SSDT를 이용한 웜 탐지 시스템 설계 및 구현

정회원 황 유 동*, 이 유 리*, 박 동 규*, 임 황 빈**, 오 진 태***, 장 종 수***

Design and Implementation of a Worm Detection System Using SSDT (System Service Dispatch Table)

Yu-dong Hwang*, You-ri Lee*, Dong-gue Park*, Hwang-bin Yim**,
Jin-tae Oh***, Jong-soo Jang*** *Regular Members*

요 약

네트워크 침해 유형이 특정 호스트나 서버를 목표로 하여 개별 시스템이나 해당 시스템이 제공하는 서비스에 대한 공격에서 광역 네트워크 서비스 제공에 대한 공격으로 변화 하고 있다. 또한 네트워크 인프라의 속도가 증가되고 서버들의 성능이 개선됨에 따라서 공격 피해의 전파 속도 또한 빨라지고 있다. 이에 따라서 웜은 그 특성상 빠른 공격이 예상된다. 따라서 본 논문에서는 웜을 통한 공격으로부터 네트워크 인프라를 보호하기 위해서 SSDT(System Service Dispatch Table)를 이용한 웜 탐지 시스템을 설계 및 구현한다.

Key Words : Worm Detection, SSDT

ABSTRACT

The specific host or the server is appointed as a goal and the server has the network breach type in an attack to the individual system or the service which the corresponding system provides to the attack about the wide area network service providing with a change. Moreover, as the speed of the network infrastructure is increased and the performance of the servers is improved, the speed moreover gets raged with the propagation velocity of the attack damage. Accordingly, as to the worm, the fast attack of the on characteristic real-time is predicted. Therefore, in this paper looked, it tries to design and implementation of the worm detection system for protecting the network infrastructure from the attack through the worm.

Ⅰ. 서 론

정보기술의 발달로 누구나 네트워크를 통하여 텍스트, 이미지 등의 멀티미디어 데이터를 접할 수 있게 되었으며, 이를 통한 원격 회의, 가상 학습, 원격 진료 등의 네트워크를 이용한 산업 서비스가 가능하게 되었다. 이러한 통신과 네트워크 기술의 발달과 함께 네트워크를 통한 웜의 공격 형태 역시 광역 네트워크 서비스 제공에 대한 공격으로 변화 하였으며 이에 네트워크 인프라 속도의 증가와 서버들의 성능 개선으로 인하여 공격 피해의 전파 속도가 단축되면서 더욱 치명적인 형태로 진화하고 있다. 본 논문에서는 이러한 공격에 대처하기 위하여 웜의 가장 근본적인 속성인 자기 복제 성질을 이용하고자 한다.

마이크로 소프트의 윈도우즈의 운영체제는 매초 마다 많은 프로세스로부터 수백 개의 호출을 수신한다. 어플리케이션에 의한 대부분의 func-

* 순천향대학교 정보통신공학과 (hwangyudong@gmail.com, {thisglass, dgpark}@sch.ac.kr)
** 강원도립대학 정보통신공학과 (hbinyim@gw.ac.kr), *** 한국전자통신연구원({showme, jsjang}@etri.re.kr)
논문번호 : 07089-1119, 접수일자 : 2007년 11월 19일

tion calls은 system calls로 넘어가게 되고, 커널 레벨에서 SSD(System Service Dispatcher)에 의해 동작한다. 자기 복제 하는 웜들은 커널 모드에서 동작하게 되므로 커널 모드에서 system calls을 모니터링하게 되면 자기 복제를 위한 웜 코드를 탐지 할 수 있게 된다. 따라서 본 논문에서는 SSDT(System Service Dispatch Table)를 이용하여 커널 모드에서의 system calls을 모니터링하여 자기 복제하는 웜을 탐지하기 위한 시스템을 구현하고자 한다.

Ⅱ. 자기 복제 성질을 이용한 웜 탐지 기법 연구

네트워크의 발달로 네트워크를 통한 데이터 전송 속도가 빨라지면서 웜 공격에 대한 피해 속도 또한 빨라지게 되었다. 따라서 이에 대처하기 위한 여러 가지 웜 탐지 기법이 연구 되었고 대표적인 웜 탐지 기법으로 시그니처 기반 생성 기법[1][2][3][4], 데이터 마이닝 기법[5], 하이브리드 방법[6][7][8][9][10]에 의한 웜 탐지 기법들이 있다. 이러한 탐지 기법은 웜의 다양한 성질을 이용한 것이다. 그 중 웜의 가장 기본이 되는 근본적인 성질은 웜의 자기 복제 성질이다. 자기 복제 특성을 이용한 웜 탐지 기법을 이용하면 가장 넓은 범위의 웜을 탐지함으로써 신뢰 있는 웜 탐지 시스템 구현이 가능하다.

자기 복제 성질을 이용한 최초의 연구는 정상적인 프로그램의 모든 시스템 호출 시퀀스를 저장한 후 모니터 되는 프로그램이 발생 시킨 시스템 호출 자료중의 부정합된 시퀀스가 일정빈도 이상을 넘어설 경우 침입으로 탐지하는 방법[11]으로 시스템 콜 인수를 고려하지 않음으로 두 번째 변종 웜 탐지 에러를 발생 시킬 수 있으며 고정된 슬라이딩 윈도우를 사용하여 오랜 시간 동안의 상관관계가 발생 할 때는 제한적일 수 있다. 이를 확장하여 변화 할 수 있는 길이의 파라미터를 가진 윈도우를 사용한 방법[12]도 제안되었으나 위와 같은 단점을 가지고 있다. 자기 복제 성질을 이용한 최근 연구는 바이러스의 알려진 시그니처에 의존하지 않고, 실행시간 동안에 자기 복제를 위한 실행 코드를 발견하는 방법[13]이다. 이 방법은 GSR(Gen of Self Replication) 구조를 사용한다. GSR은 컴퓨터 운영체제에서 프로그램이 실행 될 때 시스템 콜들의 시퀀스로 이뤄지며 블록으로 구성된다. 즉 GSR은 시스템 콜의 연관성을 고려하여 서브 패턴들을 만들고 이들을 하나의 블록으로 생성한다. 각 블록들은 피라미드 구조를 가지며 가장 기본적인 시스템 콜은 하위에 위치하며 블록들에 의해서 연관되어 결합되는 콜들은 중간에 그리고 GSR은 상단에 위치하게 된다. 따라서 운영체제에 의해 실행되는 시스템 콜을 모니터링하여 하위 블록부터 GSR 블록까지의 시퀀스가 구성 될 때 웜으로 탐지 된다.

Ⅲ. SSDT를 이용한 웜 탐지 시스템 설계 및 구현

윈도우즈는 유저모드(User Mode)와 커널모드(Kernel Mode)로 구분되어 프로세스가 실행된다. 일반적인 윈도우 프로그램의 경우 유저모드 프로세스가 작동하게 되므로 시스템의 자원을 직접 액세스할 수 없으며, 커널모드에서 동작하는 프로세스만 시스템의 자원을 직접 액세스할 수 있다.[14][15]

따라서 웜 탐지를 위해 프로그램이 실행될 때 어떤 시스템 콜이 호출되는지 알기 위해서는 커널모드에서 동작하는 프로세스를 사용해야하며, 본 논문에서는 시스템의 자원에 직접 액세스 가능한 디바이스 드라이버를 제작하여 프로세스가 동작 시 호출되는 시스템 콜을 모니터링하여 웜을 탐지하였다.

3.1 SSDT(System Service Dispatch Table)

SSDT는 시스템에서 이용 가능한 모든 시스템 서비스들의 주소를 가지고 있으며 프로세스에 의해 인터럽트 발생 시 운영체제는 이 테이블을 참고하여 적절한 결과 값을 돌려주게 된다. 이 테이블을 조작 / 변경 함으로써 시스템의 모든 서비스를 다룰 수 있기 때문에 커널 루트킷, 안

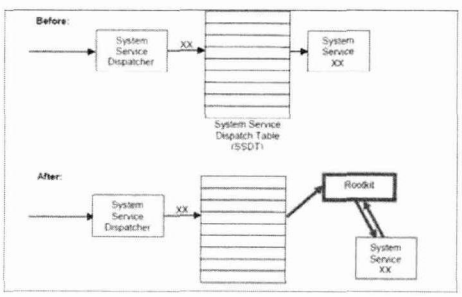


그림 1. SSDT Hooking[16]

티 바이러스 프로그램에 모두 사용된다.[14][15]

프로세스 동작 시 호출되는 시스템 콜을 분석하기 위하여 위 그림 1과 같이 SSDT에 루트 킷을 삽입한 후 시스템 콜이 호출되면 먼저 호출된 시스템 콜에 대한 정보를 저장한 다음 프로세스가 정상적으로 동작하도록 해준다.

3.2 시스템 구성

SSDT를 이용한 웜 탐지 시스템은 SSDT를 이용하여 시스템 콜을 캡처하기 위한 윈도우 디바이스 드라이버와 캡처된 시스템 콜 정보를 분석하여 웜인지 정상적인 프로세스인지를 판별하기 위한 소프트웨어로 구성된다.

그림 2는 SSDT를 이용한 웜 탐지 시스템의 구성도이다. 점선 사각형 부분은 프로세스로부터 호출되는 시스템 콜을 가상의 디바이스 드라이버로 모니터링하여 붉은 색의 유저모드에서 동작하는 분석 시스템으로 데이터를 전달하고 데이터 분석 시스템은 전달받은 데이터를 이용하여 시스템 콜을 한 프로세스가 정상적인 프로세스인지 웜인지 판별하게 된다.

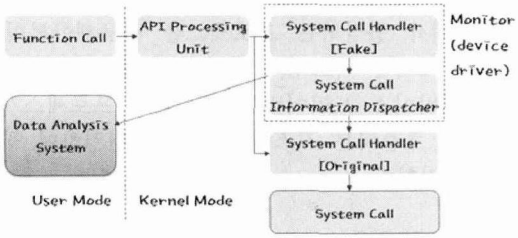


그림 2. 시스템 구성도

3.3 SSDT를 이용한 웜 탐지 시스템

SSDT를 이용한 웜 탐지 시스템은 커널모드에서 동작하는 가상의 디바이스 드라이버[17][18][19]를 이용하여 유저모드의 프로세스로부터 전달된 시스템 콜을 모니터링한다.

마이크로소프트의 윈도우는 디바이스 드라이버를 설계하기 쉽도록하기 위하여 WDM (Windows Driver Model) 모델을 제시하고 있다. 디바이스 드라이버는 DriverEntry Routine, AddDevice Routine, IRP Dispatch Routine, DriverUnload Routine의 4가지 엔트리[17][18][19]로 구성되며, 각 엔트리를 설명하면 다음과 같다.

- DriverEntry Routine : 디바이스 드라이버의 시작점. 드라이버가 로드된 후 처음으로 실행되는 부분. 드라이버가 로드될 때 단 한번 수행되므로 초기화 과정에 이용.
- AddDevice Routine : 새로운 디바이스를 추가하고자 할때 사용되는 부분. 새로운 Device Object를 생성하게 된다.
- IRP Dispatch Routine: 디바이스와 IO Manager 사이에서 명령을 전달하는 역할을 하는 구조체, 유저모드 프로세스에서 파일을 열고 쓰는 동작 등이 IRP를 통해 이루어진다.
- DriverUnload Routine : 드라이버가 종료될 때 수행되는 부분.

본 논문에서 제작한 가상의 디바이스 드라이버는 DriverEntry Routine에 SSDT를 액세스하여 정상적인 서비스의 주소를 시스템 콜이 호출될 때 시스템 콜을 모니터링 할 수 있는 주소로 변경하는 프로그램을 하고, AddDevice Routine에는 서비스가 호출되면 호출된 시스템 콜을 데이터화 하여 사용자 모드의 분석 시스템으로 전송한 후, 시스템 콜이 정상적으로 수행될 수 있도록 DriverEntry Routine에서 변경해 두었던 정상적인 서비스를 호출하는 프로그램을 추가한다. 마지막으로 DriverUnload Routine에는 DriverEntry Routine에서 변경했던 SSDT를 원래의 상태로 되돌리는 프로그램을 추가한다.

다음 그림 3은 DriverEntry Routine의 일부이다. 그림 3은 원래의 시스템 콜 서비스 함수의 주소를 저장하는 프로그램으로 시스템 콜을 모니터링 하기 위한 새로운 서비스 함수의 주소로 SSDT를 변경한 다음, 디바이스 드라이버가 종료될 때 SSDT를 다시 변경하기 위해 저장한다. 그림 3은 디바이스 드라이버의 DriverEntry Routine에 추가된다.

다음 그림 4는 시스템 콜을 모니터링하기 위한 디바이스 드라이버가 시스템에 로드되면서 변경되었던 시스템 콜의 주소를 원래의 주소로 되돌려 주는 프로그램의 일부로 디바이스 드라이버의 DriverUnload Routine에 추가된다. 다음 그림 5는 시스템 콜을 모니터링 한 다음, 서비스가 정상적으로 수행될 수 있도록 SSDT를 변경하기 전의 시스템 콜을 다시 호출해주기 위한 프로그램의 일부분이다. 그림 5의 프로그램은 디바이스 드라이버의 AddDevice Routine에 추가되며, DriverEntry Routine에서 저장해 두었던 원래의 시스템 콜 서비스 함수의 주소를 이용한다.

```
...
oldZwWriteFile = (ZWWRITEFILE)SYSTEMSERVICE(ZwWriteFile);
oldZwOpenFile = (ZWOPENFILE)SYSTEMSERVICE(ZwOpenFile);
oldZwCreateFile = (ZWCREATEFILE)SYSTEMSERVICE(ZwCreateFile);
oldZwQueryDirectoryFile = (ZWQUERYDIRECTORYFILE)SYSTEMSERVICE(ZwQueryDirectoryFile);
oldZwCreateSection = (ZWCREATESECTION)SYSTEMSERVICE(ZwCreateSection);
oldZwMapViewOfSection = (ZWMAPVIEWOFSECTION)SYSTEMSERVICE(ZwMapViewOfSection);
oldZwSetInformationFile = (ZWSETINFORMATIONFILE)SYSTEMSERVICE(ZwSetInformationFile);
...
```

그림 3. 원래의 시스템 콜 서비스 함수의 주소를 저장

```
...
(ZWWRITEFILE)InterlockedExchange((PLONG)&SYSTEMSERVICE(ZwWriteFile),(ULONG)old-
ZwWriteFile);
(ZWOPENFILE)InterlockedExchange((PLONG)&SYSTEMSERVICE(ZwOpenFile),(ULONG)oldZwOpenFile);
(ZWCREATEFILE)InterlockedExchange((PLONG)&SYSTEMSERVICE(ZwCreateFile),(ULONG)old-
ZwCreateFile);
(ZWQUERYDIRECTORYFILE)InterlockedExchange((PLONG)&SYSTEMSERVICE(ZwQueryDirectoryFile),(
ULONG)oldZwQueryDirectoryFile);
(ZWCREATESECTION)InterlockedExchange((PLONG)&SYSTEMSERVICE(ZwCreateSection),(ULONG)old-
ZwCreateSection);
(ZWMAPVIEWOFSECTION)InterlockedExchange((PLONG)&SYSTEMSERVICE(ZwMapViewOfSection),(UL
ONG)oldZwMapViewOfSection);
(ZWSETINFORMATIONFILE)InterlockedExchange((PLONG)&SYSTEMSERVICE(ZwSetInformationFile),(UL
ONG)oldZwSetInformationFile);
...
```

그림 4. 시스템 콜 서비스 함수의 주소를 원래 시스템 콜 서비스 함수의 주소로 되돌림

```
...
NTSTATUS status;
PUCHAR szProcessName;
PEPROCESS CProcess;
CProcess = (PEPROCESS)PsGetCurrentProcess();

szProcessName = CProcess->ImageFileName;
status = oldZwWriteFile( //원래함수를호출해서정상적으로처리되게함
        FileHandle, Event, ApcRoutine, ApcContext, IoStatusBlock, Buffer, Length, ByteOffset, Key);
...
```

그림 5. 시스템 콜 모니터링 후 정상적인 서비스 호출

3.4 SSDT를 이용한 웜 탐지 시스템 구현 예

SSDT를 이용한 웜 탐지 시스템은 시스템 콜을 모니터링하기 위한 가상의 디바이스 드라이버와 모니터링한 데이터를 분석하기 위한 분석

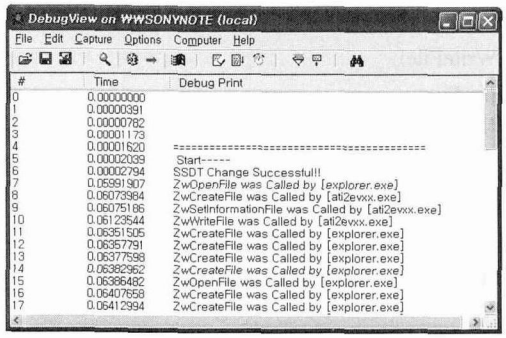


그림 6. 시스템 콜 모니터링 디바이스 드라이버 시작 후 모니터링 예

시스템으로 이루어졌다. 디바이스 드라이버는 instDrv[20]를 이용하여 로드하였으며, 디바이스 드라이버에서 출력하는 메시지를 보기위한 도구로 DebugView[21]를 사용하였다.

다음 그림6, 7은 정상적으로 로드된 시스템 콜 모니터링 드라이버의 메시지 출력 화면이다.

3.5 SSDT를 이용한 웜 탐지

본 논문에서 제작한 SSDT를 이용한 웜 탐지 시스템은 GSR(Gene of Self Replication)[13]구조를 이용하여 웜을 탐지하는 방법을 사용한다. 본 논문에서 제작한 시스템의 웜 탐지 가능성을 판단하기 위하여 윈도우 실행 파일에 기생하여 실행시에 윈도우 디렉토리에서 .exe 파일을 찾아 파일의 entry 포인터의 변화시에 복제를 일으키는 I-Worm.Xanax 기생형 웜의 GSR구조를 이용하였으며, 웜의 시스템 콜은 위 표 1과 같다. 위 표 1의 첫 번째 NtOpenFile 시스템콜의 입력의 12는 디렉토리 핸들을 나타낸다. 이는 다음 NtCreateFile 시스템 콜을 호출할 때 사용되며 이 첫 번째 두 번째 단계의 시스템 콜은 웜의 code generation 부분에 해당된다. NtOpeFile의 출력의 12는 네 번째 NtQueryDirectoryFile에서 사용되며 세 번째 네 번째 단계는 code integration 부분에 해당된다. 2번째 NtCreateFile의 input arguement인 68은 바이러스 포인트 핸들로 5번째 NtCreateSection의 input argument로 사용된다. 또한 output argument인 72는 세션핸들로 NtMapViewOfSection 시스템콜에서 사용되며 이 5번째 6번째 콜은 ready for transmission 부분에 해당된다. 7번째 NtCreateFile의 output argument인 52는 8번째 9번째 시스템콜에서 사용되며 이 시스템 콜들에 의해서 복제가 일어나게 되며 마지막 9번째 시스템 콜에서 바이러스성 코드와 코드 사이즈가 input argument에 나타나게 된다. code generation, code integration, ready for transmission, replication occured의 모든 과정을 시퀀스로 모니터링하여 모든 과정이 차례대로 다 나타날 때 웜이 자기 복제를 하기 위한 과정으로 웜의 탐지가 가능하다. 이렇듯 시스템 콜의 각 input argument와 output argument를 고려하여 시스템 콜을 모니터링 함으로써 자기 복제가 발생할 때의 웜을 탐지해 낼 수 있음을 알 수 있다.

표 1. I-Worm.Xanax 시스템 콜

	system calls		input argument	output argument
1	NtOpenFile	100020h	{24,0,42h,0,0,"\??\c:\Virlab\"},3,33 ···12,0h,1) result=0	
2	NtCreateFile	80100080h	{24,12,42h,0,1243404,"Xanax.exe"},0h,128,3,1,96,0,0···,68,0h,1) result=0	
3	NtOpenFile	0x100001	{24,0,0x400,0,"\??\C:\WINDOWS\""},3,16417	12,{0x0,1}
4	NtQueryDirectoryFile	12	0,0,0,1243364,616,3,1,"<.exe",0	{0x0,110}
5	NtCreateSection	0xfoo1f	0h,0h,2,134217728,68	72
6	NtMapViewOfSection	72	-1,0h,0,0,{0,0},0,1,0,2	0x980000,0,0,36864
7	NtCreateFile	0x40110080	{24,0,40h,0,1242788,"\??\C:\WINDOWS\Clac.exe"},0h,32,0,5,100,0	52,{0h,3}
8	NtSetInformationFile	52	1241948,8,20	{0h,0}
9	NtWriteFile	52	0,0,0,"Mz\220\0\3\0\0\0\4\0\0\0\377\37···\0",33792,0h,0	{0h,33792}

V. 결 론

본 논문에서는 웜을 통한 공격으로부터 네트워크 인프라를 보호하기 위해서 SSDT(System Service Dispatch Table)를 이용한 웜 탐지 시스템을 설계 및 구현하였다. 구현된 시스템은 SSDT를 커널모드에서 액세스하여 시스템 콜을 모니터링 하는 가상의 디바이스 드라이버와 모니터링된 데이터를 분석하는 분석 시스템으로 구성된다.

모니터링 된 데이터는 GSR[13]구조에 따라 분석하였으며, 자기 복제 성질을 갖는 웜의 GSR을 이용하여 시스템의 웜 탐지 여부를 시뮬레이션 하였다.

하지만, 본 논문에서 구현된 시스템은 실제 웜이 있는 환경에서 동작되지 않았고, 다양한 웜으로 시험 해 보지 않았으므로, 다양한 웜의 탐지 가능성은 예측할 수 없다.

향후, 시스템을 보완하고 웜 모니터링과 탐지를 위한 GSR구조를 보완하여, 실제 환경에서 다양한 웜의 탐지가 가능한지 테스트하여야 할 것으로 사료되며, 또한 다양한 웜 탐지가 가능할 경우 웜의 감염을 차단할 수 있는 백신 기능 추가도 고려하여야 할 것으로 사료된다.

참 고 문 헌

[1] Snort: an open source network intrusion prevention and detection system, 2005. http://www.snort.org

[2] Bro Intrusion Detection System, 2003. http://www.bro-ids.org

[3] J.Newsome, B. Karp, and D.Song. Polygraph : Automatically Generating Signatures for Polymorphic Worms. *InProceedings of 2005 IEEE Symposiumon Securityand Privacy*, pp. 226-241, May2005.

[4] Z. Li, M.Sanghi, Y. Chen, M. Kao, and B. Chavez. Hamsa : Fast signature generation for zero-day polymorphic worms with provable attack resilience. *In Proceedings of 2006 IEEE Symposium on Security and Privacy*, pp. 32-47, May 2006.

[5] U.Payer, M.Lamberger, and P.Teufl. Hybrid engine for polymorphic code detection. *In Proceedings of the Conference on Detection of Intrusions and Malware & Vulnerability Assessment*, pp. 19-31

[6] R.Chinchani and E. Berg. A Fast Static Analysis Approach To Detect Exploit Code Inside Network Flows. *In Proceedings of the 8th International Symposium on Recent Advances in Intrusion Detection*, pp. 284-308, September 2005.

[7] P.Akritidis, E.Markators, M.Polychronakis, and K.Anagnostakis. STRIDE: Polymorphic Sled Detection through Instruction Sequence Analysis. *In Proceedings of the 20th IFIP International Information Security Conference*, pp. 375-392, June 2005.

[8] X.Wang, C.Pan, P.Liu, and S.Zhu. SigFree : A Signature-free buffer Overflow Attack Blocker. *In Proceedings of the 15th USENIX security Symposium*, pp. 225-240, July 2006.

[9] M.Polychronakis, K.Anagnostack, and E. Markatos. Network-Level Polymorphic Shellcode Detection Using Emulation. *In Proceedings of the Conference on Detetion of Intrusions and Malware and Vulnerability Assessment*, July 2006.

[10] Qinghua Zhang, Douglas S. Reeves, Peng Ning, S.Purushothaman Iyer. Analyzing Network Traffic To detect self-decrypting exploit code. *In Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security*, pp. 4-12, March 2007.

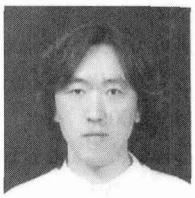
[11] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T.A.Longstaff. A Sense of Self for Unix Processes, *In Proceedings of 1996 IEEE Symposium on Computer Security and Privacy*, 1996.

[12] Jiang N., Hua K., and Sheu S. Considering Both Intra-pattern and Inter-pattern Anomalies in Intrusion Detection. *Proc. Intel. Conf.* Data Mining, 2002.

[13] V.Skormin, A.Volynkin, D.Summerville and J.Moronski. Prevention of information attacks by run-time detection of self-replication in computer codes. *Journal of Computer Security 15*, pp. 273-301, 2007.

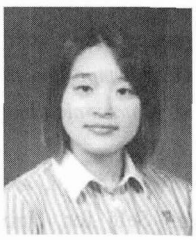
〔14〕 http://lucid7.egloos.com/
〔15〕 김도균외 역, Microsoft Windows Internal, *정보문화사*, 2005
〔16〕 http://www.vigilantminds.com/
〔17〕 김성은, 이태현 공역, Windows 2000 디바이스 드라이버, *정보문화사*, 2004
〔18〕 Art Baker, Jerry Lozano, The Windows 2000 Device Driver Book: A Guide for Programmers, *Prentice Hall*, 2001
〔19〕 박햇님역, 윈도우즈 드라이버 모델 WDM, *에이콘*, 2002
〔20〕 http://www.rootkit.com
〔21〕 http://www.sysinternals.com

황 유 동 (Yu-dong Hwang) 정회원



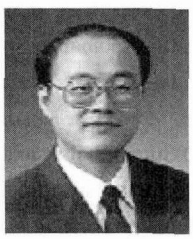
1998년 2월 순천향대학교 제어계측 공학과 공학사
2000년 8월 순천향대학교 전기전자공학과 석사
2003년~현재 순천향대학교 전기전자공학과 정보보호전공 박사과정
<관심분야> 네트워크 보안, 시스템 보안

이 유 리 (You-ri Lee) 정회원



2002년 2월 순천향대학교 정보통신공학과 공학학사
2004년 2월 순천향대학교 정보통신공학과 공학석사
2004년~현재 순천향대학교 정보통신공학과 박사과정
<관심분야> 접근제어, 보안

박 동 규 (Dong-gue Park) 정회원

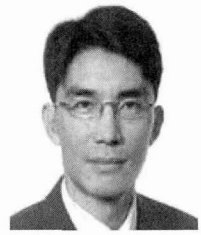


1992년 한양대학교 대학원 전자공학과 공학박사
1999~2003년 순천향대학교 정보기술공학부 부교수
2004년~현재 순천향대학교 정보통신공학과 교수
<관심분야> 네트워크 보안, 유비쿼터스 컴퓨팅 보안

임 황 빈 (Hwang-bin Yim) 정회원

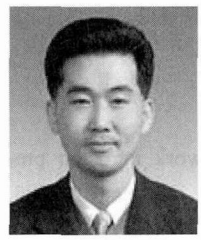
2003년 순천향대학교 전기전자공학과 (공학박사)
2003년~현재 강원도립대학 정보통신과 조교수
<관심분야> 통신응용시스템, 광통신, 정보 보호

오 진 태 (Jin-tae Oh) 정회원



1990년 경북대학교 전자공학과 학사
1992년 경북대학교 전자공학과 석사
1992년~1998 한국전자통신 연구원 선임연구원
2003년~현재 한국전자통신연구원 네트워크보안 연구팀 팀장/선임 연구원
<주관심분야> 네트워크보안, 비정상행위 탐지기술

장 종 수 (Jong-soo Jang) 정회원



1984년 경북대학교 전자공학과 학사
1986년 경북대학교 전자공학과 석사
2000년 충북대학교 박사
1989년~현재 한국전자통신연구원 네트워크보안 그룹 그룹장/책임연구원
<주관심분야> 네트워크보안, 정책기반 보안관리기술