엔트로피를 이용한 실행 압축 해제 기법 연구

정구현*, 추의진*, 이주석*, 이희조*

Generic Unpacking Using Entropy Analysis

GuHyeon Jeong*, Euijin Choo*, Joosuk Lee* and Heejo Lee*

본 연구는 지식경제부 및 정보통신연구진흥원의 IT핵심기술개발사업의 일환으로 수행하였음.
[2008-S-026-01. 신종 봇넷 능동형 탐지 및 대응 기술]
본 연구는 지식경제부 및 정보통신연구진흥원의 대학IT연구센터 지원사업의 연구결과로 수행되었음.
(IITA-2008-C1090-0801-0016)
본 연구는 교육부 BK21 사업의 지원을 받아 수행되었음.

요 약

봇이나 웜 등의 악성코드는 오늘날 가장 큰 위협으로 자리 잡았다. 공격자들이 이들을 이용하여 일반 사용자들의 개인 정보를 수집하거나 금전적 피해를 입히고 있기 때문이다. 최근의 악성 코드들은 안티 바이러스 프로그램을 회피하기 위해서 실행 압축 등의 코드 혼란 기법을 사용하고 있다. 이로 인해 악성 코드의 탐지 및 방어가 점점 더 어려워지고 있다. 이를 해결하기 위해서 본 논문에서는 엔트로피 분석을 이용한 범용적인 실행 압축 해제 기법을 제안한다. 실험한 결과 실행 압축 알고리즘의 종류에 따라서 초기의 엔트로피와 실행 압축 해제 후의 엔트로피 차이가 작게는 2.5, 크게는 4 정도까지 눈에 띄는 엔트로피 값의 변화를 보였다. 이를 통해 본 논문에서 제안하는 실행 압축 해제 알고리즘을 이용하여 실행 압축을 해제할 수 있음을 확인하였다.

Abstract

Malwares, such as bot, worm and so on, have become the biggest threat in these days. Attackers collect private information and cause monetary damage by taking advantage of them. Recent malwares use packing, which is one of code obfuscation mechanisms, in order to bypass an Anti-Virus program. It makes analysis and detection of malwares harder and harder. This paper suggests a generic unpacking mechanism using an entropy analysis. Experimental results show that measured entropy is changed during an unpacking process with a noticeable change of 2.5 or even 4 from its initial value depending upon a packing algorithm. The suggested mechanism can be used for getting unpacked original codes from various packed executables.

Key words
Unpacking, Entropy, Malware, Dynamic analysis

* 고려대학교 정보통신대학 컴퓨터·통신공학부
· 제1저자 (First Author) : 정구현
· 교신저자 (Corresponding Author) : 이희조
· 접수일자 : 2008년 12월 23일, 수정일 : 1차 – 2009년 01월 09일, 게재확정일 : 2009년 02월 17일

Ⅰ. 서 론

실행 압축 기술은 실행 파일을 압축하여 실행 파일이 점유하는 저장 공간의 크기를 줄이는 기술이다. 실행 압축된 프로그램은 저장하고 있던 실제 코드와 데이터를 압축해제 루틴에 의해서 풀어낸 후 압축 해제된 실행 파일의 엔트리 포인트로 실행 흐름을 바꾸어 원래의 프로그램을 실행한다. 따라서 실행 압축 기술을 이용하면 보다 작은 크기의 저장 공간을 점유하면서 원래의 실행 파일과 같은 기능을 하는 또 다른 실행 파일, 즉 실행 압축 파일을 만들 수 있다.

최근의 많은 악성 코드들은 이러한 실행 압축 기법을 이용하여 파일 크기를 줄임으로써 악성 코드들의 전파 속도를 빠르게 하고 있으며 원래 코드를 변형시킴으로써 악성 코드의 분석을 어렵게 만들고 있다. 또한 많은 악성 코드들이 실행 압축 기술과 함께 암호화 기법을 사용하여 분석을 더욱 어렵게 만들고 있다. WildList의 2006년 3월 자료에 따르면 92% 이상의 악성 코드가 실행 압축 혹은 암호화되어 있다고 조사되었다[1]. 그러므로 이러한 악성 코드들에 대응하기 위하여 실행 압축하기 전의 프로그램을 복원하여 분석할 수 있는 실행 압축 해제 기술이 필요하다.

실행 압축 파일은 크게 실행 압축 해제 부분과 실제 기능을 수행하는 부분으로 나누어진다. 오리지널 엔트리 포인트(Original Entry Point)는 실행 압축 파일이 스스로 실행 압축을 해제한 뒤 첫 번째로 실행하는 명령어의 주소로서 원래 프로그램의 엔트리 포인트이다. 즉, 오리지널 엔트리 포인트 이후에 실행되는 코드가 원래 프로그램의 코드이므로 실행 압축 해제 기술에 있어서 가장 중요한 점은 오리지널 엔트리 포인트를 찾는 것이다.

악성 코드 제작자들은 여러단계에 걸쳐 실행 압축하거나 기존 실행 압축 기법의 변형된 버전을 사용하는 등 다양한 기법을 이용하여 오리지널 엔트리 포인트를 찾기 어렵게 한다. 이에 본 논문에서는 엔트로피 분석을 통하여 효율적으로 오리지널 엔트리 포인트를 찾는 기법을 제안한다.

실행 압축된 파일은 일반적인 실행 파일에 비해서 엔트로피가 더 높다[2]. 본 논문에서는 실행 압축 파일이 실행될 때 메모리 상태를 관찰하고 데이터들의 엔트로피 값을 분석하여 실행 압축 파일의 오리지널 엔트리 포인트를 찾는다. 또한, 실험 결과들을 통하여 본 논문에서 제안한 기법이 실행 압축 알고리즘에 의존하지 않고 오리지널 엔트리 포인트를 찾을 수 있음을 보인다.

본 논문의 2장은 실행 압축 해제 관련 연구에 대해서 서술하고 3장에서는 엔트로피의 정의와 이를 이용한 관련 연구에 대해 기술한다. 4장에서는 제안하는 실행 압축 해제 기법을 자세히 설명하고 이 기법의 효율성을 5장에서 실험 결과를 통하여 보인다. 마지막으로 6장에서 결론을 맺으며 본 연구를 토대로 앞으로 더욱 연구하고 개선할 내용을 기술한다.

Ⅱ. 실행 압축 해제 기법

실행 압축 기술을 적용한 악성 코드들은 탐지와 분석 시간을 지연시켜 악성 코드의 피해를 누적시킬 뿐 아니라 악성 코드 제작자들로 하여금 다른 악성 코드를 제작할 시간적 여유를 갖게 한다. 이에 따라 실행 압축 해제 기술에 관한 많은 여구가 이루어져왔다.

기존의 연구들은 인력을 이용하여 수동으로 직접 악성코드를 분석하거나 특정 실행 압축 기법에 대하여 작동하는 알고리즘의 개발 혹은 범용적으로 사용할 수 있는 실행 압축 해제 기술에 대한 연구 등 크게 세 가지로 분류할 수 있다.

첫 번째 방법은 디버거 및 분석 도구들을 이용하여 사람이 직접 분석하는 것이다. 이는 실행 압축 해제 연구의 초창기 단계에 이루어졌던 방법으로 가장 대표적인 분석 도구로는 OllyDbg[4] 플러그 인, Immunity Debugger[5], IDA 플러그 인[6] 등이 있다. 하지만 이러한 방식을 통한 분석은 실행 프로그램의 수많은 명령어들을 모두 분석하여야 하므로 너무 많은 시간을 소모한다.

수동 분석의 부담을 줄이기 위해 특정 실행 압축 알고리즘의 특징을 기반으로 하여 압축을 해제한 후 악성 코드를 탐지 혹은 분석하는 기법이 제안되었다. 이 방식은 안티 바이러스 프로그램들에서 일반적으로 쓰이고 있지만 이 방식은 새로운 악성 코드뿐만

아니라 기존에 알려진 실행 압축 기법이 일부 변형된 경우에도 기민하게 대처하기 어렵다. 따라서 실행 압축 기법에 의존하지 않는 실행 압축 해제에 관한 연구가 진행되었다.

PolyUnpack[7]은 실행 압축 파일의 실행 흐름은 결국 원래의 실행 파일의 명령어에 도달할 수밖에 없다는 가정 하에 실행 압축한 코드를 연속적으로 일부분씩 실행한다. 그리고 그 부분에 대한 분석을 통하여 해당 코드가 악성 코드인가를 판단한다.

OmniUnpack[8]은 모든 실행 코드를 분석하지 않고 메모리를 관찰하여 프로세스가 데이터를 쓴 메모리 영역으로 실행 흐름이 바뀌는 순간 그 영역을 분석한다. 이를 통하여 OmniUnpack은 PolyUnpack에 비해서 필요 없는 분석 시간을 줄여 전체 분석에 걸리는 시간을 단축하였다.

Renovo[9]는 가상 머신을 사용하여 실행 압축 해제하는 기법을 제안한다. 가상 머신을 이용하면 분석에 걸리는 속도가 다소 느려질 수 있지만 분석하는 동안 악성 코드 분석 도구가 원래의 시스템으로부터 격리되어 감염의 위험성이 현저히 낮아진다는 점과 기존의 기법들과 달리 프로세스의 관점이 아니라 전체 시스템 관점에서 악성 코드의 분석이 가능해진다는 장점이 있다.

Renovo는 여러 단계에 걸쳐 압축된 코드 수행 시 메모리 맵인 쉐도우 메모리를 이용하여 각 단계 별 메모리 상태를 기록하는 등 실행 압축 파일의 실행 압축 해제 과정을 고려하였으나 원래 코드가 시작되는 부분, 즉 오리지널 엔트리 포인트가 정확히 어디인지 알 수 없어 한계가 있다.

Ⅲ. 엔트로피

일반적으로 사용되는 엔트로피는 열역학적 계의 상태 함수 중의 하나로 통계적인 무질서도를 나타낸다. 이 개념을 차용하여 Claude Elwood Shannon이 정보 엔트로피라는 개념을 제안하였다[10].

Shannon은 정보 엔트로피의 개념을 통하여 정보의 양을 수치화하여 다음과 같은 수식으로 정보 엔트로피 H를 정리하였다.

$$H(X) = \sum_{i=1}^{n} p(x_i) I(x_i) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i) \quad \cdots(1)$$

$p(x_i)$는 x_i가 발생할 확률이고 I는 이산 확률 변수 X의 자기 정보량(Self-information)을 의미한다. log의 밑인 b의 값으로는 일반적으로 2, 오일러 수 e, 10을 자주 사용한다.

일반적으로 정보 이론에서의 엔트로피는 메시지의 압축에 관한 분야에 대해서 연구할 때 많이 사용한다. 예를 들어 압축 알고리즘의 압축률을 평가할 때 유용하게 사용할 수 있다. 엔트로피가 높은 데이터일수록 나타날 수 있는 모든 비트들이 고루 존재함을 의미하므로 어떤 압축 파일의 엔트로피 수치가 높을수록 압축률이 높다고 말할 수 있다.

[2]는 이러한 특성을 바탕으로 실행 압축한 파일이 일반 실행 파일보다 더 높은 엔트로피를 갖는다는 것을 보였다. [2]에서의 실험에 따르면 일반적인 텍스트 파일과 실행 파일, 실행 압축한 실행 파일 그리고 암호화한 실행 파일의 평균 엔트로피 수치가 각각 4.347, 5.099, 6.081 그리고 7.175의 값을 갖는다. 다시 말해서 실행 압축된 파일과 일반 실행 파일은 엔트로피 값의 비교에 의해서 구분할 수 있다.

Ⅳ. 엔트로피를 이용한 실행 압축 해제 기법

이 장에서는 제안하는 실행 압축 해제 기법을 설명한다. 실행 압축 파일을 실행하면 메모리에 값을 읽고 쓰면서 실행 압축 해제 과정을 거치게 된다. 본 논ㄴ문에서 제안하는 기법은 실행 압축 해제 과정을 진행하고 있을 때, 메모리 상태의 엔트로피 값의 변화를 관찰하여 오리지널 엔트리 포인트를 찾아낸다. [2]와 같이 실행 파일을 대상으로 실행 압축 여부에 대한 판단을 한 다른 연구는 있지만[10] 본 논문과 같이 실행 파일을 실행하고 있는 프로세스 상태를 분석하여 오리지널 엔트리 포인트를 찾으려는 시도는 없었다.

실행 압축된 파일이 일반 시랭 파일에 비해 높은

엔트로피 값을 가지므로 압축 해제가 진행되는 초기 단계에는 실행 압축된 코드로 인해서 해당 프로세스 메모리의 엔트로피 값이 크고 해제하는 과정을 거치면서 점차 메모리의 엔트로피 값이 작아질 것이라고 생각할 수 있다. 그러나 프로세스의 가상 메모리 전체에 비하여 실행 압축한 원래의 프로그램이 차지하는 비중이 얼마 되지 않기 때문에 전체 가상 메모리 영역을 대상으로 엔트로피 값을 측정한 결과로는 엔트로피의 변화 추이를 알아보기가 힘들다. 그러므로 엔트로피를 이용하여 메모리 상태의 변화 추이를 알기 위해서는 특정 메모리 영역에 대한 관찰이 필요하다.

본 논문의 저자들은 한 개의 프로세스에서 엔트로피의 변화 추이를 관찰한 가상 메모리 영역을 최소화하기 위해 실험을 진행하였다. 실제로 사용되는 25개의 다른 실행 압축 알고리즘 – Epack, alternative exe packer, ASProtect, AZProtect, CryptX 1.0, Enigma Protector 1.16, fsg, MoleBox 2.6.1, morphine, mpress, PE_Armor_ExeCRyPT, npack, nspack, petite23, RLPack.1.16.Full, Themida 1.9.5, WinUpack039e, DalKrypt, yoda cryptor, UPX, UPX-iT, upxn, Zprtect_Demo, UPX_scramble, UPX_shell – 을 대상으로 분석한 결과 모두 4GB의 가상 메모리 영역 중에서 실행 압축되지 않은 파일의 실행 코드가 있던 위치와 같은 메모리 주소에 원래 프로그램을 실행 압축 해제 하는 것을 알아낼 수 있었다. 원래의 코드가 쓰이는 위치는 실행 압축 프로세스의 첫 번째 섹션에 해당하기 때문에 본 논문에서는 실행 압축 프로세스 첫 번째 섹션의 엔트로피 값의 변화 추이를 관찰하였다.

그림 1은 제안하는 압축 해제 기법의 흐름도이다. 전체적인 엔트로피 변화의 추이를 살피기 위해 타겟 프로세스에 대해서 명령어 단위의 실행을 해야 한다. 하지만 엔트로피의 측정을 명령어가 실행되는 매 순간 수행하는 것은 비효율적이다. 오리지널 엔트리 포인트는 결국 분기문 이 후가 되기 때문에 본 논문에서는 명령어가 JMP 혹은 CALL 계열인 경우에만 현재 지정된 메모리의 엔트로피 값을 계산하였다. 실행 압축 해제 과정을 통해 원래 파일의 데이터와 코드를 다 쓰고 나면 더 이상 관찰하고 있는 영역의 엔트로피는 변하지 않고 고정된다. 엔트로피 값이 고정된 이후 관찰 영역으로 실행 흐름이 옮겨지면 실행 흐름이 옮겨진 주소가 오리지널 엔트리 포인트이다. 이러한 실행 아축 해제 기법은 실행 압축 알고리즘에 의존하지 않는 엔트로피 분석을 이용하여 압축 해제를 함으로써 알려지지 않은 기법으로 압축된 실행 파일에 대해서도 효율적으로 실행 압축 해제를 할 수 있다.

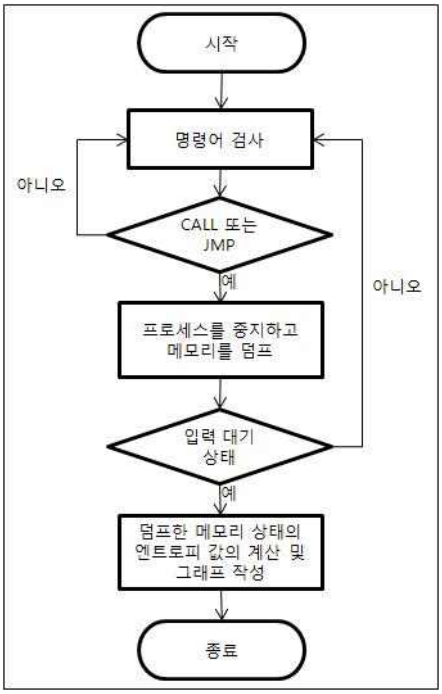

그림 1. 엔트로피를 이용한 실행 압축 해제 기법의 흐름도

Fig. 1. Flowchart of a generic unpacking mechanism with an entropy analysis

V. 실험 및 결과 분석

5장에서는 제안하는 실행 압축 해제 기법을 이용하여 실제로 오리지널 엔트리 포인트를 찾을 수 있는지 실험을 하고 그 결과에 대해서 분석한다.

5.1 실험 환경

실험은 Microsoft Windows XP SP3 운영체제에서 이루어졌다. 실험이 이루어진 머신은 2.13GH의

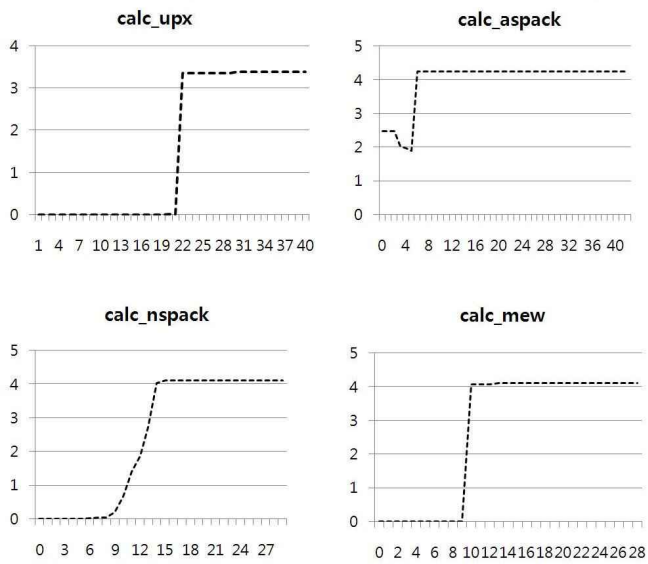

그림 2. UPX, aspack, nspack, mew로 실행 압축한 'calc.exe' 프로그램의 엔트로피 변화
Fig. 2. Entropy changes of 'calc.exe' program packed with UPX, aspack, nspack and mew

Intel Core(TM)2 CPU를 사용하였고 메모리는 2.00 GB이다. 엔트로피 값을 계산하기 위해서 3장에서 언급한 Shannon의 엔트로피 공식을 사용하였고 log의 밑으로는 오일러 수 e를 사용하였다.

오리지널 엔트리 포인트의 탐지를 위해서 실험에 사용할 프로그램을 작성하였다. 작성한 프로그램은 실행 압축 파일을 입력받아 CALL 및 JMP 계열의 명령어를 확인하여 지정된 메모리 영역의 엔트로피 수치를 계산한다. 이 과정을 반복적으로 수행하여 실행 프로그램이 사용자의 입력을 받기 위해 대기하고 있는 상태가 될 때까지 진행한다.

실행 압축 알고리즘의 동작 방식을 분석하기 위해서 사용한 실행 파일은 윈도우즈 운영체제에서 제공하는 계산기 프로그램(clac.exe)을 사용하였다. 이 파일을 실제로 많은 악성 코드들이 사용하고 있는 44개의 실행 압축 알고리즘들을 이용하여 실행 압축하고 OllyDbg, Imuunity debugger, Zeta debugger 등 총 15개의 디버거를 이용하여 분석을 진행 하였다.

5.2 실행 압축 해제 실험 결과 및 분석

그래프의 가로축은 엔트로피 수치를 측정한 시점으로 JMP 혹은 CALL 계열의 명령어가 수행되는 시점이고 세로축은 엔트로피 수치를 나타낸다.

그림 2에서 upx, nspack, mew를 이용하여 실행 압축한 실행 파일의 경우 그래프의 전체적인 추이가 aspack의 경우와 다른데 이는 앞의 세 가지 실행 압축 알고리즘이 관찰 영역이 0으로 초기화되어 있기 때문이다. 이에 비해 aspack은 코드 섹션이 초기화되어있지 않아 초기 값이 0이 아닌 값을 갖는 것을 확인할 수 있었다. 이러한 차이는 실행 압축 알고리즘들을 구분할 수 있는 특징이 될 수 있다.

압축 해제 과정에 있어 초기에는 엔트로피 값이 지속적으로 소폭의 변동을 보이지만 이는 그 폭이 너무 작아 직선의 추이를 보인다. 하지만 오리지널 엔트리 포인트 근처에서는 그래프에서 확인할 수 있듯이 큰 변화를 관찰할 수 있고 그 이후에는 값이 일정하게 유지된다. 이는 실행 압축 해제 과정이 종료되어 관찰 영역에 더 이상 데이터를 쓰지 않기 때문이다. 그러므로 엔트로피가 일정하게 유지되기 시작하는 부분이 오리지널 엔트리 포인트라고 할 수 있다.

이러한 실험 결과는 본 논문에서 제안하는 실행 압축 해제 기법이 특정 실행 압축 기법에 의존하지 않고 오리지널 엔트리 포인트를 탐지하여 실행 압축을 해제 할 수 있음을 보여준다.

Ⅵ. 결론

본 논문에서는 최근 악성 코드의 방어를 더욱 어렵게 만들고 있는 실행 압축 알고리즘을 해결하기 위해서 엔트로피를 이용한 실행 압축 해제 기술을 제안하였다.

실행 압축 파일이 첫 번째 코드 섹션에 실행 압축을 해제함에 따라 그 영역의 엔트로피 변화를 분석하였고 실험의 결과로서 실행 압축 알고리즘에 따라서 2.5에서 4의 엔트로피 값의 차이를 보인다는 것을 확인하였다. 이 결과는 제안한 기법이 실행 압축 알고리즘의 종류에 의존하지 않는 범용적 실행 압축 해제 기법임을 보인다.

앞으로의 연구에서는 본 논문에서 제안한 실행 압축 해제 기법을 더욱 효과적으로 발전시키고 자동화된 도구를 이용하여 실제로 실행 압축된 악성 코드들을 대상으로 실험하여 오리지널 엔트리 포인트를 찾고 실행 압축을 해제한 실행 파일을 복원하는 연구를 진행할 예정이다.

감사의 글

본 연구는 2008년도 지식경제부 및 정보통신 연구진흥원, 한국과학재단, 교육부 BK21 사업의 지원에 의하여 이루어진 연구로서, 관계부처에 감사드립니다.

참 고 문 헌

[1] Richard Ford and Michael Howard, "Revealing Packed Malware," IEEE Security and Privacy, Vol. 6, pp. 65-69, Sep/Oct, 2008.

[2] Robert Lyda, James Hamrock, "Using Entropy Analysis to Find Encrypted and Packed Malware," IEEE Security and Privacy, Vol. 5, no. 2, pp. 40-45, Mar/Apr, 2007.

[3] OllyDbg, http://www.ollydbg.de/.

[4] Immunity debugger : http://debugger.immunityinc.com/.

[5] IDA 플러그인 : http://www.hex-rays.com/idapro/.

[6] P. Royal, M. Halpin, D. Dagon, R. Edmonds and W.Lee, "PolyUnpack: Automating the Hidden-Code Exetraction of Unpack-Executing Malware," In Proc' of 2006 Annual Computer Security Applications Conference (ACSAC) pp. 289-300, 2006. IEEE Computer Society.

[7] Martignoni, L. Christodorecu. M, and Jha, S, "OmniUnpack: Fast, Generic and Safe Unpacking of Malware," In Proc' of 2007 Annual Computer Security Applications Conference (ACSAC), pp. 431-441, 2007. IEEE Computer Society.

[8] M. G. Kang, P, Poosankam and H. Yin. "Renovo: A hidden code extractor for packed executables," In Proc' of the 5th ACM Workshop on Recurring Malcode(WORM'07), Oct. 2007.

[9] Thomas M. Cover and Joy A. Thomas, "*Elements of Information Theory,*" Second Edition. Wiley Interscience, New York, NY, 2006.

[10] Roberto Perdisci, Andrea Lanzi and Wenke Lee, "Classification of packed executables for accurate computer virus detection," SciencDirect, Vol. 29, pp.1941-1946, Oct, 2008.

저자소개

정 구 현 (GuHyeon Jeong)

2008년 2월 : 고려대학교 컴퓨터학과(공학사)

2008년 3월~현재 : 고려대학교 컴퓨터·전파통신공학과 석사과정

관심분야 :악성코드, 네트워크 보안

추 의 진 (Euijin Choo)

2006년 2월 : 고려대학교 컴퓨터학과, 수학과 전공(이학사)

2008년 2월 : 고려대학교 컴퓨터학과 석사

관심분야 : 악성코드, P2P 보안

이 주 석 (Joosuk Lee)

2008년 2월 : 고려대학교 컴퓨터학과(공학사)

2008년 3월~현재 : 고려대학교 컴퓨터·전파통신공학과 석사과정

관심분야 : 악성코드, 네트워크 보안

이 희 조 (Heejo Lee)

1989년 3월~2001년 2월 : 포항공대 컴퓨터공학과 학사/석사/박사

2000년 3월~2001년 2월 : Perdue University 박사후연구원

2001년 3월~2003년 10월 : 안철수 연구소 CTO

2004년 3월~현재 : 고려대학교 컴퓨터·전파통신공학과 부교수

관심분야 : 네트워크 보안, 악성 코드 탐지 및 분석