를 강제 종료한다.

SENS(System Event Notification Service)
Alerter

⑥ 관리자 권한 설정 및 바탕화면 변경

사용자 계정의 임시 폴더(TEMP)에 [그림 3]과 같은 명령이 쓰여진 '[임의의 문자열]tmp.bat' 파일을 생성하고 해당 파일을 실행해 감염된 시스템의 관리자 계정(Administrator)의 암호를 변경한다.



[그림 3] 로그인 패스워드 변경 배치 파일 정보

RDPSHELLEX.exe의 리소스 영역에 갖고 있는 데이터를 이용해 desktop_image001.tmp라는 파일을 생성하고 해당 파일을 이용해 감염된 시스템의 바탕화면으로 변경한다. 이때 바탕화면에 교체되는 이미지는 [그림 4]와 같다.

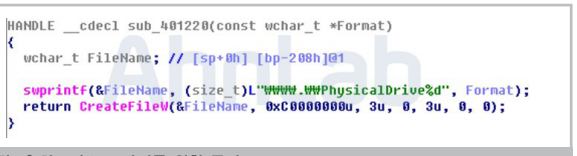
[그림 4] 변경되는 바탕화면 그림 파일

파괴 증상 분석

해당 악성코드는 하드디스크 파괴 및 특정 파일에 대한 삭제를 유발한다.

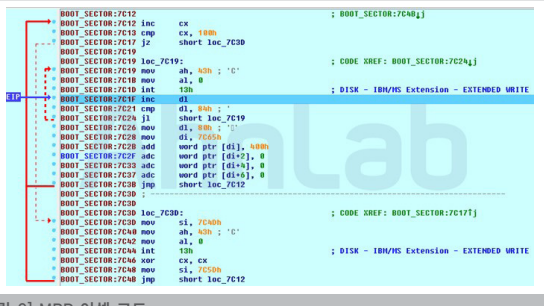
1. 하드디스크 파괴

해당 악성코드에 의해 감염(패치)된 MBR을 부팅하는 경우, MBR 코드가 수행되어 하드디스크가 파괴된다. 악성코드는 디스크 8개까지 MBR에 0x60 크기만큼을 감염(패치)시킨다. MBR 감염 방법은 PhysicalDrive를 연 후 WriteFile을 하는 방식이다.



[그림 5] 하드디스크 파괴를 위한 준비

감염(패치)된 MBR 코드의 주요 코드는 [그림 6]과 같다. INT 13H의 43 AH(Extended Write)를 이용해 디스크를 파괴한다는 것이 핵심이다.



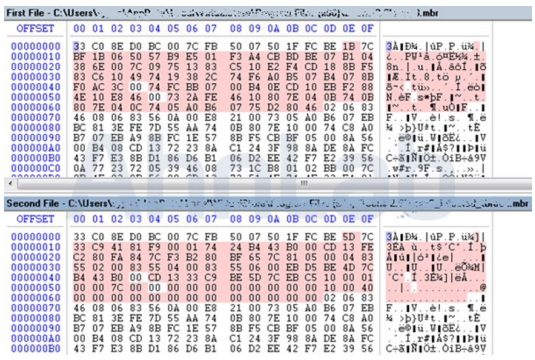
[그림 6] MBR 어셈 코드

위의 코드를 간단히 정리하면 다음과 같다.

```
Global_var_offset = 0
START :
For( cx=0; cx<100h; cx++)
{
    For( dl=80h; dl<84h; dl++)
```

```
    {
        write_memory_[0_40h sector]_to_HDD[dl] at
[(Global_var_offset+cx)*400h] sector
    }
}
Global_var_offset += (cx-1);
write_memory_[7C00_1sector]_to_HDD[80] at MBR
goto START:
```

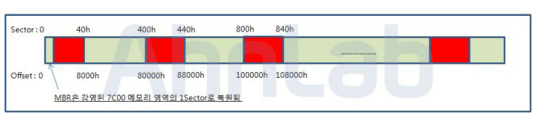
[그림 7]은 감염 전후의 MBR을 비교한 것으로, 코드가 변경된 것을 확인할 수 있다.



[그림 7] MBR 감염 전후 비교

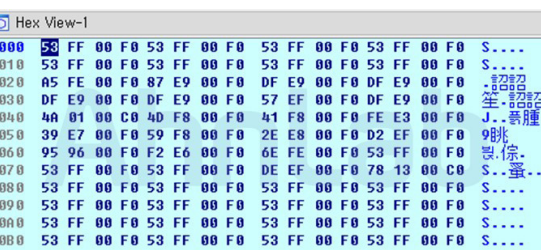
부팅 시, 첫 번째 루프에서는 하드디스크의 0번 섹터에 메모리 0번지에 위치한 데이터를 40h 섹터 크기(8000h)만큼 쓰는(write) 작업을 시작한다. 두 번째 루프에서는 400h 섹터의 크기만큼을 더한 하드디스크의 80000h 위치의 동일한 메모리 영역인 0번지의 데이터 40h 섹터 크기를 쓰는(write) 방식으로 구동된다. 코드상으로는 하드디스크 4개를 대상으로 진행하나, 편의상 한 개의 디스크로 가정하고 살펴본다.

디스크의 파괴 위치는 대략적으로 [그림 8]과 같다.



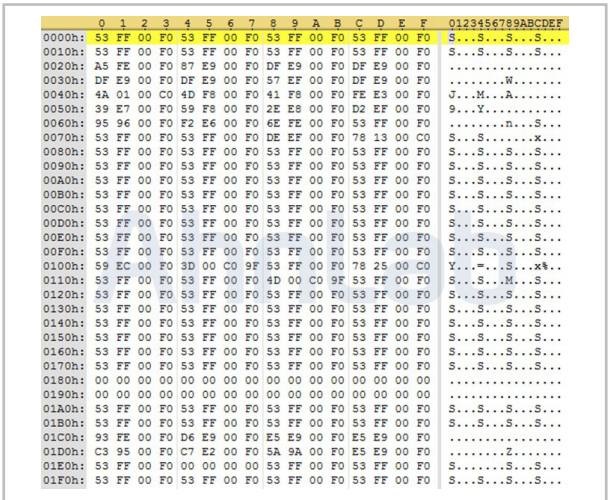
[그림 8] 감염 MBR 코드가 파괴하는 디스크 영역 개요

디버깅 시 확인된 메모리 0번지의 값은 [그림 9]와 같으며, 이와 같은 내용이 디스크에 쓰여진다.



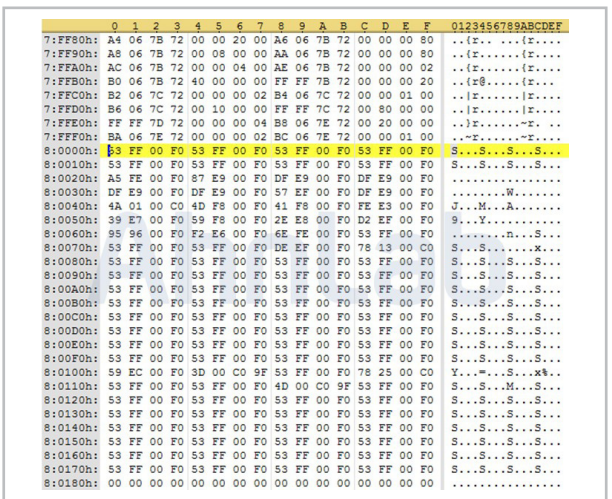
[그림 9] 감염 MBR 코드가 디스크 파괴시 참조하는 메모리 0번지

하드디스크의 0번 섹터의 값이 [그림 10]과 같이 변경된 것을 확인할 수 있다.



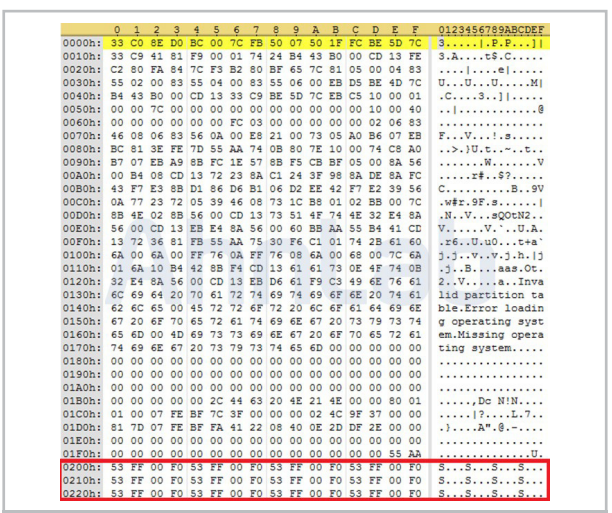
[그림 10] 첫 번째 루프 시 변경된 하드디스크 0 섹터

두 번째 루프에서 변경되는 영역은 [그림 11]과 같이 80000h 위치부터 8000h 크기만큼이다.



[그림 11] 두 번째 루프 시 변경된 하드디스크 40h 섹터

100h만큼 루프를 돌면서 하드디스크를 파괴한 후 현재 (감염된)MBR이 올라와 있는 메모리 영역인 7C00h 오프셋의 1 섹터 크기(200h)를 하드디스크의 0번 섹터에 쓰는(write) 코드가 수행된다. 최초 루프의 시작 위치로 점프하여, 디스크 끝부분까지 수행된다.



[그림 12] 루프 종료 시 복원되는 MBR 영역

[그림 12]에서 확인할 수 있는 바와 같이 루프 종료 시 MBR 영역은 감염 시와 동일하게 복원된 것처럼 보이나, 200h 이후의 디스크 영역은 최초 루프 시 변경된 데이터임을 확인할 수 있다.

2. 파일 파괴

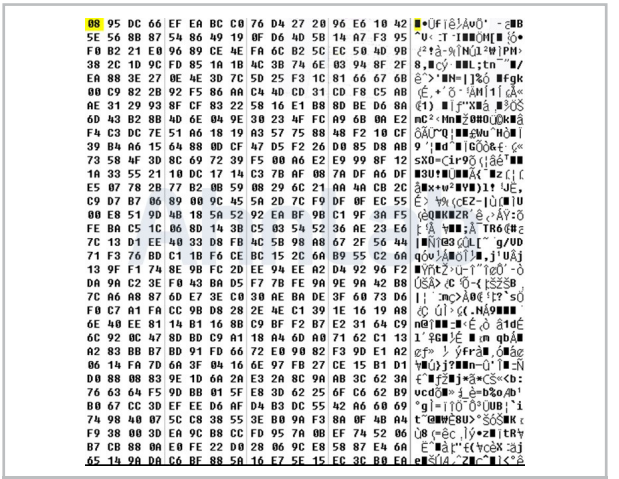
시스템의 모든 드라이브의 파일과 관련해 8.3 DoS 이름 규칙이거나 확장자가 *.exe, *.dll, *.ocx, *.sys인 경우, 즉시 해당 파일을 삭제한다. 그렇지 않으면 파일의 마지막에 0x80을 덧붙이고 파일의 시작부터 0xE00(또는 파일 사이즈)만큼 특정 코드로 덮어 씌운다.

'abcdefghijklmnopqrstuvwxyz0123456789' 테이블에서 파일명만큼 난수를 발생시켜 랜덤하게 파일의 이름을 변경한 후 삭제한다. 이를 정리하면 [표 1]과 같다.

	파일 확장자명	삭제 시점 및 방법
1	*.sys, *.ocx, *.dll, *.exe	즉시 삭제
2	*.nms, *.png, *.jpeg, *.jpg, *.gif, *.bmp, *.mp4, *.wmv, *.mpeg, *.flv, *.mpg, *.avi, *.php3, *.php, *.do, *.jsp, *.asp, *.aspx, *.htm, *.html	랜덤한 이름으로 바꾼 후 삭제
3	그 외 파일 파일명	마지막에 0x80, 파일크기 만큼 [그림 13]과 같은 쓰레기값을 덮어 쓰는 방식으로 삭제

[표 1] 파일 확장자별 삭제 증상

해당 악성코드가 웹 스크립트 언어 확장자에 대한 처리를 달리한 점으로 미루어 공격자는 기업의 웹 서버 파괴 목적에 좀 더 무게를 둔 것으로 짐작된다. 또한 서버와 PC의 사용 행태를 고려한 다양한 변형을 제작했을 것으로 추측할 수 있다.



[그림 13] 파일 삭제를 위해 쓰여지는 쓰레기값([표 1]의 조건 3의 경우)

이전과는 또 다른 하드디스크 파괴 증상 나타나

6.25 사이버테러에 사용된 하드디스크 파괴 악성코드는 지난 3.20 사이버테러의 그것과는 차이를 보인다. 하드디스크 파괴 기능이 MBR에 삽입되어 있으며, 데이터 영역 삭제 시에 특정 문자열을 이용했던 3.20 사이버테러와 달리 이번 공격에는 랜덤한 문자열로 덮어쓰기를 시도한다. 또한 감염 즉시 데이터 영역을 삭제하는 것이 아니라 재부팅 시 삭제하는 점, 이 밖에 패스워드나 바탕화면을 변경하는 점 등도 이전에는 없던 것이다.

한편, 해당 악성코드는 안랩 내부 테스트 결과 개인 PC에서도 동작하는 것으로 확인됐다. 기업 및 기관의 서버를 타깃으로 제작된 악성코드로 추측되지만 개인 사용자 또한 각별한 주의가 요구된다.

1부_카빙, 복구와 복원 : PE 카빙

사라진 악성코드까지 추적한다

카빙(carving)의 동사 원형인 'carve'의 사전적 정의는 '조각하다, 깎아서 만들다'로, 어떠한 덩어리를 조각하여 의미 있는 형태로 만드는 행위를 뜻한다. 컴퓨터 기술 용어인 '카빙' 역시 어떤 덩어리, 즉 데이터에서 불필요한 부분을 제거하거나 삭제하고 필요한 부분을 합치는 등의 작업를 통해 애플리케이션이 읽을 수 있는 형태의 데이터를 만드는 작업이라 할 수 있다. 즉, 필요에 따라 데이터를 추출하고 재구성함으로써 파일을 복구하거나 시스템을 분석할 수 있어 복잡다단한 시스템과 나날이 고도화되는 공격을 대면하고 있는 분석가들에게 필수적인 기술이다.

이에 월간 안을 통해 PE 카빙을 시작으로 다양한 파일 포맷에 따른 카빙 기술과 활용 방안을 살펴본다.

〈연재 목차〉

이번 월간 안에서 카빙 방법과 활용을 살펴볼 대상은 PE(Portable Executable)다. 윈도우 운영체제에서 실행 가능한 파일로, 윈도우에서 동작하면서 서비스 및 기타 유용한 기능을 제공한다. 일반적으로 많이 사용하는 계산기, 그림판, 메모장 등도 PE 구조체로 이루어져 있다.

PE 포맷

PE(Portable Executable) 포맷은 마이크로소프트의 윈도 3.1부터 지원되는 실행 파일의 형식을 말한다. 유닉스 COFF(Common Object file format)를 기반으로 나왔으며, PE 포맷을 사용하는 파일의 확장자는 cpl, exe, dll, ocx, vxd, sys, scr, drv가 있다.

다양한 운영 체제에서의 이식성을 보여준다는 뜻에서 이식이 가능한 '실행 형식(Portable Executable)'이라는 이름이 붙었다.

* 출처: 위키피디아, http://ko.wikipedia.org/wiki/PE_포맷

이처럼 PE는 편리한 기능을 제공해 다양한 프로그램에 이용되지만 악성코드 제작에 악용되는 경우도 빈번하다. PE가 악성코드로 동작할 경우, 자기 자신을 숨기거나 공격 후 자기 자신을 삭제하기도 한다. 이 때문에 분석가들은 악성코드를 찾는데 실패할 수도 있고, 특히 해당 악성코드가 삭제된 경우에는 분석할 대상을 확인할 수 없는 경우도 있다. 바로 이럴 때 이용할 수 있는 방법이 카빙이다.

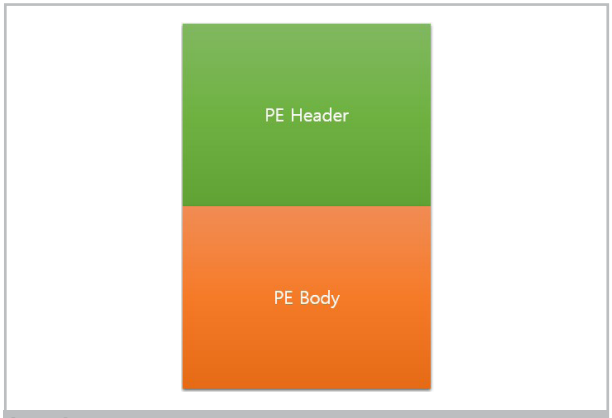
삭제된 데이터를 되살리는 카빙

파일 시스템은 파일이 삭제되더라도 데이터는 갖고 있다. 메타데이터에만 '삭제'된 것으로 표시하고 실제 데이터는 하드디스크에 남아있다는 의미이다. 즉, 하드디스크에 데이터가 남아 있기 때문에 이 같은 경우에는 데이터 복원이 가능하다.

그렇다면 PE 파일이 삭제된 경우에는 어떨까? OS에 의해 생성된 데이터나 기타 데이터가 삭제된 데이터의 자리를 덮어쓰지 않았다면, PE 파일 역시 삭제됐더라도 하드디스크에 남아있다. 남아있는 데이터는 그 형태를 알고 있다면 완벽히 복원할 수 있다. 복원을 위해 PE의 생김새(?) 먼저 알아보도록 하자.

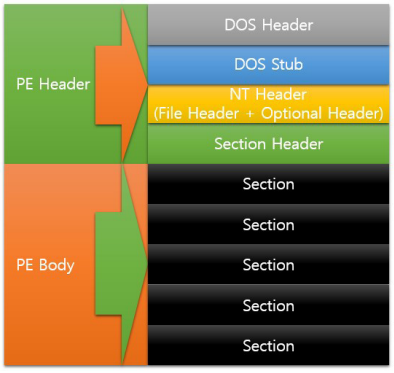
생김새(?)를 알아야 카빙도 한다

데이터를 카빙하려면 우선 원하는 데이터가 어떻게 생겼는지, 어디에 위치했는지를 알아야 한다. PE 데이터를 임의의 덩어리에서 뽑아내기 위해서는 PE가 어떻게 생긴 것인지, 그 구조부터 알아야 한다. [그림 1]과 같이 PE는 크게 헤더(Header)와 바디(Body)로 나뉜다.



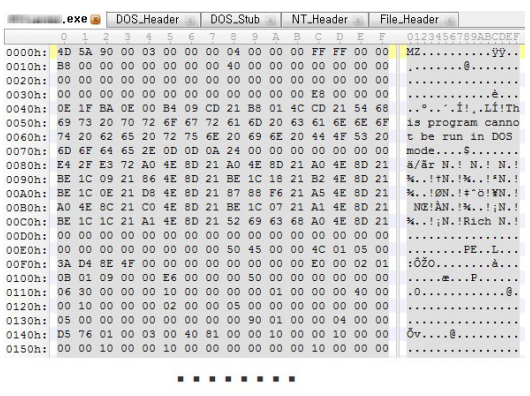
[그림 1] PE 기본 구조

PE 헤더는 DOS Header + DOS Stub + NT Header(File Header + Optional Header) + Section Headers를 포함하고 있으며, PE 바디는 여러 섹션(Section)을 포함하고 있다. 대략 [그림 2]와 같이 구성되어 있는 것으로 이해할 수 있다.



[그림 2] PE 구조체의 구성

그럼 지금부터 실제 헥스 에디터를 이용해 PE의 구조를 살펴보도록 하자. PE 구조체를 가진 데이터를 헥스 에디터로 열어보면, 육안으로 식별이 불가능한 숫자들이 가득하다. 그러나 하나씩 천천히 살펴보면 어떤 것이 중요한 부분인지 파악할 수 있다.

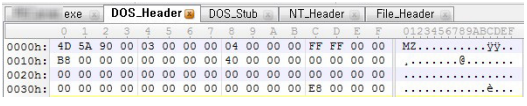


[그림 3] 헥스 에디터로 확인한 PE 구조 예시

PE 상세 구조

1. DOS Header

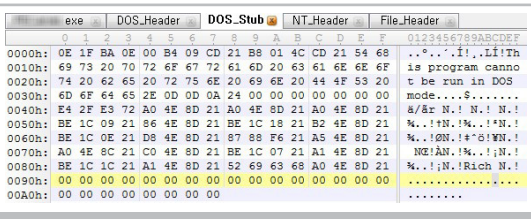
[그림 3]의 PE 구조체를 살펴보면, 처음 위치에 0x4D 0x5A라는 값을 확인할 수 있다. 이 값을 매직(Magic)이라고 하며, 아스키(ASCII) 값으로 MZ라고 표현된다. 이 데이터는 MZ를 기준으로 0x40만큼의 크기를 갖고 있다. 또한 0x3C 위치에서 DWORD로 0xE0 0x00 0x00 0x00라는 값을 갖고 있다. 이 값을 e_lfanew라고 하며, 리틀 엔디안(Little Endian)으로 0x000000E0이라고 읽는다. 이 값은 다음 구조체인 NT 헤더(Header)가 시작되는 위치를 가리킨다.



[그림 4] DOS 헤더(Header) 예시

2. DOS Stub

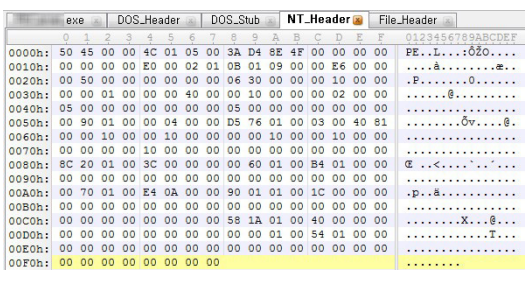
DOS 스터브(Stub)는 DOS에서 사용하는 코드를 담고 있다. 이 데이터는 존재 자체도 옵션으로 부여되고 사이즈 또한 일정하지 않다. 게다가 해당 데이터가 PE에서 확인되지 않아도 동작에는 전혀 지장이 없다.



[그림 5] DOS 스터브(Stub) 예시

3. NT Header

DOS 스터브 부분을 지나면 [그림 6]과 같이 아스키 값으로 'PE'라고 표시된 글씨를 확인할 수 있다. 이 지점부터가 NT 헤더(Header)라고 볼 수 있다. PE는 헥스(Hex) 값으로 0x50 0x45 0x00 0x00라는 값으로 표현된다. NT 헤더의 크기는 0xF8이며, 해당 데이터의 크기는 파일 헤더(File Header)와 옵셔널 헤더(Optional Header)를 포함한 크기이다. 즉, 이 데이터의 크기는 NT 헤더 시그니처(PE) + 파일 헤더 + 옵셔널 헤더 = 0xF8인 것이다.

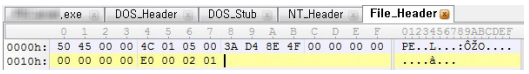


[그림 6] NT 헤더(Header) 예시

4. File Header

NT 헤더 중 파일 헤더(File Header)는 'PE'라는 아스키 값으로 시작한다. 이 값은 0x50 0x45 0x00 0x00이며, NT 헤더의 시작이자 파일 헤더의 시작이 된다. 파일 헤더는 PE가 갖고 있는 섹션(Section)의 수와 옵셔널 헤더(Optional Header)의 크기를 담고 있다. 섹션 개수는 오프셋(offset) 0x7에 위치하며, 크기는 WORD이다.

[그림 7]에서 확인할 수 있는 바와 같이, 이 데이터의 섹션 개수는 0x05이다. 옵셔널 헤더(Optional Header)의 크기는 NT 헤더가 시작되는 부분으로부터 0x14 위치에 WORD값으로 존재한다. [그림 7]에서는 0x00 0xE0이며, 해당 값은 224이다.

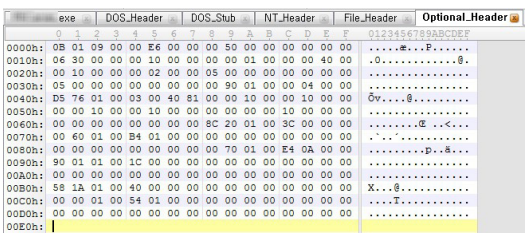


[그림 7] 파일 헤더(File Header) 예시

5. Optional Header

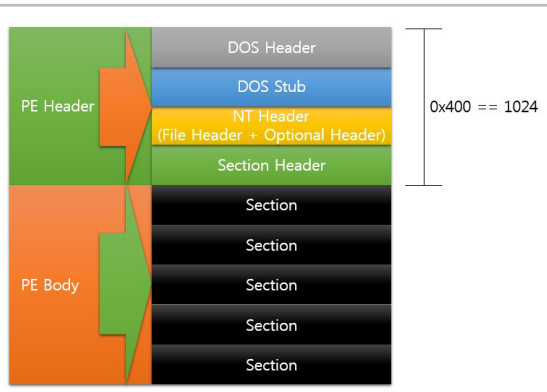
NT 헤더 중 옵셔널 헤더(Optional Header)는 파일 헤더가 끝나는 부분에서 바로 이어진다. 이 크기는 [그림 7]에서 확인한 바와 같이 E0h 이다. 옵셔널 헤더의 매직 값(Magic Value)은 32bit 또는 64bit에 따라 각기 다르다. 32bit는 10Bh, 64bit는 20Bh이다. 이 값을 리틀 엔디안으로 보면 0x0B 0x01, 그리고 0x0B 0x02가 된다. [그림 8]에서 확인할 수 있는 PE파일은 32bit로, 0x0B 0x01의 값을 갖고 있다.

한편, 옵셔널 헤더는 PE 헤더의 전체 크기에 대한 정보를 담고 있다. 이 정보의 위치는 옵셔널 헤더가 시작하는 곳으로부터 0x3C 위치에 있다. [그림 8]에서는 400h라는 값으로 표현되고 있다. 이를 통해 확인할 수 있는 PE 헤더의 전체 크기는 400h, 즉 1024이다.



[그림 8] 옵셔널 헤더(Optional Header) 예시

현재까지 파악한 내용을 통해 알 수 있는 이 데이터의 PE 헤더 크기는 [그림 9]와 같다.

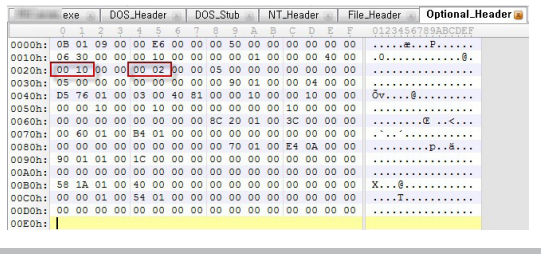


[그림 9] PE 헤더 크기

PE 카빙과 관련해 한 가지 더 고려해야 하는 부분이 있다. 바로 섹션 얼라인먼트(Section Alignment)와 파일 얼라인먼트(File Alignment)이다. 섹션 얼라인먼트는 메모리에서 섹션이 차지하는 크기를 뜻하며 파일 얼라인먼트는 메모리에 올라가기 전, 파일 시스템에 남아있는 파일 안의 섹션 크기를 말한다. 따라서 카빙하고자 하는 대상이 메모리 파일인 경우 섹션 얼라인먼트를, 파일 시스템인 경우 파일 얼라인먼트를 참고하면 섹션의 크기를 확인 할 수 있다.

또한 섹션 얼라인먼트는 옵셔널 헤더의 0x20 위치에 WORD 값으로 표시된다. [그림 10]의 경우, 이 값은 0x00 0x10이다.

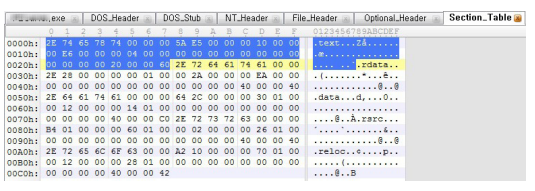
파일 얼라인먼트는 옵셔널 헤더에서 0x24 위치에 WORD 값으로 존재하며 값은 0x00, 0x02이다. 즉 메모리 상에서 섹션은 1000h의 크기를, 파일 시스템에서는 200h의 크기를 갖는다는 것을 알 수 있다.



[그림 10] 옵셔널 헤더에 표시되는 섹션 얼라인먼트 및 파일 얼라인먼트

6. Section Header

섹션은 1개 이상 존재하며, 섹션의 수는 NT 헤더의 파일 헤더에서 확인할 수 있다. 일반적으로 코드(CODE), 데이터(DATA), 리소스(RESOURCE) 등의 섹션으로 나뉘며 각 섹션의 사이즈는 0x28이다. 만약 섹션의 수가 3개일 경우, 섹션 헤더(Section Header)의 크기는 0x28x3이라고 생각하면 된다.



[그림 11] 섹션 헤더(Section Header) 예시

섹션 헤더 뒤에는 섹션들이 붙어 있다. 이 같은 섹션들을 PE 바디(PE Body)라고 부르며, 해당 데이터의 크기를 모두 구하려면 각 섹션 헤더의 처음으로부터 0x10만큼 이동한 곳의 DWORD 값을 확인해야 한다. PE 바디를 구성하고 있는 5개 섹션의 총 크기를 구하면 [그림 12]와 같다.

PE Body

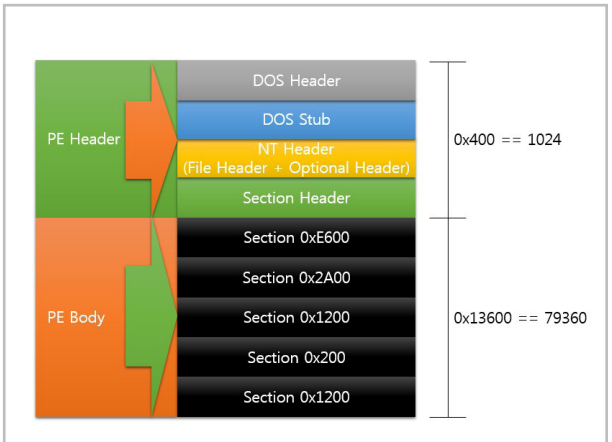
text == E600h

rdata == 2A00h

data == 1200h

rsrc == 200h

reloc == 1200h



[그림 12] PE 바디(PE Body)내 5개 섹션의 크기 총합

PE 카빙을 위한 툴 작성 방법

지금까지 카빙에 필요한 PE의 속성을 모두 살펴봤다. 이를 토대로 이제 임의의 더미 파일에서 PE만을 카빙하는 도구를 제작해 보도록 하자. 이를 위해 먼저 PE 속성 중 어떤 부분을 확인해야 하는지 간략하게 정리하면 다음과 같다.

① MZ로 PE 시작 부분 확인

더미 파일 안에서 PE의 시작인 MZ를 찾는다. 실제로 찾는 값은 헥스 값인 **0x4D 0x5A**이다.

② e_lfanew로 NT 헤더 확인

e_lfanew의 값(**Offset 0x3C**)을 확인한 후, 이 값이 아스키 값의 PE인지 확인한다. 아스키 값 PE는 헥사 값으로 **0x50 0x45 0x00 0x00**이다.

③ 섹션 수 확인

섹션 수는 파일 헤더의 첫 부분으로부터 오프셋(Offset) 0x06을 이동 후, WORD 크기로 확인할 수 있다. 위치 계산 방법은 다음과 같다.

```
Offset (MZ 시작) + Offset (NT 헤더 시작) + Offset 0x06의 WORD 값
```

④ 옵셔널 헤더 크기

옵셔널 헤더의 크기는 파일 헤더의 시작부터 **Offset 0x14** 이동 후 **WORD** 크기로 확인할 수 있다. 위치 계산 방법은 다음과 같다.

```
Offset (MZ 시작) + Offset (NT헤더 시작) + Offset 0x14의 WORD 값
```

⑤ 옵셔널 헤더 증명

옵셔널 헤더의 시작이 **0x0B 0x01**이나 **0x0B 0x02**인지 확인한다. 오프셋의 위치 계산은 아래와 같다.

```
Offset (MZ 시작) + Offset (NT 헤더) + Offset 0x18의 WORD 값
```

⑥ PE 헤더 크기

옵셔널 헤더의 시작부터 **오프셋 0x3C**만큼 이동하면 PE 헤더의 크기를 얻을 수 있다. 위치 계산 방법은 아래와 같다.

```
Offset (MZ 시작) + Offset (NT 헤더) + Offset 0x18+ Offset 0x3C의 DWORD 값
```

⑦ 섹션 헤더 위치

섹션의 크기를 알기 위해서는 섹션 헤더로 이동해야 한다. 섹션 헤더는 NT 헤더 다음에 위치한다. 위치 계산 방법은 아래와 같다.

```
Offset (MZ 시작) + Offset (NT 헤더) + Offset 0x18 + 'Offset [Offset (MZ 시작) +
Offset (NT 헤더 시작) + Offset (0x14)]'의 WORD 값
```

⑧ 섹션 헤더의 크기

섹션 헤더의 크기는 **각각 0x28**이다. 섹션 수 x 0x28을 하면 섹션 테이블의 전체 크기를 알 수 있다.

⑨ 섹션의 크기

섹션의 크기는 섹션 헤더에 존재한다. 각각의 섹션 헤더의 시작으로부터 **Offset 0x10**을 이동 후 DWORD 크기로 확인 할 수 있다.

위의 항목들에 대한 확인을 마친 후 PE 카빙을 수행하기 위해 파이썬으로 간단한 프로그램을 작성했다. 관련 링크와 해당 코드는 아래와 같다.

* 참고 링크 : https://dorumugs-tools.googlecode.com/files/PE_Carver.py

```
import optparse
import glob
import struct
import time
import binascii
import os

def Timestamp(epoch=None):
  if epoch == None:
    localTime = time.localtime()
  else:
    localTime = time.localtime(epoch)
  return '%04d%02d%02d-%02d%02d%02d' % localTime[0:6]

def LogLine(line):
  print('%s: %s' % (Timestamp(), line))


def File2Data(filename):
  try:
    f = open(filename, 'rb')
  except:
    return None
  try:
    return f.read()
  except MemoryError:
    return MemoryError
  except:
    return None
  finally:
    f.close()

def Data2File(data, filename):
  try:
    f = open(filename, 'wb')
  except:
    return False
  try:
    f.write(data)
  except:
    return False
  finally:
    f.close()
  return True


def CheckPEStructure(baseAddress, data, rData, bSize, filename):
  rSize = os.path.getsize(rData)
  found = False
  index = 0

  while index != -1:
    index = data.find('\x4D\x5A', index)
    # MZ찾기
    if index != -1:
      e_lfanew = index + 60
      e_lfanew_value = int(binascii.hexlify(data[e_lfanew:e_lfanew+4]), 16)
      e_lfanew_value_little = struct.pack('<I', e_lfanew_value)
      e_lfanew_value_little = e_lfanew_value_little.encode('hex')
      e_lfanew_value_little = int(e_lfanew_value_little, 16)

      NT_Header = index + e_lfanew_value_little
      # NT Header 위치 확인

      verifyOptional_Header = binascii.hexlify(data[NT_Header+24:NT_Header+26])

      if '50450000' ==  binascii.hexlify(data[NT_Header:NT_Header+4]) and \
      '0b01' == verifyOptional_Header or '0b02' == verifyOptional_Header:
      # PE 및 Optional_Header 확인
```

```
        numberOfSections = binascii.hexlify(data[NT_Header+6:NT_Header+7])
        if len(numberOfSections) > 0:
            numberOfSections_little = int(numberOfSections, 16)
            numberOfSections_little = struct.pack('<I', numberOfSections_little)
            numberOfSections_little = numberOfSections_little.encode('hex')
            numberOfSections_big = int(numberOfSections_little[0:2], 16)
            # Section 개수 확인

        sizeOfOptional_Header = binascii.hexlify(data[NT_Header+20:NT_Header+22])
        if len(sizeOfOptional_Header) > 0:
            sizeOfOptional_Header_little = int(sizeOfOptional_Header, 16)
            sizeOfOptional_Header_little = struct.pack('<I', sizeOfOptional_Header_little)
            sizeOfOptional_Header_little = sizeOfOptional_Header_little.encode('hex')
            sizeOfOptional_Header_big = int(sizeOfOptional_Header_little[0:4], 16)
            # Optional Header 크기 확인

        sizeOfPE_Header = binascii.hexlify(data[NT_Header+84:NT_Header+88])
        if len(sizeOfPE_Header) > 0:
            sizeOfPE_Header_little = int(sizeOfPE_Header, 16)
            sizeOfPE_Header_little = struct.pack('<I', sizeOfPE_Header_little)
            sizeOfPE_Header_little = sizeOfPE_Header_little.encode('hex')
            sizeOfPE_Header_big = int(sizeOfPE_Header_little, 16)
            # PE Header 크기 확인

        if len(sizeOfOptional_Header) > 0:
            count = numberOfSections_big
            defaultSize = 40
            # Section Default 크기는 0x28
            totalSizeOfSection = 0
            startOfSection_Header = NT_Header + 24 + sizeOfOptional_Header_big
            startOfSection_HeaderOffset = startOfSection_Header
            # Section Header 시작 위치 찾기
            sizeOfSection_header = 40 * numberOfSections_big
            endOfSection_Header = startOfSection_Header + sizeOfSection_header
            while count > 0:
                sizeOfSection = startOfSection_Header + 16
                    sizeOfSection_little = int(binascii.hexlify(data[sizeOfSection:sizeOfSecti
on+4]), 16)
                sizeOfSection_little = struct.pack('<I', sizeOfSection_little)
                sizeOfSection_little = sizeOfSection_little.encode('hex')
                sizeOfSection_big = int(sizeOfSection_little, 16)
                #print "sizeOfSection is " + str(sizeOfSection_big)
                startOfSection_Header = startOfSection_Header + defaultSize
                totalSizeOfSection = totalSizeOfSection  + sizeOfSection_big
                count = count - 1

            totalLength = sizeOfPE_Header_big + totalSizeOfSection

        filenameExists = os.path.exists(filename)

        if filenameExists == False:
            os.mkdir(filename)
            running = 1
        elif filenameExists == True and running == 1:
            running = 1
        else:
            print "Directory exists"
            return True

        resultDir = str(os.path.abspath(filename)) + "\\" + str(index)
        print resultDir
        f = open(resultDir, 'wb')
        f.write(data[index:index+totalLength])
        # 검증이 끝난 데이터를 파일로 생성

        print "MZ Offset             " + str(index)
        print "MZ Hex Value          " + binascii.hexlify(data[index:index+2])
        print "NT_Header Offset      " + str(NT_Header)
            print "NT_Header Hex Value      " + binascii.hexlify(data[NT_Header:NT_
Header+4])
        print "Number Of Sections     " + str(numberOfSections_big)
        print "Size Of Optional_Header " + str(sizeOfOptional_Header_big)
        print "Verify Optional_Header  " + verifyOptional_Header
        print "Size Of PE_Header      " + str(sizeOfPE_Header_big)
        print "Size Of Section        " + str(sizeOfSection_header)
        print "Start of Section       " + str(startOfSection_HeaderOffset)
        print "End of Section         " + str(endOfSection_Header)
        print "Total Size Of Section   " + str(totalSizeOfSection)
        print "Total Length          " + str(totalLength) + '\n\n'

        index = index + int(bSize)
      index = index + int(bSize)
    return True


def ExtractPEFromFile(rawDataName, blockSize, folderName, filename):
    LogLine('Start')
    LogLine('Reading file %s' % rawDataName)
    rawData = File2Data(rawDataName)
    #print rawData
    if rawData == None:
        LogLine('Error reading file')
    if rawData == MemoryError:
        LogLine('File is too large to fit in memory')
    else:
        LogLine('Searching for PE Format\n')
        CheckPEStructure(0, rawData, rawDataName, blockSize, filename)
        LogLine('Done')

    return True


def Main():
     Parser = optparse.OptionParser(usage='usage: python PE_Carver.py -i input_dummy
-b block_size -o output_folder')
    Parser.add_option('-i', '--input', dest='input', default=False, help='input dummy')
    Parser.add_option('-b', '--bsize', dest='bsize', default=False, help='Block Size')
     Parser.add_option('-o', '--output', dest='output', default=False, help='The Directory
included Carved Files')
    (options, args) = Parser.parse_args()

    if options.input == False or options.bsize == False or options.output == False:
        print('')
        print('     python PE_Carver.py --help\n\n')
        Parser.error("incorrect number of arguments\n\n")
        return
    else:
        a = []
        rData = options.input
        bSize = options.bsize
        folder = options.output

        #filenames = sum(map(glob.glob, filenames), [])

        ExtractPEFromFile(rData, bSize, folder, options.output)

if __name__ == '__main__':
    Main()
```

위의 도구를 이용해 카빙된 데이터는 아웃풋 폴더(output folder)에 쌓인다. 이때 각 파일의 이름은 카빙된 시작 오프셋으로 부여된다.

```
사용법 : python PE_Carver.py -i input_dummy -b block_size -o output_folder
옵션 :
 -h, --help          show this help message and exit
 -i INPUT, --input=INPUT
                     Input dummy
 -b BSIZE, --bsize=BSIZE
                     Block Size
 -o OUTPUT, --output=OUTPUT
                     The Directory included Carved Files
```

*** 예제 : python PE_Carver.py -i ntfs_dd -b 1024 -o result**

```
C:₩Safe7M_Carving-PE>python PE_Carver.py -i ntfs_dd -b 1024 -o result
20130617-101022: Start
20130617-101022: Reading file ntfs_dd
20130617-101023: Searching for PE Format

C:₩Safe7M_Carving-PE₩result₩8052736
MZ Offset               8052736
MZ Hex Value            4d5a
NT_Header Offset        8052976
NT_Header Hex Value     50450000
Number Of Sections      6
Size Of Optional_Header 240
Verify Optional_Header  0b02
Size Of PE_Header       1024
Size Of Section         240
Start of Section        8053240
End of Section          8053480
Total Size Of Section   242688
Total Length            243712


C:₩Safe7M_Carving-PE₩result₩8298496
MZ Offset               8298496
MZ Hex Value            4d5a
NT_Header Offset        8298736
NT_Header Hex Value     50450000
Number Of Sections      6
Size Of Optional_Header 240
Verify Optional_Header  0b02
Size Of PE_Header       1024
Size Of Section         240
Start of Section        8299000
End of Section          8299240
Total Size Of Section   1073664
Total Length            1074688


20130617-101023: Done

C:₩Safe7M_Carving-PE>
```

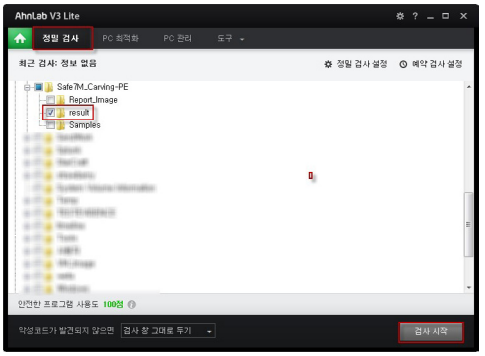
[그림 13] PE 카빙 툴에 의해 카빙된 데이터

카빙된 파일의 활용

대상 파일의 구조부터 구조별 항목 확인까지 녹록지 않은 과정을 거쳤다. 이제 카빙을 통해 얻은 결과물을 활용하는 방법에 대해 살펴 보자.

(1) 삭제된 파일 복구 및 악성 여부 점검

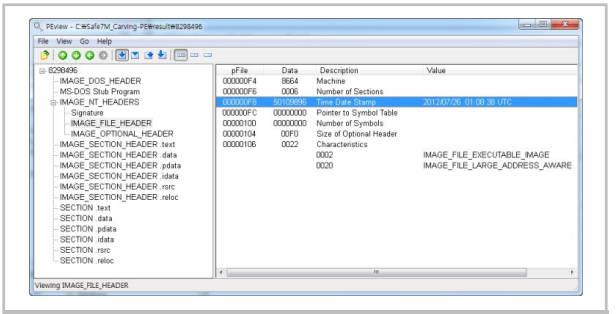
카빙을 수행하면 삭제된 파일도 복구가 가능하다. 이렇게 복구된 파일은 악성 여부의 판별을 위해 백신 프로그램을 이용해 점검한다.



[그림 14] 복구된 파일에 대한 악성 여부 검사

(2) APT 공격 등의 연관성 파악

PE 안에는 컴파일된 시간 정보가 남아 있다. 해당 파일이 언제 만들어졌는지 알 수 있는 것이다. 이를 통해 침해사고 발생 시, 공격과 관련한 파일의 추적이 가능하다. 예를 들어, 만약 PE를 통해서 APT 공격을 받았을 경우 최근 컴파일된 시간 정보를 이용해 해당 파일이 APT 공격에 이용된 악성코드인지를 판별할 수 있다.



[그림 15] PE에 존재하는 시간 정보

PE 카빙으로 공격의 실체를 파악하는 단서를 얻는다

지금까지 임의의 데이터를 깎아서 PE를 만들어내는, 즉 PE 카빙 방법을 살펴봤다. 파일 시스템에서 PE를 카빙할 경우, 삭제된 PE 및 존재하는 PE에 대해서도 악성 여부를 점검하고 판단할 수 있다. PE 카빙이 APT 공격으로 침해를 입은 시스템에서 흔적도 없이 사라진 악성코드를 발견할 수 있는 열쇠가 될 수도 있다는 의미다.

이처럼 하나의 파일이 전체적인 분석의 실마리를 제공하기도 하는 만큼 카빙을 통해 파일을 복구하고 분석하는 것은 포렌식에 있어 매우 중요하다. 특히 최근 자주 등장하는 공격 이후 자기 자신을 삭제하는 악성코드를 카빙을 이용해 복구할 수도 있으므로 카빙을 이용해 분석하는 것을 강력히 추천한다.

[안랩 기술 전문 파트너 ATP(Advanced Technology Partner)] ㈜앤솔루션

"망분리 솔루션 분야의 최고를 향해 도약"

최근 대규모 해킹 사건이 빈발하면서 망분리 솔루션에 대한 관심이 급증하고 있다. 업무망과 일반 네트워크망을 분리함으로써 네트워크를 통해 유입되는 악성 파일을 원천 차단할 수 있어서다. 특히 각 기업과 기관에서는 구축 기간과 비용 등을 고려해 물리적 망분리보다 논리적 망분리에 주목하고 있다. 논리적 망분리 솔루션인 안랩 트러스존(AhnLab TrusZone)에 대한 문의가 쇄도하고 있는 이유다. 그 덕분에 안랩 트러스존의 ATP(Advanced Technology Partner)인 ㈜앤솔루션 역시 어느 때보다 바쁜 여름을 보내고 있다.

앤솔루션

앤솔루션(대표 진영인)은 2010년 정부통합전산센터 구축 때부터 안랩과 인연을 맺었다. 안랩이 통합전산센터 진입을 위한 전문파트너로 앤솔루션에 역할을 해줄 것을 요청하면서부터다. 성공적인 통합전산센터 수주는 2011년 채널파트너 계약 체결로 연결됐고 이듬해인 2012년 트러스존 기술 전문 파트너(ATP) 선정으로 이어졌다.

▲진영인 앤솔루션 대표이사

물리적 망분리 대신 논리적 망분리를 택하다

양사는 논리적 망분리를 미래지향적 보안사업으로 높이 평가, 2009년 망분리 시장에 뛰어들었다. 하지만 당시 시장의 반응은 사뭇 달랐다. 논리적 망분리 시장이 물리적 망분리에 비해 아직 성숙기에 접어들지 않았던 것. 물리적 망분리는 '1인 2PC' 환경을 조성해 업무망과 인터넷망을 사실상 각기 다른 PC를 설치하는 방식으로 비용 부담이 높았다. 그럼에도 불구하고 '논리적 망분리는 잠재적인 보안 위협에 원천 대응이 어렵다'는 세간의 인식을 크게 개선하지는 못했다.

그러나 논리적 망분리에 대한 이 같은 '홀대 아닌 홀대'는 그리 오래 지속되지 않았다. 논리적 망분리 솔루션 자체의 경쟁력이 높아졌기 때문이다. 논리적 망분리 가운데 안랩의 트러스존이 택하고 있는 클라이언트 기반 가상화 방식(CBC방식)은 초기 도입 비용이 낮은 데다 쉽고 간편하게 설치할 수 있는 장점이 있다. 한때 물리적 망분리에 비해 보

안성이 낮다는 지적도 있었으나 이 역시 기술적으로 완성도를 높이면서 물리적 망분리를 대체할 수 있는 차세대 솔루션으로 주목받기 시작했다.

일부 기관이 선제적으로 도입한 논리적 망분리가 안정적으로 구축·운용된 것도 긍정적인 영향을 줬다. 그 구체적인 예가 2012년 안랩과 앤솔루션이 공동으로 참여한 공공 기관 망분리 구축 사업이다. 진영인 앤솔루션 대표는 "당시 우리는 다른 사업을 거의 전폐하다시피 매달렸다"며 "안랩의 전폭적인 지원과 우리의 노력이 더해져 성공리에 마무리된 사업"이라고 밝혔다.

▲앤솔루션 로고. 영문 'and'를 축약한 로고와 사명이다. 서비스와 솔루션, 장비 등을 통합해서 제공할 수 있는 컨버전스가 가능한 기업이라는 의미가 담겨 있다.

안랩과의 성과, 망분리 패러다임을 바꾸다

진영인 대표의 말처럼 해당 사업은 양사 모두에 트러스존, 나아가 논리적 망분리에 대해 새로운 기회를 모색할 수 있는 발판이 됐다. 언론의 인터뷰 요청을 시작으로 다른 공공 및 금융분야의 문의가 쇄도했다.

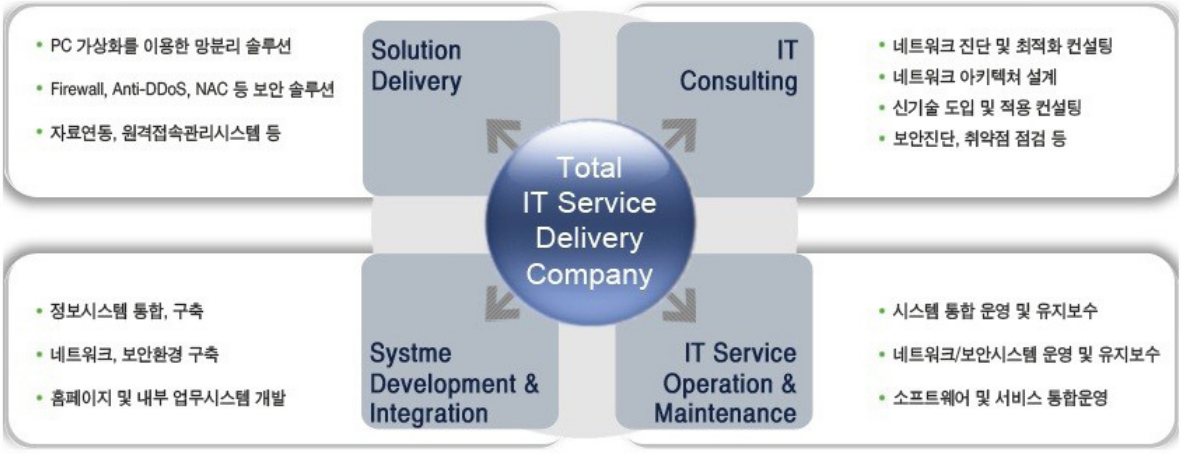
최근 지속적으로 발생한 대규모 보안 위협도 망분리 솔루션에 대한 필요성을 한 단계 끌어올리는 데에 기여했다. 올해 초 잇달아 열린 안랩의 고객 대상 세미나에서 어느 제품보다 트러스존에 대한 관심과 열기가 뜨거웠던 것이 이를 방증한다.

앤솔루션의 매출 규모가 나날이 증가하고 있다는 점도 망분리 솔루션의 시장성이 확대되고 있음을 짐작케 한다. 앤솔루션은 지난해 20억여 원의 매출을 달성했다. 설립 초기 미약한 성과에 비하면 가히 괄목할만한 성적표다. 특히 올해 앤솔루션은 40억여 원의 매출 목표를 세워두고 있다. 현재의 추세대로라면 매출 목표를 무난히 달성할 수 있을 것으로 앤솔루션 측은 전망하고 있다.

안경효 앤솔루션 솔루션사업본부 이사는 "성장 속도가 빠르지는 않지만 꾸준히 발전하고 있다는 사실에 주목해야 한다"며 "올해가 지나면 여느 중소기업 못지 않은 성장을 이룰 것으로 기대하고 있다"고 말했다.

경쟁력은 특화된 기술력(Advanced Technology)

앤솔루션의 이런 자신감은 단순히 망분리 사업의 시장 확대에만 기인한 것은 아니다. 앤솔루션은 자사의 경쟁력을 국가정보통신망 등 '망분리에 특화된 기술력'으로 꼽는다. 그만큼 망분리 분야의 기술력에서 자신이 있다는 얘기다. 실제 진영인 대표를 비롯한 창립 멤버들은 관련 분야에 오랜 기간 종사해온 이들이다. 공든 탑이 무너지지 않듯이, 단단한 내공을 지니고 있는 기업이기에 미래 역시 밝을 것이라고 그들은 자신하고 있다.



▲앤솔루션의 다양한 사업 영역

젊은 그대 앤솔루션, 미래는 밝다

사실 앤솔루션은 '망분리 사업에 의해, 망분리 사업을 위해' 설립된 기업이라고 할 수 있다. 앤솔루션은 망분리 솔루션이 한창 화두로 떠오른 지난 2009년 창업했다. 정부의 통합전산센터 구축작업을 시작으로 다년간 공공분야 망분리 사업의 컨설팅과 기획, 구축을 담당하던 이들이 모여 본격적으로 망분리 사업에 뛰어든 것이다.

안경효 이사는 “당시 정부 등 대다수가 물리적 망분리에 주목했지만 우리는 비용 등 물리적 망분리의 단점을 보완할 수 있는 논리적 망분리를 더 주시했다”며 “그즈음 안랩의 논리적 망분리 솔루션도 개발 단계에 있어 양사가 협력적 관계로 발전할 수 있었다”고 밝혔다.

앤솔루션의 태동에서 유추할 수 있듯이 그들은 우선 망분리 사업에서 타의 추종을 불허하는 기업이 되겠다는 목표를 세우고 있다. 하지만 궁극적으로는 고객의 요구(needs)에 맞는 여러가지 보안 솔루션을 통합 제공하는 보안 기업으로 우뚝 설 수 있기를 기대하고 있다.

실제로 앤솔루션은 망분리 사업 외에도 컨설팅과 구축, 통합 유지보수를 주요 사업으로 내걸었다. 앤솔루션의 사명에 적힌 ‘앤’이라는 글자가 ‘앤드(and)’의 축약어라는 점 역시 이를 뒷받침한다. 컨버전스 시대에 걸 맞는 통합 솔루션을 제공할 수 있는 기업이 되고픈 꿈이 사명과 그들의 가슴에 깊이 아로새겨져 있다.

앤솔루션은 기술력 외에 또 다른 경쟁력으로 책임감이 강한 구성원, 캐주얼한 조직문화를 강조하고 있다. 평균 36.5세의 젊은 조직임을 감안해 자유로운 조직 속에서 저마다의 개성을 존중하며 전문 영역을 넓혀가는 것이 결국 앤솔루션의 밝은 미래를 담보하는 길이라 그들은 강하게 믿고 있다. 진영인 대표는 “망분리와 네트워크 분야에서 최고의 기술력을 인정받는 기업이 되겠다”며 “그 성장가도를 우리의 조직원들은 물론, 안랩과 함께해 나가길 기대한다”고 말했다.

▲앤솔루션의 임직원들

안전한 여름 나기의 첫걸음

기업이 놓치지 말아야 할 IT 자산 관리법

한 여름 무더위가 성큼 다가왔다. 찌는 듯한 더위와 지루한 장마까지, 휴가를 떠나고 싶은 마음이 그 어느 때보다 간절해지는 시기다. 이런 때에 간과하기 쉬운 게 하나 있다. 바로 IT 자산 관리다. 온도에 민감한 IT 기기를 미리 관리해놓지 않는다면 필요한 경우 IT 기기를 활용하지 못해 업무에 차질이 빚어질 수 있고 기본적인 관리를 빼먹어 에너지 낭비와 각종 사고 발생 등의 문제가 생길 수도 있다. 이 때문에 차제에 이를 방지하기 위한 시의적절한 IT 자산 관리 방법을 숙지하는 게 필요하다.

IT 자산 관리가 중요한 이유

PC부터 각종 네트워크, 서버 등 기업 내 IT 자산은 비즈니스 측면에서 비중과 중요도가 높아가고 있다. 이처럼 IT 자산의 비중이 높아지면서 자연스레 화두로 떠오른 것이 하드웨어와 소프트웨어 등 IT 자산의 관리다.

IT 자산 관리는 IT와 관련한 자산의 지출 내역과 성격을 한눈에 파악할 수 있게 분류하고 통계 수치를 만들어 각종 IT 자산을 효율적으로 관리하는 체계다. 이것이 중요한 이유는 기업이 IT 자산의 취득, 운용, 보전, 이동, A/S, 업그레이드, 폐기 등의 전 과정에서 관리를 통해 비용 절감 효과를 가져올 수 있어서다. 이뿐만 아니라, 업무 효율화와 보안 측면에서도 IT 자산 관리는 긴요하게 받아들여지고 있다.

이 같은 이유로, 각 기업들은 자사의 정보시스템팀(혹은 전산실)을 별도로 마련해 전문가들을 통해 체계적이면서도 주기적으로 IT 자산 관리를 수행하고 있다. 하지만 담당 부서가 있다는 것만으로 완벽한 관리가 이뤄지고 있다고 판단하긴 이르다.

IT 기기는 구성원 개개인이 사용하는 형태가 많아, 그들 스스로 관리에 신경을 써야 하는 측면이 존재한다. 이를 테면 전원 및 분실 관리, 안전한 시스템 운용 면에서 백신(안티바이러스)의 최신 버전 업데이트 및 검사 실행, 각종 애플리케이션 업데이트 등이 그것이다. 각 기업에서도 이를 놓치지 않기 위해 정책 적용 등을 통해서 구성원들의 자발적인 관리를 독려하고 있다.

기업과 각 구성원들이 노력을 하고 있음에도 불구하고 놓치는 부분이 발생할 수 있다. 특히 여름철에는 변덕스러운 날씨로 인해 IT 기기들의 관리가 더욱 까다로운 데다가, 휴가 기간으로 업무 공백이 발생하면서 크고 작은 문제들이 생기곤 한다. 평상시에도 IT 자산 관리를 꼼꼼히 챙겨야 하지만 유독 여름에 더 신경써야 할 요소들이 적지 않은 이유다.

IT기기, 항온항습 관리 철저히 해야

IT 기기는 열과 습도에 아주 민감하다. 이는 변덕이 심한 여름 날씨에 치명적인 타격을 입을 수 있다는 얘기기도 하다. '항온항습'이 여름철 IT 관리의 첫걸음인 것은 당연하다.

실제로 PC의 내부 부품 중 CPU와 그래픽카드 등은 열 발생이 많아 여름철 더위에 더 취약해 장애가 발생할 확률이 높다. 또한 습기는 IT 기기뿐만 아니라 전자제품의 최대의 적으로, PC 내부에 습기가 많은 경우 부품의 부식으로 인해 정상 작동이 어려울 수 있다.

대표적 IT 기업인 안랩의 데이터센터의 경우 철저한 항온항습 정책을 시행하고 있다. 전문 관리 장비와 시스템을 도입해 24시간 온도는 22도, 습도는 50% 체제를 유지한다.

박제석 안랩 IT인프라팀 팀장은 "안랩은 랙서버의 전면·후면부 배치까지 고려, 항온항습 관리를 진행하고 있다"며 "최신 IT 기기들은 평균 온도 27~30도에서도 운영할 수 있는 기기도 있으므로 신규 장비를 도입하는 기업이라면 참조하는 게 좋다"고 밝혔다. 다만, 무리한 항온항습 정책은 자칫 에너지 낭비로 이어질 수 있다는 점에 유의해야 한다.

▲박제석 안랩 IT인프라팀 팀장

기업 내 구성원들이 각자 실시하는 항온항습 정책도 있다. 적정한 실내온도가 유지되는지 자주 살펴보며 PC를 장시간 사용하기 보단, 3~4시간 사용 후 잠시 종료해 식혀주는 게 좋다.

밀폐된 공간은 내부 온도가 쉽게 상승해 고열이 발생할 수 있으므로 밀폐된 공간에 장시간 노트북 등의 IT 기기를 방치하지 않아야 한다. 평소 열을 잘 발산하는 냉각 팬이나 방열패드를 장착해 사용하는 것도 좋은 방법이다.

장마철과 같이 습기가 많은 날에는 하루 한번 20여 분 PC를 가동해 내부 습기를 제거해야 하며, 휴가를 떠나 일주일 이상 PC를 사용하지 않을 시엔 내부에 습기가 생겨 고장이 발생할 수 있다는 점을 유념해야 한다.

또한 천둥번개 및 벼락이 치는 날씨도 주의가 요구된다. 번개와 벼락으로 인해 전력이 끊어지는 경우 전력선이나 인터넷선을 타고 PC에 과전류가 흘러 고장을 유발할 수 있으므로 PC의 전원 플러그를 뽑아두고 인터넷 선을 제거하는 편이 낫다.

▲IT 자산 관리에는 잠금 장치 및 전원 플러그 등 구성원 개개인의 평소 노력도 필요하다.

기본적인 자산 관리 수칙 재점검도 필수

여름철에 특화된 관리 방안은 아니지만, 다수가 자리를 비우는 휴가기간임을 감안한다면 기존에 실시해오던 기본적인 IT 자산 관리 수칙들을 재점검할 필요도 있다. 보안의 관점에서도 이는 중요하다.

예를 들어 IT 자산 관리 측면에서 가장 위험한 PC는 네트워크에 접속된 채로 아무도 사용하지 않는 이른바 '방치된 PC(기기)'다. 이런 PC들은 제로데이 취약점이 발견되더라도 업데이트가 제때 이뤄지지 않거나, 백신(안티바이러스)이 설치되어 있더라도 주기적으로 검사를 실시하지 않아 외부의 공격에 더 취약한 편이다.

휴가를 떠나기 전 주변에 이런 PC가 없는지 체크하고 필요하지 않다면 전원을 꺼두어야 한다. 특히 IT 기업의 경우 기술 개발을 위해 사내 보안 시스템이나 IT 자산 관리에서 제외시킨 '테스트 PC'가 사내 네트워크망에 접속한 채로 방치되고 있지는 않은지 살피는 것이 필수적이다.

	항목명
1	본 자산의 등록된 정보가 실물과 일치합니다.
2	본 자산의 OS, 용도, 위치, 실사용자를 정확하게 변경 입력 하였습니다.(상세하게 수정
3	본 자산의 로그인 패스워드를 영문,숫자,특수문자 혼합하여 10자 이상으로 사용합니다.
4	본 자산의 로그인 패스워드는 최소 1분기에 1번은 변경하여 사용합니다.
5	본 자산의 부팅 패스워드를 설정하여 사용합니다.
6	본 자산의 Windows의 Guest계정을 포함한 사용하지 않는 계정을 비활성화 합니다.
7	본 자산의 공유폴더를 사용하지 않습니다.
8	본 자산은 10분 이내 대기시간을 적용한 화면보호기 사용 및 잠금 기능을 사용하고 있습
9	본 자산을 이용하여 P2P를 사용하지 않습니다.
10	본 자산은 정품 소프트웨어만을 사용 합니다.

▲개인이 수행하는 '실물자산관리' 항목의 예

이밖에도 보안 계정과 패스워드 주기적 변경, 각종 애플리케이션의 업데이트, 백신의 업데이트 및 점검 수행 등도 놓쳐선 안 되는 기본 수칙들이다. 또 기업이 평소 실시해오던 IT 자산 관리 점검도 다시 한번 살펴본 뒤 휴가를 떠나는 게 좋다.

휴가철에는 IT 자산 관리자의 공백에 대한 대비책도 미리 세워둬야 한다. 안랩의 경우 24시간 수행이 가능한 비상연락망을 구축해 놓고 있다. 비상시 외부에서 접속이 가능한 노트북을 지급해 빠른 복구가 가능한 시스템을 상시 운영 중이다.

박제석 팀장은 "전문가에 의한 체계적인 관리 못지 않게, 구성원 개개인의 기본 가이드 이행도 중요하다"며 "사소하지만 반드시 지켜야 하는 원칙들을 돌아보고 잘 챙김으로써 그 어느 때보다 안전한 여름을 보내시길 바란다"고 조언했다.

V3 모바일, AV-TEST 글로벌 인증 잇따라 획득

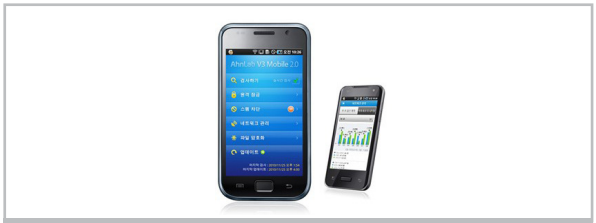
안랩(대표 김홍선, www.ahnlab.com)의 모바일 보안제품인 V3 모바일 2.0(이하 V3 모바일)이 글로벌 보안제품 평가 기관인 AV-TEST(www.av-test.org)가 지난 5월 실시한 테스트에서 인증을 획득했다.

국내 기업 중 유일하게 참가한 안랩은 지난 1월에 이어 연속 3회 인증을 획득하는 쾌거를 달성했다. 서구권 위주의 악성코드 샘플이라는 핸디캡에도 불구하고 높은 진단율을 기록했으며, 성능 부분에서는 만점을 받는 등 모바일 분야에서도 글로벌 기술력을 인정받았다.

스마트엔진을 새롭게 탑재해 테스트에 참가한 V3 모바일은 총 13.0점 만점에 12.5점이라는 높은 점수로 인증을 획득했다(8.5점 이상 인증 통과). 악성코드 탐지율(Protection)에서 6.0 만점에 5.5점을 받았고, 제품 실행 시 단말기의 성능에 미치는 영향을 평가하는 CPU 점유율에서도 상위권 성적을 기록해 사용편의성(Usability) 부분에서 6.0 만점을 기록했다. 악성코드 탐지 외에도 도난 방지(Anti-Theft) 기능, 스팸 전화 및 문자 방지 기능 등의 부가적인 보안 기능에서도 추가 점수를 얻었다.

AV-TEST의 모바일 보안제품 평가는 올해 1월 새롭게 시작했다. 이번 테스트에는 글로벌 보안업체 30곳이 참여해 27개가 테스트를 통과했다.

안랩은 AV-TEST 이외에도 국내 백신업체 중 유일하게 순수 국산 엔진을 기반으로 VB100, AV 컴패러티브(AV Comparatives), ICSA, 체크마크(Checkmark) 등 글로벌 5대 인증을 잇따라 획득한 바 있다.



▲ V3 모바일 2.0

김홍선 안랩 대표는 "스마트폰 보편화로 모바일 보안 시장의 중요성이 날로 높아지고 있다. 이번 인증 획득은 안랩이 모바일 분야에서도 글로벌 수준의 기술을 갖추었음을 증명하는 것"이라며 "안랩은 국제 인증으로 증명된 높은 순수 국산 기술력을 바탕으로 국내·외 모바일 시장 공략을 강화할 것이다"라고 밝혔다.

안랩, '기업지배구조 우수기업 우수상' 수상

안랩이 지난 6월 21일 한국거래소에서 개최된 '2013 기업지배구조 우수기업' 시상식에서 우수상을 받았다.

기업지배구조 우수기업이란 한국기업지배구조원이 매년 모든 거래소 및 코스닥 상장 법인을 대상으로 지배구조와 사회 책임, 환경 등의 지속 가능 경영 능력을 평가해 선정하는 것이다. 안랩의 이번 수상은 지난 2008년 첫 수상 이래 올해로 5회째다.

안랩은 지난 2005년 CEO와 이사회의 역할을 분리해 견제와 균형이 가능한 지배구조를 갖춘 바 있다. 2006년 사외이사제를 도입했으며, 현재는 전체 이사회 구성원 중 사외이사의 비율이 60%에 달해 법적 구비 의무 비율인 25%를 훨씬 웃돌고 있다.

또한 안랩은 창립 이래 사회적 역할과 책임을 다하고 있다. 개인용 무료 백신 V3 Lite를 배포하고 국가적 사이버 재난이 발생할 때마다 솔선수범해 위기를 돌파해왔다. 청소년용 무료 IT 교육 프로그램인 'V스쿨'을 꾸준히 개최하고 있으며, 여러 대학과의 산학 협력을 통해 IT·보안 인재 양성에도 기여하고 있다.

안랩은 환경적으로도 건전하고 지속 가능한 발전을 도모하기 위해 노력하고 있다. 판교 사옥의 친환경 업무 환경을 조성하는 한편, IT 기반 에너지 관리 솔루션 운용 등으로 에너지 절감을 실천하고 있다.

김홍선 안랩 대표는 "안랩은 국내 1위 보안 업체로서 매출 같은 양적 성과 외에도 연구개발(R&D), 회계 투명성, 지배구조, 사회공헌(CSR), 복리후생, 환경경영 등 질적 관점에서도 지속적으로 노력해왔다"며 "앞으로도 한국형 글로벌 SW 기업의 모범 사례를 만들어 나가겠다"고 수상 소감을 밝혔다.

▲ 안랩은 지난달 '2013 기업지배구조 우수기업' 시상식에서 우수상을 수상했다.

앞서 안랩은 모범적인 투명 경영을 인정받아 지난 5월 한국회계학회로부터 '투명회계대상'을 2회째 수상한 바 있다. 또한 '제 11회 경제정의기업상(경실련, 공정거래위, 2002년)', '제 1회 한국윤리경영대상–투명경영 부문 대상(신산업경영원, 산업자원부, 2003년)', '제 1회 경영정보대상–투명경영 부문 대상(한국회계정보학회, 2003)' 등을 받기도 했다.

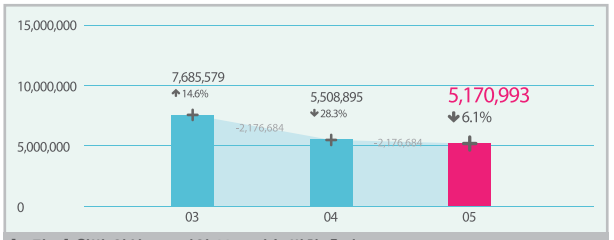
ASEC, 5월 통계 및 보안이슈 발표

5월 악성코드, 트로이목마 과반 이상 차지

안랩 시큐리티대응센터(ASEC)는 지난 2013년 5월의 보안 통계 및 이슈를 전했다. 5월의 주요 보안 이슈를 살펴본다.

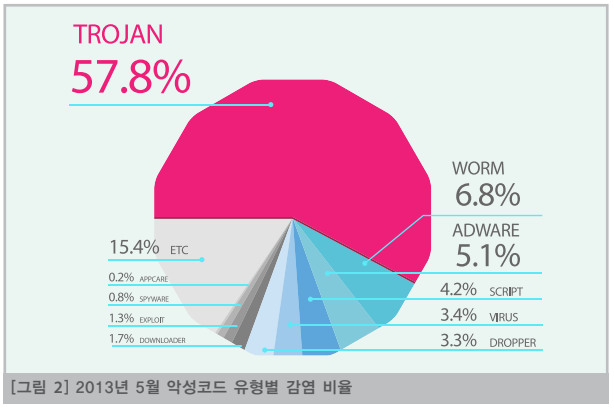
5월 악성코드, 전월 대비 33만여 건 감소

ASEC이 집계한 바에 따르면, 2013년 5월에 감염이 보고된 악성코드는 517만 993건이었다. 이는 전월 550만 8895건에 비해 33만 7902건이 감소한 수치다.



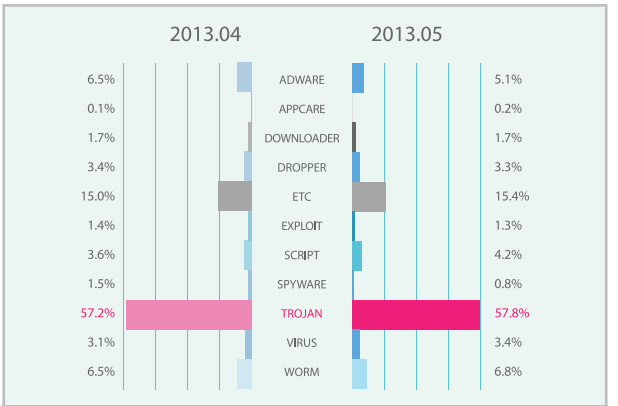
[그림 1] 월별 악성코드 감염 보고 건수 변화 추이

악성코드를 유형별로 살펴보면 트로이목마가 57.8%로 가장 높은 비율을 나타냈고, 웜과 애드웨어는 각각 6.8%, 5.1%의 비율을 차지하는 것으로 집계됐다. [그림 2]는 2013년 5월, 1개월 동안 안랩 고객으로부터 감염이 보고된 악성코드의 유형별 집계 결과다.



[그림 2] 2013년 5월 악성코드 유형별 감염 비율

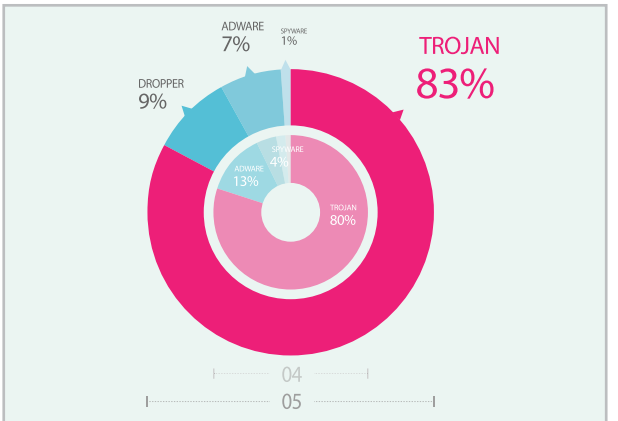
악성코드의 유형별 감염 비율을 지난 4월과 비교하면 [그림 3]과 같다. 트로이목마, 웜, 스크립트, 바이러스, 애프케어는 전월에 비해 소폭 증가했고 애드웨어, 드롭퍼, 익스플로이트, 스파이웨어는 감소했다. 다운로더 계열은 전월 수준을 유지했다.



[그림 3] 2013년 4월 vs. 2013년 5월 악성코드 유형별 비율

5월 최다 신종 악성코드 역시 트로이목마

5월에 신규로 접수된 악성코드 중 감염 보고가 가장 많았던 20건을 살펴본 결과, 트로이목마(Trojan/Downloader.205360)가 7만 6360건으로 전체의 39%를 차지해 1위를 기록했다.



[그림 4] 신종 악성코드 유형별 분포

신종 악성코드를 유형별로 살펴보면 트로이목마가 83%로 가장 많았고 드롭퍼는 9%, 애드웨어는 7%로 각각 집계됐다.

■ 시스템 복원 기능을 가진 '좀비 악성코드'

악성코드를 치료해도 시스템을 재부팅하고 나면 다시 악성코드가 나타난다?

최근 기존에 없던 기능이 포함된 악성코드가 확인됐다. 시스템을 복원하는 기능, 일명 '롤백(Rollback)' 증상이 있는 악성코드가 발견된 것이다.

해당 악성코드가 최종적으로 감염시키는 악성코드는 온라인게임핵으로, 이는 특정 온라인게임의 사용자 계정 탈취를 목적으로 하고 있다.

이 악성코드는 지속적인 감염 상태를 유지하기 위해 시스템 복원 기능을 악용하고 있어 치료에도 어려움이 있다. 치료를 한 뒤 재부팅을 하고 나면 다시 악성코드가 나타나기 때문이다.

롤백 기능이 있는 해당 악성코드에 감염되면 아래와 같은 파일이 생성된다.

[생성되는 파일]

C:\WINDOWS\system32\wininit[영문자].exe(Dropper)

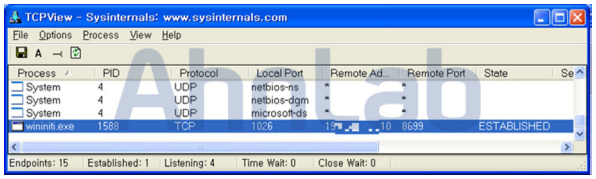
C:\WINDOWS\system32\Black.dll

C:\WINDOWS\system32\RCX[영숫자].tmp

C:\WINDOWS\system32\Drivers\diskflt.sys

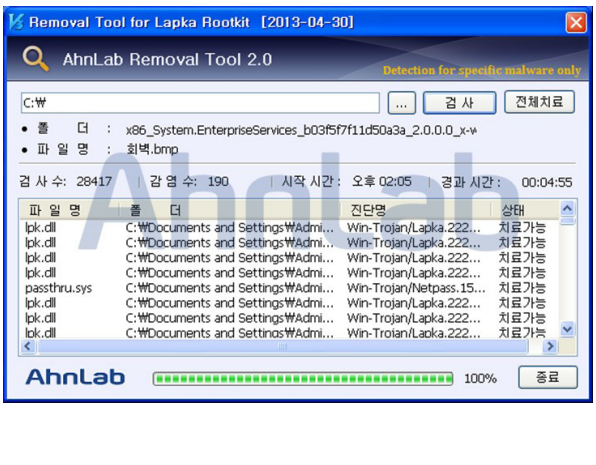
C:\WINDOWS\system32\drivers\passthru.sys

또한 black.dll의 복사본이 악성 lpk.dll 이름으로 모든 폴더에 생성되며, [그림 5]와 같이 특정 IP주소로의 네트워크 연결이 확인된다.



[그림 5] 네트워크 연결 정보

해당 악성코드는 V3를 통해 진단이 가능하다.



[그림 6] 전용 백신 실행 화면

안랩은 변종 악성코드가 끊임없이 나타나고 있다는 점에서 지속적으로 전용백신 및 V3 업데이트 등으로 대응하고 있다.

■ 사서함 용량이 초과됐습니다!

정상적인 메일로 위장해 악성코드를 유포하는 사례가 지속적으로 발생하고 있다.

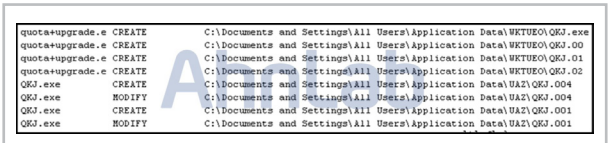
최근 국내에서 메일 사서함 용량이 초과됐다는 내용으로 수신된 메일이 의심스럽다는 문의가 접수됐다. 해당 메일은 용량을 늘리기 전까지는 메일을 수·발신할 수 없다며, '메일박스 업그레이드'라고 돼 있는 링크를 클릭해 파일을 다운로드하도록 유도했다.

[그림 7] 수신된 메일 원문

링크를 통해 다운로드받은 파일은 인터넷 익스플로러(Internet Explorer) 아이콘으로 위장하고 있으며, 해당 파일을 실행하면 악성코드에 감염된다.

[그림 8] 링크를 통해 다운로드한 파일

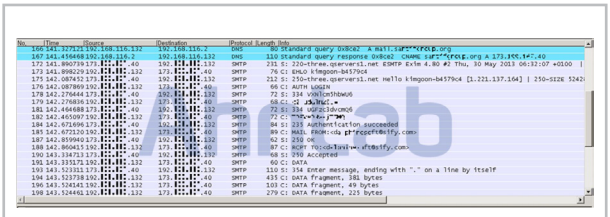
다운로드한 파일을 실행하면 [그림 9]와 같은 파일이 생성된다.



[그림 9] 생성되는 파일

또한 시스템 재시작 시에도 동작할 수 있도록 레지스트리 값을 등록한다.

해당 악성코드는 사용자의 시스템에서 실행된 프로그램 정보, 접속한 웹 정보, 키로깅 정보, 스크린샷 등의 정보를 수집한 뒤 악성코드 제작자가 생성한 것으로 보이는 특정 메일로 정보를 전송한다.

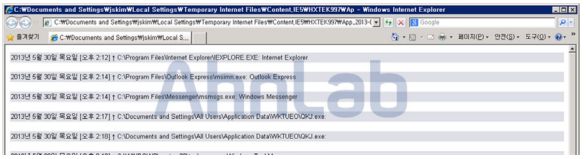


[그림 10] 특정 메일 서버를 통한 메일 발송 패킷

[그림 11]은 분석 당시 전송된 메일 내용으로, 수집된 정보가 HTML 형태의 파일로 첨부된 것을 확인할 수 있다.

[그림 11] 수집된 정보가 첨부되어 발송된 메일 원문

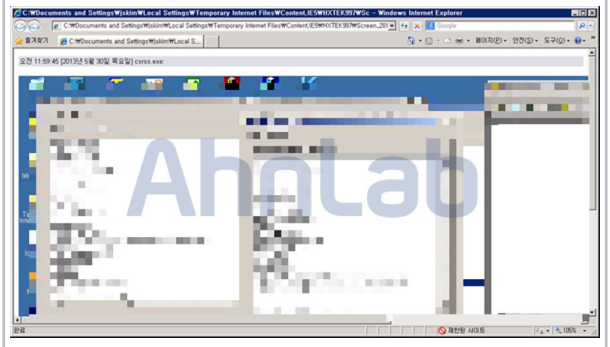
첨부된 내용은 다음과 같다.



[그림 12] 실행 중인 프로그램 정보

[그림 13] 웹 접속 정보

[그림 14] 키로깅 정보



[그림 15] 화면 캡쳐 정보

이러한 형태의 악성코드에 의해 수집된 정보는 지능형 지속 보안 위협(APT·Advanced Persistent Threat) 공격에 악용될 수 있어 각별한 주의가 필요하다. 이는 사용자가 속한 조직에 더 큰 보안 위협을 가져올 수 있기 때문이다.

해당 악성코드는 V3 제품을 통해 진단 및 치료가 가능하다.

발행인 : 김홍선
발행처 : 주식회사 안랩
경기도 성남시 분당구 삼평동 673
T. 031-722-8000 F. 031-722-8901
편집인 : 안랩 세일즈마케팅팀
디자인 : 안랩 UX디자인팀

http://www.ahnlab.com

http://blog.ahnlab.com

http://twitter.com/ahnlab_man

AhnLab

경기도 성남시 분당구 삼평동 673

T. 031-722-8000 F. 031-722-8901