리버싱 윈도우

한주성 지음

이것이 알고싶다
특별판

리버싱 윈도우 특별판 〈이것이 알고 싶다〉 Ver. 2013.08

© 한주성 & 로드북
이 〈특별판〉의 저작권은 한주성과 로드북에 있습니다.
자유롭게 배포 및 출력을 할 수 있으며 수정 및 발췌하여 재배포할 때는 출처를 명시해주시기 바랍니다.

인터넷 서점 바로가기 [예스24] [교보문고] [알라딘] [인터파크] [11번가] [반디앤루니스]

이메일 chief@roadbook.co.kr
블로그 www.roadbook.co.kr

1장 부팅의 시작: MBR (실습)

Q: 1장에서는 MBR 영역을 코드로 들여다보는 부분이 나오는데요. 사실, 간략하게 주석 등이 나오기는 하지만 초보자에게는 엄두가 나질 않을 것 같습니다. MBR 영역의 코드를 어느 정도 수준까지 들여다볼 수 있는 능력이 있어야 할까요?

A: 2부 9장의 어셈블리를 이해할 수 있는 수준이면 됩니다.

Q: 이 장에서 가상머신을 설치하고 운영체제를 설치하여 MBR을 들여다보고 있는데요. 실제 장애 발생이나 바이러스 침투 시 등에도 이런 식으로 부팅 과정에 어떤 문제가 있었는지를 살펴보나요?

A: 네, 분석 중 MBR과 관련이 있다고 판단하면 분석을 진행하게 됩니다. 단 장애로 판단될 때에는 MBR을 복원하는 작업을 진행할 것입니다.

Q: 앞으로 1장에서 구축한 실습환경으로 실습을 진행하나요?

A: 네. 1장에서 구축한 실습환경을 통해서 진행합니다. 하지만 다양한 가상화 환경과 분석 도구를 보여드리기 위해 차후에 더 많은 환경을 구축해보도록 할 것입니다.

Q: 1장에서 구축한 환경과 실제 현장 환경의 차이점이 있나요?

A: 차이는 없습니다. 다만 실제로 분석가들은 분석을 원활히 진행하기 위해서 이보다 더 다양한 도구와 더 많은 환경을 구축하여 사용합니다.

Q: 리버싱을 잘하기 위해서는 어떤 분야를 공부해야 하나요?

A: 보안 관련 분야를 공부하시기 바랍니다. 기술로는 어셈블리와 프로그램 개발을 함께 하시면 좋습니다. 그리고 커널 영역과 운영체제 만들기와 같은 임베디드 운영체제를 공부하셔도 도움이 됩니다.

Q: 저자님은 현업에서 어떤 일을 주로 하시나요? 그리고 리버싱과 어떤 관련이 있는지요?

A: 과거 보안 전문 업체와 기술 지원 업체에서 근무했으며, 현재는 시스템 관리와 보안 업무를 함께 하고 있습니다. 리버싱은 보안 업무와 프로그램 개발에 관련이 있습니다. 최근 IT 회사들은 시스템 관리보다 보안 업무가 차지하는 비중이 높습니다. 대다수 업무들이 보안과 맞물려 있다고 볼 수 있죠. 이에 저자는 업무 중 발생하는 침해 사고 해결에 리버싱 기술을 이용하여 빠르게 대응하며, 리버싱을 배울 때 익힌 프로그램 개발 능력을 통해 모니터링과 사용자 지원 도구를 개발하고 있습니다.

2장 Windbg 준비 (실습)

Q: 역분석을 진행하기 위해 Windbg를 소개해주셨는데요. 앞으로도 정말 많은 툴들이 나올 것 같습니다. 역분석을 위해 필수적으로 알아야 할 툴을 소개한다면 어떤 것이 있을까요?

A: Ollydbg, IDA, Windbg 정도가 필수적으로 알아야 할 도구입니다.

Q: Windbg로 분석하는 것과 IDA로 분석하는 것의 차이는 무엇인가요?

A: Windbg는 커널모드 분석에 특화된 도구라 할 수 있습니다. 그에 반해 IDA는 유저모드 분석에 특화된 도구로 일반 프로그램 분석 용도로 좋습니다. 분석의 차이점보다는, 효율적으로 분석할 수 있는 도구를 이용한다고 생각하시면 될 것입니다.

Q: Windbg는 2장에서 설명한 정도로는 많이 부족한 것 같습니다. 앞으로도 기본적인 사용법을 그때그때 설명을 해주나요? 아니면 따로 참고서적을 보아야 하나요?

A: Windbg는 2부에서 집중적으로 설명할 것이기 때문에 2장에서는 필요한 내용들만 다루었습니다. 만약 추가로 공부를 해보고 싶다면 〈Windows Internals〉를 추천합니다.

3장 시스템 시작: 커널 생성 (Ntos****.exe)

Q: 3장까지 보면 windbg가 아주 중요한 툴인 것 같습니다. 이 책을 학습하면서 함께 Windbg를 공부해보고자 하는데, 추천할 만한 학습 레퍼런스가 있는지요?

A: Windbg의 최고 레퍼런스는 Windbg 내 도움말이라고 할 수 있습니다. 도움말을 자주 이용하시기 바랍니다.

Q: 커널 초기화 관련하여 장애나 보안 관련 이슈 등의 에피소드가 있다면 간략하게 소개해주실 수 있는지요?

A: 커널이 디바이스 드라이버를 로드하다가 블루스크린이 발생하는 경우가 종종 있습니다. 이는 잘못 개발된 디바이스 드라이버 때문인데, 이때 메모리 덤프를 분석해 보면 어떤 디바이스 드라이버로 인해 부팅이 불가능한지 확인이 가능하여, 안전모드로 해당 디바이스 드라이버를 제외하고 부팅이 가능한 적이 있었습니다.

4장 커널과 유저의 소통: 시스템 프로세스

Q: 이 장에서는 커널과 유저의 소통을 해주는 시스템 프로세스를 배웠는데요. 어떻게 보면 시스템 프로세스는 수많은 명령어들로 구성되어 있는 것 같습니다. 이 명령어들에 익숙해지는 연습법이 있나요?

A: 실제 업무를 통해 익히는 것이 제일 빠르게 배울 수 있는 방법이라 할 수 있습니다. 이외에 스크립트 등을 통한 자동화 프로그램 작성에 이용하시면 명령어 사용에 익숙해지실 겁니다.

Q: 4장은 우리가 흔히 Alt+Control+Del 하면 볼 수 있는 프로세스에 관한 얘기인가요?

A: Alt+Control+Del을 지원하는 프로세스는 Winlogon입니다. 4장의 일부분입니다.

Q: 시스템 프로세스는 정확히 부팅 단계에서 어느 위치에 속하나요?

A: 커널 생성 이후 사용자를 지원해주기 위해 유저 모드에서 활동하는 프로세스들이라고 보면 됩니다.

Q: 수많은 시스템 프로세스를 하나하나 일일이 열거하면서 역할을 얘기해주었는데, 이것들을 외어야 하나요?

A: 외울 필요는 없습니다. Smss가 세션 관리자구나, 정도만 알고 있다가 나중에 찾아볼 수 있을 정도면 됩니다. 하지만 Smss를 이용하여 구성되거나 변경되는 정보들에 대해서도 알고 있어야겠죠. 이것을 알아야 필요할 때 써먹을 수 있으니깐요.

Q: 물론, 이해를 해야겠죠? 그런데, 어디에 쓰이는지, 왜 그런 프로세스를 설명하는지 잘 모르겠습니다. 수많은 옵션들이 있는데, 이것으로 무엇을 하라는 건지.

A: 레지스트리 및 Dll들의 값들은 해당 시스템 프로세스들의 환경을 구성하는 정보들로서, 시스템 오류나, 환경구성을 변경할 때 이용할 수 있는 값들입니다.

Q: 시스템 관리에 필요한건지, 아니면 운영체제가 어떤 문제가 발생되었을 때(보안이든, 성능이든) 파악하라는건지.

A: 시스템 관리적인 면이 크며, 보안(자동시작 프로그램)과 성능(메모리 구성과 같은 유저 환경 설정) 모두 관련되어 있습니다.

Q: 보통 보안 문제나 성능 문제는 서버 문제가 아닌지...그래서 클라이언트의 시스템 프로스세를 배우는 게 맞는지...

A: 시스템 프로세스는 동일하기 때문에 서버와 클라이언트에 대한 구분은 크게 중요하지 않습니다.

Q: 이 책은 다양한 윈도우 버전을 갖고 설명하고 있는데요. 버전별 큰 차이가 있을까요?

A: 윈도우의 버전별 차이에 따른 변화는 큰 뿌리는 변화하지 않습니다. 스레드와 프로세스를 생성하는 것처럼요, 거기에 부가적인 기능들이 업데이트되거나, 새로운 프로세스가 추가되는 정도(Wininit.exe)라 할 수 있습니다. 즉 메인 설계는 바뀌지 않게 됩니다. 물론 20~30년 전인, 윈도우 98, 윈도우 3.1과는 많이 다를 것입니다. 하지만, 현재 Xp부터 7까지의 기본 구조는 같습니다. 단지 덧가지가 붙은 정도입니다.

Q: 시스템 프로세스가 보안이나 장애 등의 실무에서 중요하게 활용된 예를 몇가지 소개해주실 수 있는지요?

A: 과거 고객 서버 장애에서 Smss의 메모리 레지스트리 설정을 통해 서버의 메모리 오류를 해결하였습니다.

Q: 게임이나 서버 프로그램에 문제가 생겼을 때도 이런 시스템 프로세스까지 연결되어 분석을 하는 건가요?

A: 메모리 문제와 바이러스 문제와 같은 커널과 내부 구조와 관련된 문제들이 발생하면, 시스템 프로세스에 대한 설정 역시 분석 범위에 포함되며, 많은 엔지니어들이 이부분을 통해 문제를 해결합니다.

6장 운영: 성능/관리 (실습)

Q: 이 장의 주제는 시스템 운영에 관한 내용인 것 같습니다. 운영을 잘하기 위해서는 윈도우 구조를 잘 이해해야 한다고 생각하지만, 윈도우 구조 부팅 과정을 학습하고 나서 운영에 관한 내용을 넣은 특별한 이유가 있는지요?

A: 운영을 잘한다는 것은 장애나 성능 부분에 대해 트러블 슈팅 하거나 개선을 잘 한다는 것과 같습니다. 이러한 작업은 아키텍처, 즉 구조적인 부분을 잘 알고 있어야만 진행할 수 있습니다. 그리고 윈도우 구조 중 부팅 과정은 윈도우 운영체제에서 중요한 대부분을 다루기 때문에 먼저 구조적인 면을 보여 드리고, 이후 응용적인 운영 부분을 보여드린 것입니다.

Q: 운영적인 면도 리버싱의 범주에 들어가나요?

A: 리버싱은 하나의 완성된 작업이라기보다 무언인가 완성하기 위해서 선행되는 한 부분이라 할 수 있습니다. 따라서 운영적인 부분도 리버싱 범주에 들어간다고 할 수 있습니다.

예를 들어 비유하자면 자동차 정비사가 운전을 못한다고 생각해 봅시다. 못할 수 있다고 생각하셨다면, 정비 후 테스트를 해주지 않을 것이며, 주행 중 소리와 조작으로 알 수 있는 부분은 이 분에게 기대할 수 없을 겁니다. 즉 정비사가 아닌 정비사가 되고 싶은 실습생에게 정비를 맡긴 것과 같습니다.

Q: 일반 기업의 전산 관리자가 6장 정도의 내용까지 알 필요가 있을까요? 특히나 메모리 덤프를 이용해서 블루스크린을 해결하는 부분이나 성능 관련 부분은 상당히 많은 지식을 요구할 것 같습니다.

A: 이 부분은 개인의 선택이겠지만, 기업에서 직장 생활은 성인 기준으로 보통 10년 이상을 해야 하는데, 10년 간 배워야 하는 지식 수준으로 많다고 생각되지 않으며, 오히려 관리자에서 책임자가 되었을 때, 숙련자의 모습을 보여줄 수 있는 좋은 고급 기술이라 생각됩니다.

Q: 바로 위 질문과 비슷한 질문일 수 있는데요. 시스템 운영을 제대로 하려면 프로그래밍도 잘해야 할 것 같습니다. 어느 정도까지 학습을 해야 할까요?

시스템 관리 용도의 프로그램을 직접 만들 수 있는 수준은 되어야 한다고 생각됩니다.

Q: 우리는 지금 윈도우 시스템만 다루고 있는데요. 리눅스나 유닉스 시스템 운영도 이 장에서 설명한 것과 비슷한 접근법을 갖나요?

A: 운영체제의 구조는 거의 동일하기 때문에, 윈도우 운영체제를 이해한다면, 리눅스나 유닉스도 쉽게 접근이 가능하실 겁니다.

7장 프로세스, 스레드 (실습)

Q: 운영체제는 집, 프로세스는 방, 스레드는 사람으로 비유해서 스레드의 중요성을 강조하였는데요. 이 비유는 운영체제와 프로세스는 정해진 구조이고 스레드는 시시각각 그 구조가 변화하는 동적인 구조라고 이해해도 되는 건가요?

A: 네, 비슷하다고 볼 수 있습니다. 조금 더 붙이자면, 모든 사람은 동일한 신체 구조를 가지고 있지만, 서로 다른 행동과 말, 결과를 낳게 됩니다.

Q: Windbg로 수많은 명령어들을 실행해보고 그 결과를 보았습니다. 물론 중요한 부분에 어떤 역할을 하는지 코멘트가 달려있기는 지금까지 배운 지식으로는 왜 이러한 명령어를 실행해보았고 이런 결과를 어디에 어떻게 써먹을지 감이 잘 오지 않습니다. 간단한 조언을 해주신다면?

A: 윈도우는 여러분들이 일반적으로 확인할 수 있는 이벤트 로그 이외에는 볼 수 있는 게 없을 것입니다. 이벤트 로그에서 프로세스가 오류로 중지되거나, 시스템이 이유 없이 재부팅될 때 Windbg를 이용하여 이 문제점을 확인해 볼 수 있습니다.

WIndbg를 가장 활용하기 용이한 것은 바로 메모리 덤프 분석이라 할 수 있겠네요. "서버에서 시스템 오류로 운영체제가 불안정할 때, 메모리 덤프로 오류 원인을 찾는다~", 멋지지 않나요?

메모리 덤프 분석은 프로그래머와 비교한다면 프로그램을 디버깅하여 버그를 찾는 거와 유사하다 할 수 있습니다. 하물며 그보다 더 방대한 양을 가진 운영체제를 분석한다는 것은 많은 경험이 필요한 만큼 매력적인 기술이라고 재차 강조하고 싶네요.

Q: 리버싱의 관점에서 아직도 프로세스와 스레드에 대해 왜 배워야 하는지 정확한 감이 오질 않습니다. 좀더 실제적인 사례를 들어주실 수 있는지요?

A: 프로세스와 스레드를 모르면 리버싱은 불가능합니다. 우리가 리버싱(역분석) 해야 하는 게 프로세스이고 스레드인데, 이를 모르고 분석을 어떻게 하나요?

경찰에게 시체만 주고 분석하라고 하는 거랑 비슷하다고 할 수 있습니다. 의사가 대장의 특성, 구조도 모르고 대장 내시경을 할 수는 없잖아요? 역분석 시 필요한 선수지식입니다.

Q: 프로세스와 스레드에 대해 좀도 깊이 학습할 수 있는 레퍼런스를 소개해주실 수 있는지요?

A: 프로세스와 스레드를 다루는 사이트는 아니지만, Windbg를 이용한 운영체제 분석에 도움이 되는 사이트를 소개해 드리겠습니다.

Windbg를 통한 다양한 분석 사례를 소개하는 사이트인 `http://analyze-v.com/`, 마이크로소프트 본사 플랫폼 팀 사이트인 `http://blogs.msdn.com/b/ntdebugging/`에서 유용한 정보들을 확인해 보실 수 있습니다.

8장 레지스터

Q: 8장은 레지스터에 대해 상당히 짧게 다룬 것 같습니다. 각 레지스터에 가장 빨리 익숙해질 수 있는 노하우가 있을까요?

A: 어셈블리 코드를 자주 확인하다 보면 레지스터는 자연스럽게 눈에 들어오게 됩니다. 책, 신문 읽듯이 Ollydbg, Windbg를 통해 자주 보시는 수밖에 없습니다.

Q: 저자님은 레지스터 관련시 시스템 운영이나 리버싱 관련한 에피소드가 있을 것 같은데, 한번 소개해주실 수 있는지요?

A: 회사 시스템이 장애가 발생하여 상용 프로그램을 이용해 복구를 진행하려 하였습니다. 하지만 정품이 아닌 상태에서는 제 기능이 동작하지 않고, 급히 복구할 필요가 있어 해당 프로그램을 역분석하여 정품 인증 우회을 시도하던 중 안티 디버깅으로 디버깅 레지스터를 사용하고 있는 걸 확인하였고, 디버깅 레지스터를 강제 변경하고 인증을 우회에 성공할 수 있었습니다. 물론 다음날 바로 구매했답니다. ^^

Q: 레지스터는 CPU 제작사마다 조금씩 다른 것 같습니다. 이런 부분까지 숙지를 하고 있어야 하는지요?

A: 숙지까지는 아니더라도 필요할 때 찾아보실 수 있도록 이해 정도만 하시면 됩니다.

9장 어셈블리 (실습)

Q: 이 장에서는 분석 도구로 Ollydbg를 이용하고 있는데요. 물론 중간중간 언급을 했지만, 어떤 때는 Windbg를 이용하고 어떤 때는 Ollydbg를 이용하는지 명쾌하게 정리해주실 수 있는지요?

A: 커널 디버깅, 메모리 디버깅 시에는 Windbg를 사용하고, 일반적인 프로그램 디버깅에는 Ollydbg를 사용합니다.

Q: 어셈블리는 CPU 제조사마다 언어가 많이 다르다고 알고 있습니다. 어떻게 학습하는 게 가장 좋을까요?

A: 각 제조사에서 제공하는 메뉴얼을 통해 학습하시는 게 좋습니다. 인텔의 경우 http://goo.gl/GubNT 에서 확인하실 수 있습니다.

Q: 사실, 어셈블리가 가장 기본적인 내용이라면, 이 책의 전반부에 나와야 했을 것 같은데, 중반부쯤에 등장한 특별한 이유가 있는지요?

A: 전반부에 나와도 큰 문제는 없지만, 책의 1부 내용은 부팅 과정을 이해하는 데 집중한 만큼 어셈블리와 같은 코드보다는 윈도우의 구조적인 면들을 보여드리고 싶어 2부 내용으로 넣게 되었습니다. 2부에서 보는 것이 더욱 집중도가 높을 것 같더라구요. ^^

Q: 저자님은 어떻게 어셈블리어를 학습하였는지, 간략한 학습 노하우 좀 들려주세요.

A: 책의 내용과 크게 다르지 않았습니다. 먼저 간단한 학습용 프로그램들을 다수 접한 후, 조금씩 확장해 나가는 방식으로, 점점 난이도 있는 프로그램들을 역분석 하시면 될 것 같습니다.

Q: 어셈블리 관련하여 따로 심층적으로 학습할 수 있는 레러런스를 소개해주세요.

A: http://tuts4you.com에서 어셈블리에 유용한 학습 콘텐츠를 찾으실 수 있을 것입니다.

10장 스택

Q: 주로 C++나 C# 언어로 만들어진 프로그램의 역분석 예제를 보여주고 있는데요. 예를 들어 파이썬이나 자바 등의 언어로 만들어진 프로그램의 경우 역분석은 어떻게 학습을 해야 할까요?

A: 파이썬이나 자바도 역분석이 가능합니다. 파이썬의 경우 기본 pdb라는 모듈을 이용해 가능하며

http://docs.python.org/2/library/pdb.html

자바는 개발 도구 이클립스(Eclipse)를 이용하여 가능합니다.

http://www.eclipse.org/resources/resource.php?id=503.

Q: 엉뚱한 질문일 수 있을 것 같은데요. 스택은 자료구조임과 동시에 레지스터의 하드웨어적 구조(?)인 것 같습니다. 레지스터에 저장하려면 무조건 스택 방식으로 저장해야 한다는 거죠. 그러나 자료구조는 무수히 많은데 레지스터 관련해서는 스택만 알면 되는 건가요?

A: CPU 레지스터가 사용하는 입출력 방식은 스택입니다.

Q: 이 책에서는 초간단 예제를 통해 아주 간단한 분석을 하고 있는 것 같습니다. 하지만, 현실 세계에서는 몇백 아니 수만 라인의 프로그램들을 역분석해야 할 것 같은데, 이때도 같은 방식으로 가능한가요? 다른 차이점이 있는 건지요?

A: 큰 차이점은 없습니다. 단지 흐름을 잃지 않도록 주석을 자주 달아주어야겠지요.

Q: 스택 관련하여 저자님의 에피소드가 있다면 한번 소개해주세요 장애나 침해사고 등등에 관련해서요.

A: 32비트 운영체제를 많이 사용하던 시절 커널 스택 오버플로우는 자주 발생하는 장애 중 하나였습니다. 32비트의 커널 메모리 구조에서 12Kb밖에 되지 않는 커널 스택 영역의 문제점으로 점차 운영체제가 64비트가 되면서 최근에는 보기 힘든 오류가 되었습니다. 커널 스택 오버플로우가 발생하면 블루 스크린과 함께 Bugcheck 0x7F가 발생하기 때문에 경험해보지 못했거나 운영체제 구조를 모른다면 문제 해결은 어렵겠지요. 문제점 해결을 위해서는 TIB에서 StackBase와 StackLimt 값을 확인하여 문제점이 발생한 스택을 확인할 수 있습니다. 다음 링크를 확인하면 도움이 될 것 같습니다.

http://msdn.microsoft.com/en-us/library/ff540620.aspx

11장 스레드 스케줄링

Q: 운영체제가 자동으로 해주는 스레드 스케줄링 영역에 과연 어떤 문제 때문에 사용자나 개발자가 개입하여 인위적으로 조정을 할 수 있나요?

A: 운영체제의 보다 유연한 사용을 위해서일 것입니다. 빠른 처리가 필요한 요청을 지원해주는 것이죠. 단, 전체 스케줄링을 인위적으로 조정하는 것이 아닌, 제한된 레벨 안에서 조정이 되는 것으로써 사용자 마음대로 바꿀 수는 없습니다.

Q: 스레드 스케줄링 관련하여 저자님의 에피소드가 있다면 한번 소개해주세요.

A: 서버에서 운영하는 서비스의 처리 속도 개선을 위하여, 프로세스에게 높은 우선 순위를 주어 보다 빠르게 처리할 수 있도록 조정해 주었는데 이를 더 빠르게 처리할 수 있을까 하는 생각에 실시간으로 변경해 보았습니다. 실시간으로 변경한 지 몇분 후 서버의 잦은 무응답 증상이 발생하면서 문제를 일으켜 높은 우선 순위로 되돌린 경험이 있습니다. 이처럼 임의로 사용자가 프로세스의 권한을 올리게 되면 해당 프로세스는 빠르게 처리할 수 있지만, 함께 처리되어야 하는 다른 프로세스의 스레드들의 우선순위가 밀려 운영체제가 불안정해질 수 있습니다.

Q: 리버싱에서 스레드 스케줄링은 어떤 분야에 도움이 되나요? 예를 들어, 장애 진단이나 기타 성능 문제 등등?

A: 성능 문제에 도움이 된다고 할 수 있습니다. 하지만 스레드 스케줄링을 리버싱할 일은 많이 없을 것입니다. 운영체제의 개념으로 알고 계시고, 시스템 프로그램 개발 시 이용하시면 좋을 듯합니다.

Q: 사용자나 관리자는 운영체제의 스레드 스케줄링에 어느 정도 개입하는 것이 적당한가요?

A: 되도록 개입하지 않는 것이 좋습니다. 스레드 스케줄링은 개발 시 이용하는 것이 좋습니다. 사용자와 관리자는 이러한 방식을 알고 개발에 요청하는 것이 가장 바람직합니다. 만약 필요하다면 Xperf 등을 통한 성능 분석 후 스레드 스케줄링을 조절하여야 할 것입니다.

12장 퀀텀

Q: 퀀텀 값은 주로 시스템의 성능과 관련되어 있는 건가요?

A: 네, 시스템 성능에 영향을 줍니다.

Q: 역시 이 장에서도 퀀텀과 관련된 에피소드가 있다면 한번 소개해주실 수 있는지요?

A: 과거 하드웨어를 도입할 때 PC형 모델이 서버형 모델보다 빠르다며 PC형 모델로 도입해 줄 것을 요청하는 실무자가 있었습니다. 하지만 이는 빠르게 컨텍스트 전환을 진행하는 클라이언트 운영체제인 윈도우 7의 퀀텀값과 관계가 있음을 설명한 적이 있는데요, 일반적으로 하나의 서비스에 집중해야 하는 서버 운영체제의 경우 잦은 컨텍스트 전환은 성능 하락으로 이어질 수 있기 때문에 기본적으로 긴 퀀텀값으로 설정되어 있습니다.

따라서 이용 목적에 맞게 구성하는 게 좋습니다. 여러 프로세스를 실행하는 서버는 퀀텀값을 줄이는 것이 좋고 단일 프로세스를 실행하는 환경이라면 퀀텀값을 늘리는 것이 효과적입니다.

추가로 서버형 모델을 이용해야 하는 또 다른 이유는 장시간 서비스 유지를 위한 안정성도 있답니다

Q: 퀀텀은 실행 시간이라고 했는데, 클럭 또한 같은 실행 시간인데 그의 3배수인 퀀텀이라는 용어를 만들었을까요?

A: 퀀텀은 운영체제가 프로세스에 CPU의 사용시간을 할당하기 위한 기준입니다. 정확한 이유를 알 수는 없지만, 예전 운영체제와 통일성을 위해서 계속 같은 값으로 사용하는 것 같습니다. 그리고 이 값은 운영체제에서 정의한 것으로 새로운 운영체제가 나오면 변경될 수 있을 듯합니다.

Q: 퀀텀값 또한 스레드 우선순위처럼 운영체제에서 자동으로 조정되는 건가요? 그렇다면 인위적으로 퀀텀값을 조정해야 할 때는 어떤 때인가요? 이 장에서 일부 소개가 되기는 했지만 추가해주신다면?

A: 퀀텀값은 무한 대기에 빠지는 것을 막기 위해 자동으로 조정됩니다. 그리고 퀀텀값 조정은 머신의 성능을 최적화할 때 이용하시면 됩니다. 실행할 프로세스는 적은데, 컨텍스트 전환이 자주 일어난다면 퀀텀값을 늘리는 게 좋습니다. 반대로 여러 프로세스를 사용한다면 퀀텀값을 줄여주는 것이 좋을 것입니다.

13장 디스패칭

Q: 스케줄링이 있는데 왜 디스패칭이 별도로 필요한지 아직도 잘 이해가 안 됩니다.

A: 스케줄링은 스레드 실행에 대한 우선 순위를 지정하지만 예외 상황은 관여하지 않습니다. 디스패칭은 예외 상황을 처리할 수 있는 수단으로서, 먼저 처리할 필요가 있는 요청을 바로 처리해 주기 위해 필요하다고 생각하시면 될 것 같습니다. 운영체제에서 여러 프로세스를 실행하고 있을 때, 키보드 입력이 들어 왔다면 키보드 입력을 우선적으로 처리해 주어야 할 것입니다. 만약 스레드 스케줄링만 존재한다면 현재 처리 중인 여러 스레드의 실행 시간까지 다 사용하고 처리되게 되므로, 키보드 입력은 바로 나타나지 않게 될 것이며 사용자는 운영체제에게 분노를 표출할 겁니다.

Q: 디스패칭과 관련된 에피소드가 있다면 한번 소개해주실 수 있는지요?

A: 디스패칭은 선점과 관련이 있기 때문에 이와 관련된 장애가 자주 발생하곤 합니다. 블루스크린의 버그 코드로는 0xA로 권한이 없는 메모리 영역에 접근했거나, IRQL 문제로 발생하게 됩니다.

이 문제는 대부분 잘못된 드라이버 개발로 발생하며 드라이버를 업그레이드 혹은 다운그레이드할 것을 권장하는데, 해당 드라이버를 사용하지 않는 것도 해결 방안일 수 있습니다.

Q: 디스패칭은 장애 해결 능력을 키우기 위해서는 알아야 하는 것 같습니다. 하지만 침해사고 관련 등의 해킹에서도 이 부분이 필요하나요? 필요하다면 어떤 부분이 도움이 될까요?

A: 해킹에서는 IDT를 수정하여 조작된 함수를 호출하도록 할 수 있으므로 알아두시면 좋습니다. IDT Hooking과 관련된 자료는 인터넷 상에서 쉽게 찾아보실 수 있으실 겁니다.

Q: 위와 비슷한 질문일 수 있지만 디스패칭에 익숙해지면 어떤 능력을 기를 수 있을까요?

A: 침해사고 대응, 해킹 등 악성코드 분석/대응 능력이 좋아질 것으로 보이며, 시스템 장애에서도 좋은 판단 능력이 생길 것이라 판단됩니다.

Q: 디스패칭 관련 참고할 수 있는 레퍼런스가 있다면 소개해주세요.

A: http://blogs.technet.com/b/marcelofartura/와 한국 마이크로소프트 고객 기술 지원부 플랫폼팀 블로그인 http://blogs.msdn.com/b/kocoreinternals 에서 좋은 정보를 확인하실 수 있습니다.

14장 메모리 관리

Q: 장애가 일어날 시에 윈도우 시스템의 경우 복구 기능을 잘 제공해주고 있는 것 같습니다. 리버싱을 통해 장애를 해결할 수도 있고 이렇게 제공되는 복구 시스템을 이용해서 할 수 있을 것 같은데 실무에서는 그 비중이 어떻게 되나요?

A: 서비스 중요도에 따라 다르지만 리버싱을 통해 해결하는 경우가 많습니다. 복구 시스템을 이용하게 되면 빠르게 복구는 가능하지만 문제가 재발될 가능성이 높다고 할 수 있습니다. 따라서 일정 규모 이상의 기업들은 마이크로소프트 기술 지원 부서와 연계하여 리버싱을 통해 원천 문제점을 해결하고자 노력하고 있습니다.

Q: 메모리 관리는 성능과 많은 관련이 있을 것 같은데, 메모리 누수를 사후에 찾아내는 방법과 사전에 예방하는 방법이 있을 듯합니다. 현업에서 시스템 운영 관리 시에 이런 두 가지 경우는 어떤 경우일까요?

A: 메모리 누수는 대부분 잘못된 드라이버, 프로세스 문제가 많기 때문에 사전에 적용하기가 어렵습니다. 대부분 사후 처리로 진행하게 됩니다.

Q: 메모리 관리는 어떻게 보면 프로그램의 자료구조의 영향을 많이 받는 것 같습니다. 잘못된 자료구조로 인한 메모리 관리의 실패 사례를 소개해주실 수 있는지요?

A: 메모리 관리 실패는 인자의 크기를 지정하지 않는다거나 검증 작업을 하지 않을 때 많이 발생하게 됩니다. 가장 쉽게 접할 수 있는 것이 바로 버퍼 오버런 관련 오류로서 지정된 버퍼 이상을 사용해 반환 주소를 덮어쓰는 오류인데, 이러한 오류를 최소화 하기 위해서 비주얼 스튜디오 컴파일러에서는 컴파일 시 /GS 옵션을 이용하여 개발 단계에서 검사할 수 있습니다.

Q: 메모리는 윈도우 리버싱에서 어떤 부분에 중요할까요?

A: 메모리는 운영체제가 사용한 흔적을 확인할 수 있는 매우 중요한 부분입니다. 장애 분석인 메모리 덤프 분석도 그렇고, 차후 3부에서 나오는 침해 사고 분석에도 중요하게 이용됩니다.

Q: 이 장에서 소개한 기법과 관련한 에피소드가 있다면 소개해주세요.

A: 과거 32비트 운영체제가 많이 사용되던 시절, 유저모드의 메모리 공간을 늘리기 위해 1부에서 소개한 Boot.ini를 조작해 유저모드 메모리 공간을 3GB로 설정하여 서버를 운영하는 경우가 종종 있었습니다.

32비트 운영체제는 최대 메모리가 4GB이기 때문에 이처럼 유저모드 메모리 공간을 늘리면 상대적으로 커널 모드 메모리 공간은 줄어들게 되고, 이 중 PTE가 부족한 상황이 발생하여 블루 스크린이 발생하는 상황도 많이 있었답니다. 다음 사이트를 참고하면 PTE를 이해하는 데 더욱 도움이 되실 겁니다.

http://www.microsoft.com/en-us/download/confirmation.aspx?id=12519

15장 I/O Request

Q: 입출력은 보안 관련해서는 상당히 중요한 부분인 것 같은데, 상당히짧게 다루는 것 같습니다. 특별한 이유가 있는지요?

A: 입출력 부분은 드라이버 개발과 관련된 부분이 많습니다. 따라서 이 책에서 많은 내용을 다루기에는 무리가 있다고 판단하였고, 리버싱을 진행할 때 알아야 하는 부분 위주로 정리하였습니다. 하지만 질문하신 것과 같이 보안 관련하여 중요하다 보니 보안 관련하여 4부에서 다시 한번 등장합니다.

Q: 이 장에서는 입출력 관련 개체들에 대해서만 어떻게 생겼는지 정도를 알려주고 있는 것 같습니다. 이러한 개체들을 어떻게 이용해서 해킹이 이루어지는지 간략한 예 한 가지만 소개해 주실 수 있는지요?

A: 4부에서 다루는 내용인데, 간략하게 말씀 드리겠습니다. 안티 바이러스 프로그램의 필터 드라이버 처럼 디바이스 드라이버로 설치가 되어, 입출력을 진행할 때 먼저 확인하여 처리하는 식으로 해킹을 진행합니다. 즉 파일시스템 드라이버가 처리하기 전에 자신 먼저 처리하는 것이죠. 이 부분은 4부에서도 다루고 있을 정도로 유명한 해킹 방식이라 할 수 있습니다. 인터넷 검색으로는 IRP 후킹으로 확인해 보실 수 있습니다.

Q: 입출력 관련해서 곤란을 겪었던 에피소드가 있다면 소개해주세요.

A: 시스템 운영을 해보신 분이라면 아실 수 있는 내용으로 안티 바이러스 프로그램은 대부분 커널 필터 드라이버로 등록되어 파일을 쓰거나 읽을 때 먼저 검사를 진행하게 됩니다. 그런데 블루스크린으로 시스템에 장애가 발생하고 이를 확인해 보면 대부분 안티 바이러스 프로그램에서 파일을 검사하다가 발생하는 경우가 많다 보니 저 역시 서버에서 블루스크린이 나면 우선적으로 안티 바이러스를 의심하게 되었습니다.

Q: 입출력까지가 윈도우 내부구조를 이해하는 데 전부는 아닐 것 같습니다. 내부구조에 대해 좀더 심층적으로 학습하고 싶다면 어떻게 해야 할까요?

A: 이 책이 윈도우 내부 구조를 마스터할 수 있는 책은 아닙니다. 저자마다 보는 방향이 다르기 때문에 여러 책을 통해 공부해 보시기를 추천합니다. 〈실전 윈도우 디버깅〉, 〈Windows Internals〉를 추천합니다.

Q: 입출력 관련하여 좀더 심층 학습을 할 수 있는 레퍼런스가 있다면 소개해주세요.

A: http://www.driveronline.org/ 이나 12장에서 추천 드렸던 마이트로소프트 내부 팀 블로그나 엔지니어 블로그를 이용하시면 도움이 되실 듯합니다.

16장 인지의 중요성

Q: 해킹은 툴 사용법과 침입 매뉴얼만 알아도 어느 정도 진행할 수 있는 것 같습니다. 어떻게 보면 공격보다 방어가 더 어려운 불공평한 게임인 듯 보이는데요. 그렇지 않나요?

A: 네, 방어는 어렵습니다. 보안은 완벽하다고 할 수 없습니다. 만약 보안 담당자에게 "보안 상태는 안전합니까?"라고 물었을 때, "안전하다"고 대답했다면 오히려 취약한 상태일 것입니다. 항상 안전한지 확인하고 알 수 없는 공격에 대한 준비를 통해 대비하여야 합니다.

Q: 시스템 관리자나 리버서들도 하루에도 수없이 쏟아져 나오는 해킹 툴이나 사례 등을 지속적으로 살펴보고 학습해야 하나요?

A: 어느 정도의 학습은 지속적으로 필요합니다. 이는 반복 학습을 통한 심화 학습일 수도 있으며, 새로운 영역을 통한 기술 확장일 수도 있습니다. 하지만 학습하지 않는다면 사용하지 않는 철이 녹슬듯이 점점 본인의 기술도 퇴화할 것 입니다.

Q: 저자님이 겪었던 가장 기억에 남는 해킹 피해 사례가 있다면 소개해주세요.

A: 웹 방화벽을 도입한 지 얼마 되지 않은 시기에 로그를 확인하던 중 "la -a"라는 명령이 웹 서버의 이미지 경로에서 실행하려 했던 흔적을 확인할 수 있었고 해당 경로를 확인해보니 웹 쉘이 동작하고 있는 것을 찾아낸 적이 있었습니다.

대부분 국내 기업의 경우 관리보다는 개발 중심적이다 보니 사용하지 않은 수많은 소스를 지우지 않고 계속 웹 서버에 보관하거나 보안성을 검증하지 않은 취약한 소스로 인해 발생하는 보안 사고가 많습니다.

Q: 저자님이 겪었던 기억하기 싫은 장애 사례가 있다면 소개해주세요.

A: 기억하기 싫은 장애는 없는 것 같습니다. 다만 장애에 대한 접근법이 시간이 지날수록 장애 현상 해결보다 재발 방지를 더욱 고민하게 됩니다. 이를 위해 복구, 성능, 백업, 구조 개선 방안 등으로 작업을 계획하게 되는데, 이런 근본적인 개선 작업을 통해 새로운 기술을 접할 수 있으며, 이용할 수 있는 기술 영역 역시 다양하여 더 많은 실력을 쌓을 수 있다고 생각됩니다.

17장 역분석의 묘미: 프로그램 역분석 (실습)

Q: 바이러스 백신 제작자가 아닌 시스템 엔지니어에게 프로그램 역분석 능력이 왜 필요한지 필요하다면 어느 정도의 지식이 필요할까요?

A: 프로그램의 전체적인 흐름을 파악할 수 있는 정도면 됩니다. "이런 함수를 호출해서 이렇게 됐구나" 정도랄까요.

Q: 디컴파일러의 개념을 보면, 이미 만들어진 프로그램의 소스코드를 뒤져본다는 의미에서 저작권 문제 등 많은 부분에서 문제나 악용될 소지가 있어 보이는데요?

역분석 행위는 소프트웨어 저작권 문제가 있는 만큼 저작권이 있는 소프트웨어는 역분석 행위가 불법으로 역분석을 진행해서는 안 됩니다. 만약 불가항력(?)에 의해 진행한 경우라면 외부 공개나 유출은 하지 않아야 합니다.

Q: 프로그램 역분석 경험 사례를 좀 들려주세요.

A: 역분석을 몇번 진행하다 보니 내 프로그램도 쉽게 남들이 훔쳐볼 수 있다는 생각에 "조잡한 코드가 아닌가" 하며 더욱 신경 쓰게 되더군요. 그리고 역분석을 할 수 없도록 프로그램을 패킹(4부에 나옵니다)도 하게 되고요. 저 역시 몇몇 패커의 경우 역분석을 하다가 포기한 경험이 있는 만큼 패킹은 역분석을 방지하는 좋은 수단이라 할 수 있습니다.

Q: 프로그램 역분석을 진행하기 위해서는 윈도우에서 기본으로 제공되는 수많은 API의 함수나 구조체에 상당히 익숙해야 할 것 같습니다. 빠른 학습 방법이 있다면 소개해주세요.

A: API를 자주 확인해 보는 것이 좋습니다. 마이크로소프트에서 제공하는 개발자 사이트인 http://msdn.microsoft.com 에서 검색을 통해 쉽게 API를 확인하실 수 있습니다.

Q: 프로그램 동적 분석의 경우에는 악성 프로그램일 가능성 때문에 가상화 머신을 활용하는 것 같습니다. 보통 역분석 진행시에 가상화 머신은 어떤 형태로 준비하나요?

A: 가상화 머신으로 분석을 진행하기 전에 현재 가상화 머신의 상태를 스냅샷과 같은 기능을 이용해 저장하게 됩니다. 이는 역분석을 진행하다 다시 되돌아 가고자 할 때 많이 이용됩니다. 그리고 스냅샷은 대부분의 가상화 프로그램에서 지원합니다.

Q: 역분석을 좀더 심층적으로 학습하고 싶은데요. 어떤 것을 더 보완해서 공부해야 할까요?

A: 안티 디버깅과 인라인 패치와 같은 역분석 상에서의 데이터 조작을 진행하시면 도움이 될 것 같습니다. 이와 관련해서는 4부에서 진행합니다.

18장 증거 확보/분석을 위한 데이터 수집 (실습)

Q: 데이터 수집에서 도구가 상당히 중요한 것 같습니다. 새로운 도구도 많이 있는 것 같고요. 지속적으로 새로운 도구가 무엇이 있는지 찾아보고 학습을 해야 하나요?

A: 지속적으로 새로운 도구를 찾는 것보다 자신에게 잘 맞는 도구를 더욱 잘 다룰 수 있게 하는 것이 더 좋습니다.

Q: 장애나 해킹 의심 사고 발생 시 어떤 표준적인 프로세스가 있을 것 같은데요?

기본적으로 동일하게 진행하는 부분은 데이터 수집이라고 할 수 있습니다. 수집 도구를 이용하여 데이터를 수집하고 어떻게 발생된 것인지 분석을 진행하게 됩니다.

Q: 표준 프로세스 외에 리버서의 촉(또는 감이나 경력)에 의해 해결이 될 수도 있을 것 같은데, 표준 프로세스와 리버서의 촉, 이 둘 중 어떤 게 중요한가요? 물론 촉이라는 것도 경험에 기반되어야겠지만요.

A: 아무래도 분석가의 경험이 더욱 빠른 해결을 할 수 있지만, 예측만하고 검증을 하지 않으면 오판할 경우가 많은 것이 이 분야입니다. 경험을 바탕으로 검증을 진행하는 표준 프로세스가 중요합니다.

Q: 이 장에서는 거의 툴 위주로 데이터 수집을 하는 것 같습니다. 그리고 상당히 복잡한 절차를 거치는 것 같습니다. 그래도 데이터 수집 시에 가장 중요한 원칙 등이 있을 것 같습니다. 무엇이 있을까요?

사고 당시 데이터를 최대한 많이 수집하는 것과 손실 없는 수집이라 할 수 있습니다. 이를 통해 분석가는 단서를 찾을 수 있고, 찾은 단서가 올바른 정보가 될 수 있습니다.

Q: 데이터 수집은 평상 시에도 꾸준히 해야 하나요?

A: 데이터 수집과 분석은 자주 연습해 볼 필요가 있고, 외우기보다는 왜 수집하는지를 이해하는 게 중요합니다.

Q: 데이터 수집 관련한 경험 사례가 있다면 공유해주세요.

A: 예전에 해킹 사고 발생으로 데이터 수집 중 명령 옵션이 틀려서 중요 단서를 놓친 적이 있습니다. 그때는 피해 시스템에서 직접 명령을 수행하며 수집과 분석을 진행하였는데, 초보적인 실수로 발생한 일이었죠. 다행히 다시 수집하면서 유사 데이터를 수집할 수 있었지만, 사용자가 이미 관련 프로세스를 삭제하여 이전에 나왔던 데이터와 동일한 데이터를 얻을 수는 없었습니다.

이를 통해 수집 명령을 스크립트로 작성해 놓는 것의 중요함을 다시 한번 느꼈습니다. 꼭 필요한 데이터 수집 항목은 사전에 스크립트로 수집하시기 바랍니다.

19장 분석

Q: 이 장에서 리눅스 시스템을 다루는 특별한 이유가 있나요?

A: 리눅스는 GUI보다 CLI 기반 명령들이 발달한 운영체제 입니다. 그렇다 보니 리눅스 도구들은 숙련자들이 사용하기에 유용한 경우가 많습니다. 특히 분석에 필요한 검색 필터링과 결과 제어 능력이 탁월하여 몇몇 도구는 리눅스 도구를 이용하는 게 분석에 이롭기 때문에 리눅스 도구를 다루게 되었습니다. 그리고 웹 서버의 경우 리눅스로 운영하는 경우도 많아 침해사고 분석에 리눅스 역시 알아야 하는 영역이므로 리눅스 운영체제에 대한 분석 방법도 알아두시면 좋습니다(그러나 이 책은 윈도우 시스템을 다루므로 리눅스 시스템에 대해서는 거의 다루지 않습니다).

Q: 우리는 '파일 복구' 관련된 내용을 배웠는데요. 파일 복구도 리버싱의 한 영역인가요?

A: 네, 역분석에는 파일 복구도 들어갑니다. 역분석은 지난 일의 행위를 분석한다고 생각하면 됩니다. 즉 해당 사고 일에 어떤 파일이 지워졌나 알 필요가 있습니다.

Q: 언론매체에서 사건 사고를 보도하는 것을 보면 "하드디스크를 수거해서 검사하고 있다" 등이 기사를 심심찮게 보곤 하는데, 보통 이런 경우에도 이 책에서 제시하는 복구 절차를 거치나요?

A: 보다 전문적인 복구 시스템을 이용할 것이지만, 복구 절차는 유사합니다. 책에서 설명하였듯이 원천 제거 방법을 이용하면 복구는 어려울 것입니다.

Q: '파일 복구'와 관련하여 경험 사례가 있다면 공유해주세요.

A: 오래 전에 제약 회사에서 단기 아르바이트를 했는데, PC 교체 지원이었습니다. 지원 도중 연구원 한 분이 중요 파일을 지웠다며 발을 동동 구르고 있을 때 책에서 소개한 도구를 이용하여 복구해 드린 적이 있습니다. 고맙다며 식사도 대접해 주시고, 들어와서 함께 일할 생각 없냐고 묻는데, PC 지원 업무는 맞지 않는 것 같아서 거절한 기억이 나네요. 아마 거기서 일했다면 지금쯤 난 어땠을까 라는 생각이 듭니다. ^^

Q: 이 책에서 제시하는 것처럼 시스템에서 "완전 삭제"를 한다면 0%도 복구할 수 없나요?

A: 0% 이상 복구는 가능합니다. 디가우저와 같은 장비로 자성을 없애지 않는다면요, 단지 원하는 데이터가 아닌 무의미한 데이터이거나 대다수의 정보가 유실된 상태일 것입니다(복구 시스템에 따라 복구 정도는 달라집니다).

일부를 통해 예측이 가능할 수도 있습니다. 이름 혹은 일부분의 정보들을 통해서 이것일 가능성이 높다 라는 추측이 가능할 것입니다.

Q: 완전 삭제 기법이 일반인들에게 잘 공개되어 있지 않은 것 같습니다. 그 이유가 뭘까요?

A: 데이터를 복구해야 할 필요성은 많지 않습니다. 따라서 완전 삭제 기법 역시 일반인들에게는 불필요하게 되죠. 필요하지 않다면 모를 수밖에 없지 않을까요?

20장 증명할 수 있는 근거: 무결성

Q: 데이터 무결성을 유지하기 위해 수많은 해쉬 알고리즘이 존재하는데, 따로 학습을 해야 하나요? 어느 정도까지 학습을 해야 하는지요?

A: 해쉬 알고리즘의 특성까지는 알고 있는 게 좋습니다. 대칭키, 공개키 방식 등의 특성을 알면 필요시 알맞은 알고리즘을 선택하여 개발하거나 이용할 수 있을 것입니다.

Q: 해쉬값을 남기더라도 변조유무를 숨길 수는 없나요?

A: 해쉬값도 몇만 분의 확률로 동일 값이 나올 수 있다고 합니다. 하지만 이것은 확률일 뿐 해쉬값이 있다면 변조유무를 숨기는 건 어려울 것입니다. 변조를 숨기고자 한다면, 해쉬값 대조 작업을 조작하는 방식이 될 것입니다.

Q: 무결성과 관련한 경험 사례가 있다면 소개 부탁드립니다.

A: 서버 침해 사고가 발생하였는데, 파일이 변조되었다는 서버 침해를 의심하는 사고였습니다. 우리는 절대 아니라고 해도 고객 측에서는 서버 침해라고 관리 소홀이 아니냐고 계속 의심하는 상황이었죠. 그래서 파일이 서버에서 변경되지 않았다는 것을 증명하기 위해 점검 시 해쉬값을 저장해 두었다가 하루 한번씩 모든 파일의 해쉬값이 맞는지 비교하고 틀릴 경우 틀린 파일을 메일로 알려주는 프로그램을 작성한 적이 있습니다. 이를 통해 고객의 신뢰를 회복할 수 있었습니다.

21장 난 네가 한 일을 알고 있다: 사용자 추적

Q: 사용자 추적 관련 내용은 상당히 흥미로운 영역일 것 같은데, 간략하게 설명되었네요. 특별한 이유가 있는지요?

A: 기존 내용들을 계속 봐왔다면 어렵지 않은 내용일 거라 생각하여 간략히 소개하였습니다. 그리고 악성코드 해킹이 많아 사용자 영역보다 시스템 영역을 확인하는 작업이 중요합니다.

Q: 사용자 추적은 개인 사생활 보호 문제와 관련이 있을 듯합니다. 관련하여 조언해주실 수 있는 부분이 있는지요?

A: 사용자 영역 관련해서는 시스템 청소 프로그램(다음 클리너, 네이버 클리너 등)을 이용하여 어느 정도 정보 유출을 방지할 수 있습니다. 단 앞서 보았던 파일 복구 부분도 생각해서 완전 삭제를 함께 병행하면 더욱 좋습니다.

Q: 사용자 추적 관련 저자님의 경험 사례가 있다면 소개해주세요.

A: 실제 사고 사례에 사용한 적은 없지만, 좀비 PC에서 웹 사용 기록을 조회하여 사용자가 접근하지 않은 경로가 있는지 확인한 적이 있습니다. 간혹 해커가 좀비 PC에 직접 접근하여 사용하는 경우도 있기에 웹 히스토리 분석 시 함께 보게 됩니다.

22장 보초를 세우다: 감시

Q: 시스템 감시를 위해 보초를 세우는 가장 중요한 기준이나 원칙이 있다면 소개해주세요.

A: 가장 중요한 관점은 사용자가 알지 못한 작업이 시스템에서 진행되었는가입니다. 이 관점으로 바라보면, 설치 부분과 파일 접근 부분, 로그인 부분 등 사용자가 하지 않은 작업들이 있다면, 해킹이라고 간주할 수 있습니다.

Q: 보초를 세워도 그 보초가 무기력한 경우도 있을 것 같습니다. 어떤 경우가 있을까요?

A: 이미 해킹된 시스템에 보초를 세운 경우라 할 수 있습니다. 이 경우 보초를 세운다 해도 발견할 수 있는 가능성은 미비할 것입니다.

Q: 일반적으로 많이 사용되는 시스템 감시를 위한 보초의 형태(?)가 있을까요?

A: 가장 많이 사용되는 방식은 시스템 이벤트를 이용한 SMS 전송이라 할 수 있습니다. 시스템은 운영 중에 특수 상황을 만나면 설정된 이벤트를 발생하도록 구성되어 있어 이를 이용해 관리자에게 SMS로 실시간 알림으로써 현재 시스템 상황을 감시하는 것입니다.

Q: 역시 저자님의 리얼한 경험 사례를 좀 들려주세요.

A: 고객 시스템에 실시간 파일 감시 서비스를 설치하여 운영하던 중에 파일 변경이 감지되었다는 메일을 확인하여, 시스템에 들어가 로그인한 계정을 잠금 후, 사고 조사와 함께 고객에게 이 내용을 통보하니 자신들이 현재 작업 중이라고 한 경우가 있습니다.

시스템에 접근하겠다는 통보 후 진행해야 하나 간혹 정상 이용자들도 임의로 변경하는 경우도 많이 발생합니다.

23장 후킹

Q: 원천적으로 후킹을 막을 방법은 없나요?

A: 아직은 없습니다. 원천적으로 막을 수 없어서 계속 후킹과 역후킹이 나오는 상황입니다.

Q: 후킹의 대표적인 피해 사례는 무엇이 있을까요?

A: 키로거라 할 수 있습니다. 악성코드의 많은 부분을 차지하는 부분이기도 한데, 사용자 입력을 통해 계정과 패스워드를 해커에게 전송하는 역할을 합니다.

Q: 우리나라 금융기관의 시스템은 후킹 공격으로부터 안전하다고 할 수 있을까요?

A: 어느 정도 안전하다고 할 수 있습니다. OTP 등을 이용할 경우 후킹을 하였다 하더라도 인증키가 계속 변경되기 때문에 보다 안전하다 할 수 있습니다.

Q: 후킹과 관련하여 저자님의 경험담을 들려주세요.

A: 해킹된 시스템 조사 중에 IDT 테이블이 후킹된 것을 확인하고 해당 드라이버를 비활성화한 후 삭제하여 조치를 취한 적이 있습니다. 잘 조치되었는지 확인해 보니 다시 삭제한 파일이 복원되어 있고, 후킹도 계속 진행중인 걸로 확인되어 골치를 썩은 적이 있는데요.

알고 보니 해당 파일 외에 후킹된 드라이버가 하나 더 존재하고 시스템 프로그램에 Dll 인젝션도 함께 사용되어, 이 두 가지를 함께 조치함으로써 해당 문제점을 해결한 적이 있습니다.

이처럼 보통 문제점은 하나가 아닌 여러 가지인 경우가 많습니다.

24장 안티 디버깅: 프로그램 보호

Q: 이 세상에 수많은 프로그램이 있는데요. 꼭 필요하면 모르겠지만, 안티 디버깅까지 할 필요가 없는 프로그램이 훨씬 많을 것 같습니다. 이 기술을 적용해야 할 프로그램들은 주로 어떤 것들인가요?

A: 인증 기능이 있는 프로그램이라면 안티 디버깅을 적용하는 것이 좋습니다.

Q: 우매한 질문일 수 있겠으나, 요즘엔 거의 웹을 통해서 침입을 하는데, 웹의 주 언어가 컴파일 언어가 아닌 스크립트 언어의 경우엔 안티 디버깅 기술을 적용하지 않나요?

A: 웹을 통해 칩입한 한 후에 악성코드를 심었다면 해당 악성코드에 안티 디버깅 기술을 적용해 분석을 어렵게 하여 시간을 벌 수 있을 것입니다. 그리고 스크립트 언어는 스크립트 자체가 소스이기 때문에 안티 디버깅을 적용하지 않습니다.

Q: 프로그램 라이선스를 보호하기 위해 크랙킹을 방지하는 것도 안티 디버깅 기술이 적용된 건가요?

A: 디버깅 행위가 나아가 크랙킹으로 진행되는 경우가 많기에 대부분의 안티 디버깅 기술도 함께 적용하는 경우가 많습니다.

Q: 안티 디버깅 관련 경험담이 있다면 소해주세요.

A: 저자도 안티 디버깅을 제대로 풀지 못하여 포기한 프로그램이 있는 만큼, 안티 디버깅 기술이 적용된 프로그램의 언패킹 작업은 여러 안티 디버깅 기술이 적용되어 하나하나 실타래 풀 듯이 풀어가야 하다 보니 여간 힘든 작업이 아닙니다.

이러한 고충을 알기에 저 역시 시스템 감시 프로그램을 개발해 외부에 공개할 때 안티 디버깅을 통해 해당 프로그램을 보호하였습니다.

안티 디버깅은 분석가가 분석을 못하게 막는다기보다는 분석을 편하게 할 수 없도록 괴롭히는 데 목적을 두고 만드시기 바랍니다. 그리고 중요한 처리 작업은 클라이언트가 아닌 서버에서 처리하도록 구성하시는 게 안전합니다.

25장 네트워크 보호

Q: 네트워크 관련 내용을 알기 위해서는 상당한 학습이 필요할 것 같습니다. 이론적 배경과 시스코와 같은 실제 장비를 잘 다룰 수 있는 학습 로드맵이 있다면 간단한 소개를 부탁드립니다.

A: 네트워크는 물리적인 장비를 다루어야 하는 만큼 처음 진입을 독학으로 하기는 조금 어려운 점이 있습니다. 따라서 학원에서 기초 과정을 배우는 것을 추천합니다. 그리고 시스코는 자격증 로드맵이 잘 되어 있는데요. 자격증을 목표로 공부하면서 실제 업무에 활용하면 큰 도움이 될 겁니다. 그리고 CCIE의 경우 아직도 많은 네트워크 기업에서 인정해주는 네트워크 전문가 자격증인 만큼 네트워크 전문가에 관심이 있다면 도전해보는 것도 학습 효율에 큰 도움을 줄 것입니다.

Q: DDOS 공격은 원천적으로 막을 수 없지만 최소화할 수 있다고 했는데, 최소한의 피해라도 서비스에는 피해가 가고 결국엔 기업의 이미지 추락으로 이어지지 않을까요?

A: DDoS 공격을 막기 위해서는 공격이 들어오는 트래픽을 처리할 수 있는 30~40Gbps의 큰 네트워크 대역폭과 1억 이상 나가는 고가의 DDoS 방어 장비를 갖추고 있어야 하나, 이러한 장비를 구비할 수 있는 업체는 몇 없기에 피해를 최소화 하는 방향으로 대비하게 됩니다. 이에 따른 기업 이미지 피해도 어느 정도는 감내해야 하지 않을까 합니다.

한국 인터넷 진흥원에서 주관하는 사이버 대피소 서비스 등을 이용하여 DDoS 피해를 최소화하는 대피 계획을 사전에 수립하는 게 좋습니다.

Q: 어찌 보면 보안 분야에서는 네트워크 보호가 가장 중요할 듯한데, 이 책에서는 간략하게 다루고 있는 것 같습니다. 특별한 이유가 있는지요?

A: 네트워크 부분은 3부에서도 나왔던 내용으로 25장에 필요한 내용으로만 구성한 것입니다. 그리고 네트워크 접근 제어 부분인 ACL이 네트워크 보호의 기본이 되는 부분이기에 핵심만 다루고 싶었습니다. 보다 상세한 내용을 알고 싶다면 http://www.cisco.com/cisco/web/support/index.html에서 각 제품별로 기술 자료를 찾거나, 특정 키워드를 통해 공부할 것을 추천합니다.

Q: 이 장에서 소개하는 네트워크 장비 등에 대한 에피소드가 있다면 소개해주세요.

A: 예전 용도별로 네트워크를 구분하고 신규 서비스 대역에는 방화벽이 없어 ACL로 구성하도록 네트워크 엔지니어에게 지시하였습니다. 그렇게 서비스를 개시하고 차후에 확인을 하였는데, ACL 마지막 라인에 permit ip any any가 입력되어 ICMP 패킷만을 차단하고 그 외 모든 트래픽이 허용인 상태인 것을 확인하여 ACL을 재구성한 후 신규 서비스 대역에 속한 전체 서버를 초기화하고 재설치한 기억이 있습니다. ACL을 설정하실 때는 항상 보안을 생각하고 "알고 있는 일부를 막는 것이 아닌 알 수 없는 모든 것을 막는다"는 개념으로 진행하기 바라며, 전자처럼 ICMP 패킷만 차단하는 엔지니어가 되지 마세요.

26장 IP 기반 로그인 보안 프로세스 구성과 개발 (실습)

Q: 이 장을 보면 수많은 SQL문에 웹 프로그래밍까지 필요한 듯 보입니다. 바보 같은 질문일 수도있지만 프로그래머와 리버서(?)의 경계는 과연 어디일까요?

A: 대부분의 리버서는 프로그래밍 능력을 가지고 있습니다. 즉 프로그램을 만들 수 있어야 보다 나은 리버서가 될 수 있으며, 여러 기술 분야를 알아야 하기에 어려운 분야이기도 합니다. 그리고 프로그래밍 언어는 서로 달라 보이지만 사용되는 개념은 유사하여 하나의 언어를 다룰 수 있으면 다른 언어도 쉽게 학습할 수 있으므로, 경계를 두고 다른 기술 분야를 배제하기보다는 서로 필요한 요소라고 생각하고 매사에 적극적으로 배우고자 하는 자세로 임하는 게 좋습니다.

Q: 로그인 영역은 어찌 보면 프로그래밍 영역인데, 개발 전 단계에서 헛점을 보완한다면 개발 후에 특별하게 문제가 생기지는 않을 것 같은데요?

A: 아직 로그인 보안이라는 부분이 일반화 되어 있지 않으며, 사이트마다 이용자의 특성에 맞춰야 하므로 무조건 적용할 수도 없을 것입니다. 따라서 여러 가지의 보안 방식을 습득하여 본인의 사이트에 알맞은 보안 프로세스를 개발하시기 바랍니다. 그리고 그 주체가 보안 담당자일 수도 있고, 보안에 관심이 많은 웹 개발자일 수도 있으니, 꼭 개발 주체를 나눌 필요는 없지만 개발 단계에서부터 고려된다면 완성도가 높은 프로그램이 개발되는 것은 확실합니다. 최근에는 개발 단계에서부터 보안 전문가가 개입하여 프로그램상에서 발생할 수 있는 보안 위험을 최소화하는 개발 방법론도 있으며, 개발 시 코드 작성 유의 사항인 시큐어 코딩 가이드(http://goo.gl/m1dHU)도 유용하므로 참고하시기 바랍니다.

Q: 로그인 프로세스 관련한 경험 사례가 있다면 들려주세요.

A: 여기에 소개된 로그인 프로세스는 실제 게임 업체에 적용한 프로세스로서 기술은 다르지만 개념은 같다고 할 수 있습니다. 이 로그인 프로세스를 통해 계정 해킹을 막을 수 있었고 더 이상의 계정 어뷰징 공격은 발생하지 않아 그해 업무 평가에 큰 역할을 해주었답니다.

27장 NoSQL?!: 부하 분산 (실습)

Q: 이 장은 성능이라는 영역의 극히 일부분을 다루고 있는 것 같습니다. 전체 성능의 범위를 요약해서 정리할 수 있을까요?

A: 성능이라는 부분이 서비스 요소별로 존재하기 때문에 전체를 요약한다는 것은 불가능할 것 같습니다(메일, DNS, ARP까지 IT의 모든 서비스와 이를 연결하는 각 구성 요소들도 범위에 들어갑니다). 다만, 병목지점을 찾는 작업이라고 말씀드릴 수 있을 것 같습니다. 그리고 병목이라고 판단하는 기준을 정하는 것과 구성 단위를 정의하는 작업이 중요한데, 이 부분은 경험적인 요소가 많은 비중을 차지합니다(경험자로부터 듣는 간접적 경험도 좋습니다). 성능 관리를 잘하기 위해서는 경험과 다양한 분야에 관심을 갖고 계속 배워나가야 합니다.

Q: 성능에 대한 추천 학습 방법이 있다면 소개해주세요.

A: 테스트 환경을 통한 성능 테스트를 지속적으로 진행하는 방법과 여러 분야의 기술을 찾아 경험해 보시기 바랍니다. 웹이라 하여도 웹에 이용되는 기술이 웹뿐만이 아니라 운영체제, 데이터베이스 등 각 영역별로 성능을 높일 수 있는 방법들이 존재하므로 이를 어떻게 잘 통합하는지도 엔지니어가 갖추어야 할 중요한 기술이라고 볼 수 있습니다.

Q: 저자님은 평소에 성능 관리를 어떻게 하시는지요?

A: 성능 관리를 위해서는 경험적인 지식과 여러 분야의 지식이 필요할 때가 많습니다.

기본적으로 진행하는 사항을 말씀드리면, 성능 관리가 필요한 경우 기간을 두고 성능 확인에 필요한 데이터를 수집한 후 병목 현상이 발생하는 부분을 찾아 개선방안을 마련하게 됩니다.

병목 현상을 확인하기 위해서는 만약 운영체제라면 프로세서, 메모리, 디스크, 네트워크에 대해 보다 세부적인 데이터를 수집하여야 합니다(1부 6장 "운영-성능/관리" 확인).

그리고 네트워크라면 컴퓨터 하나가 아닌 전체 네트워크에서 보다 효과적인 네트워크 환경을 위한 구조 개선도 함께 진행되어야 하는 경우도 많습니다.