

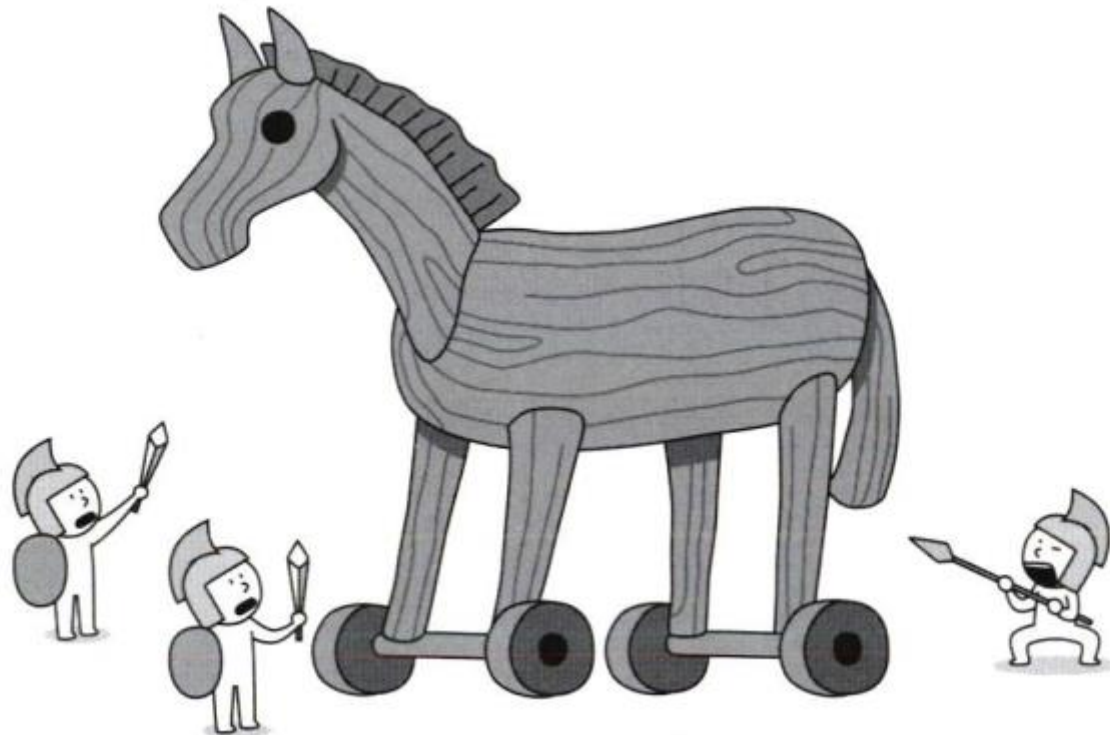
악성코드 2

목차

- 트로이 목마
- 악성코드 분석과 제거
- 소프트웨어 개발 보안

트로이 목마

- 정상적인 프로그램처럼 위장 하여 동작하면서 내부적으로는 시스템의 기밀 정보를 공격자의 컴퓨터로 빼돌리는 악성코드



트로이 목마

표 10-4 트로이 목마의 예

이름	사용 포트	설명
넷버스 (Netbus)	12345	사용하기 쉬워 가장 많이 사용되는 트로이 목마 중의 하나
백오리피스 (Back Orifice)	31337	가장 유명하여 제거 툴이 가장 많은 트로이 목마
Executor	80	감염된 컴퓨터의 시스템 파일을 삭제하거나 시스템을 파괴하는 트로이 목마 중의 하나
Striker	2565	부팅이 안 되게 하는 등 감염된 컴퓨터를 아예 망가뜨려버리는 트로이 목마

악성코드 분석과 제거

- 정적 분석 : 실행x 분석
- 동적 분석 : 실행 o 분석

정적 분석

- 정확한 악성코드 여부를 판단 하기 위해 프로그램 소스가 필요
but 해커가 공개 x

-> 정적 분석의 시작은 실행파일에서 소스 역추출부터 시작(디스어셈블, 리버싱)

IPA Pro 라는 유료 프로그램으로 분석

정적 분석

- 분석을 통해 악성행위를 하는지 판단하는 가장 좋은 방법
-> 의심되는 함수 호출, 의심되는 문자열이 있는지 확인

-->> 실제로 많은 악성코드에 취약점을 스캔하거나
프로세스를 강제종료 시키는 소스가 들어있다.

동적 분석

- 악성코드가 실제로 실행되기 때문에 통제된 환경이 필요 -> 가상머신
 - 1. 악성코드가 실행 되었을 때 작업관리자 확인(독특한 이름일 경우 구글에 검색)
 - 2. cpu 자원을 많이 소비하는 프로세스 확인
(SSDT후킹 기술로 작업관리자에서 안 보이도록 할 수도 있다)
- > 공격자가 프로그램을 숨기기 위한 목적으로 사용되는 프로그램 -> 루트킷

동적 분석

- 3. 등록된 시작 프로그램 확인(마이크로소프트 사 오토런즈)

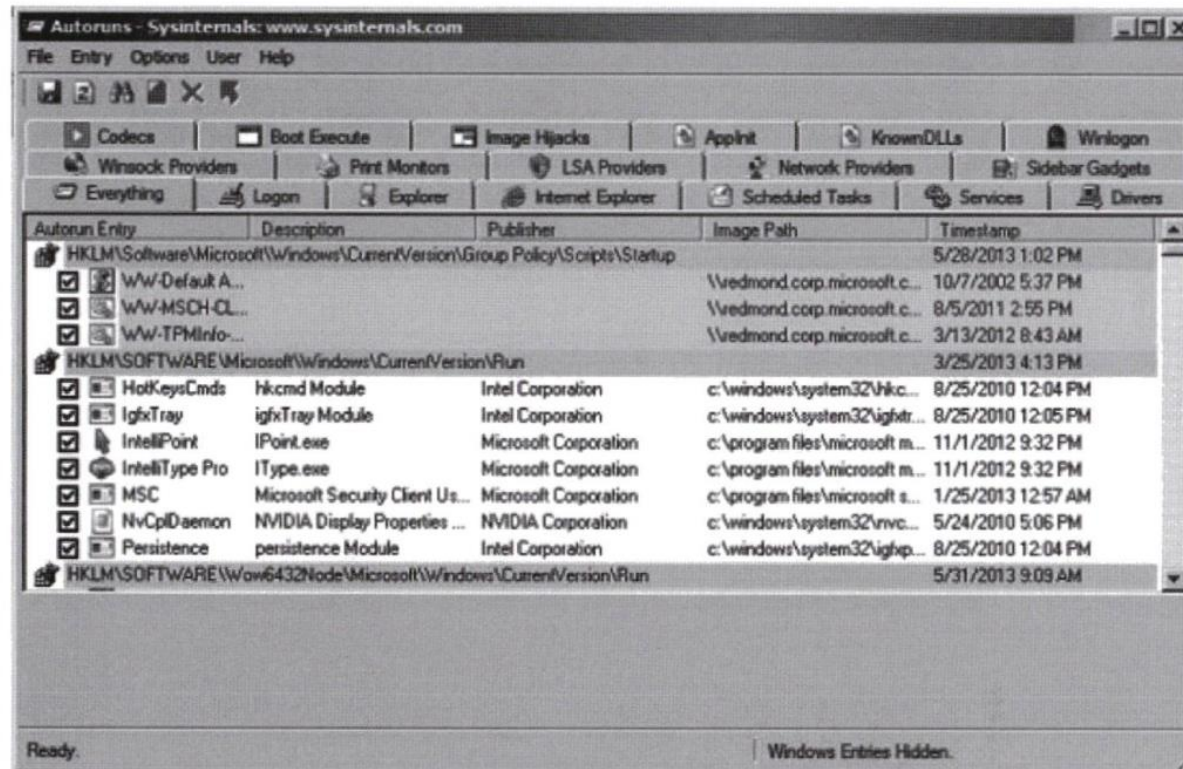
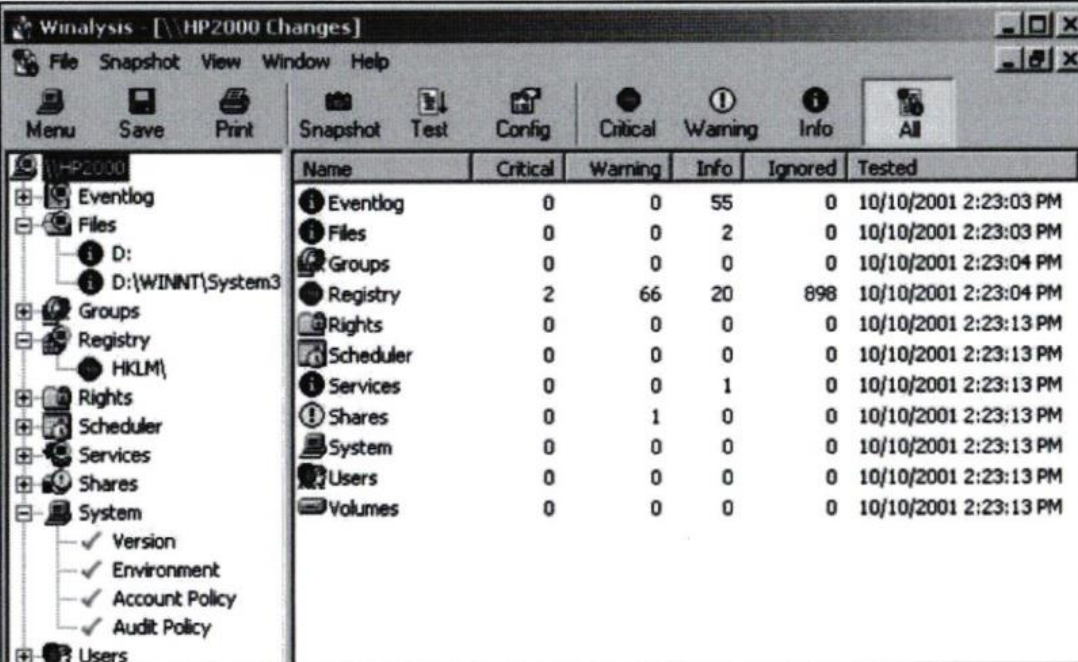


그림 10-29 등록된 시작 프로그램을 확인하는 오토런즈

동적 분석

- 4. 파일과 레지스트리의 변경내용 확인(Winalysis)
- 윈도우 운영체제에서 레지스트리는 운영체제와 응용 프로그램의 모든 설정값을 저장 해놓은 저장소이기 때문에 파일과 마찬가지로 어떤 값을 변경시켰는지 분석해야한다.



The screenshot shows the Winalysis application window titled "Winalysis - [\\HP2000 Changes]". The left pane displays a tree view of system components, including Eventlog, Files, D:, D:\\WINNT\\System3, Groups, Registry (with HKLM\\ expanded), Rights, Scheduler, Services, Shares, System, Version, Environment, Account Policy, Audit Policy, and Users. The right pane contains a table with columns: Name, Critical, Warning, Info, Ignored, and Tested. The table lists various system components and their corresponding counts and timestamps.

Name	Critical	Warning	Info	Ignored	Tested
Eventlog	0	0	55	0	10/10/2001 2:23:03 PM
Files	0	0	2	0	10/10/2001 2:23:03 PM
D:	0	0	0	0	10/10/2001 2:23:04 PM
D:\\WINNT\\System3	0	0	0	0	10/10/2001 2:23:04 PM
Groups	0	0	0	0	10/10/2001 2:23:04 PM
Registry	2	66	20	898	10/10/2001 2:23:04 PM
Rights	0	0	0	0	10/10/2001 2:23:13 PM
Scheduler	0	0	0	0	10/10/2001 2:23:13 PM
Services	0	0	1	0	10/10/2001 2:23:13 PM
Shares	0	1	0	0	10/10/2001 2:23:13 PM
System	0	0	0	0	10/10/2001 2:23:13 PM
Users	0	0	0	0	10/10/2001 2:23:13 PM
Volumes	0	0	0	0	10/10/2001 2:23:13 PM

그림 10-30 Winalysis를 통한 파일과 레지스트리 변경 내용 확인

동적 분석

- 5. 파일, 레지스트리, 네트워크의 실시간 모니터링(Process Monitor)

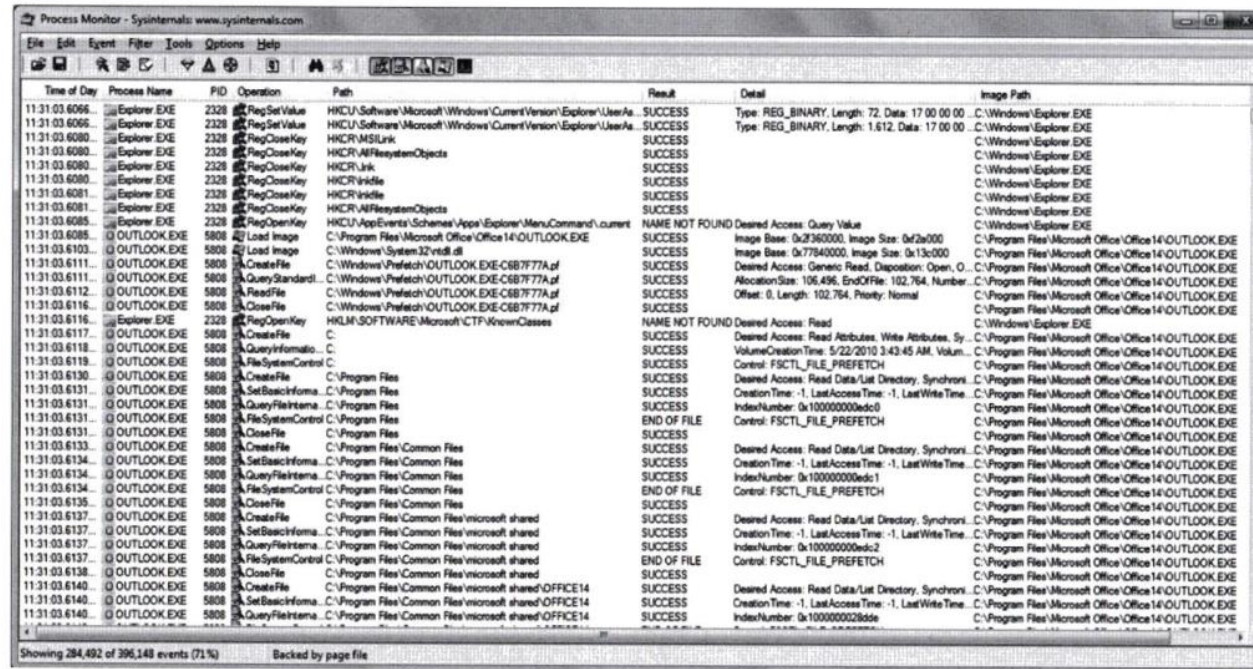


그림 10-31 프로세스 모니터를 통한 파일과 레지스트리의 실시간 확인

악성코드 제거

- 프로세스 종료 후 삭제
- 파일 삭제와 레지스트리 키/값 삭제
- 안전모드 부팅후 삭제

소프트웨어 개발 보안

- 소프트웨어 보안 <-> 안전한코딩, 시큐어코딩

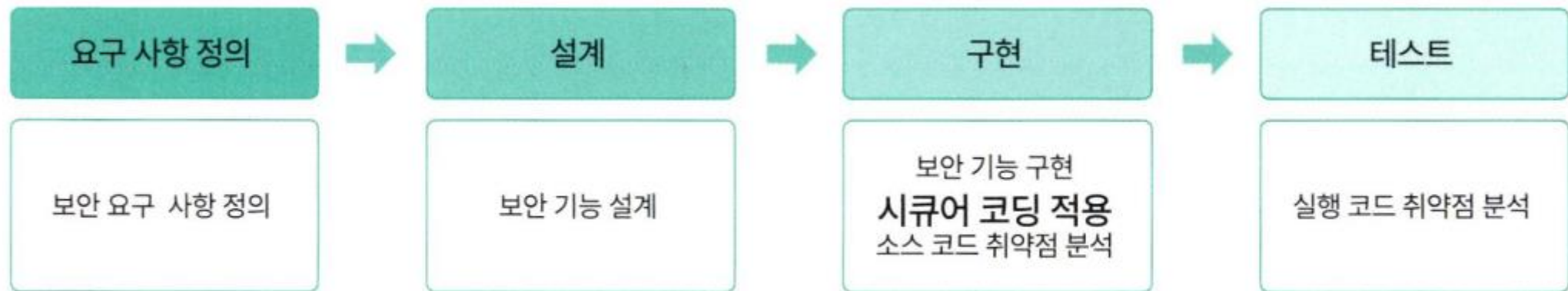


그림 10-37 소프트웨어 개발 보안 과정

소프트웨어 개발 보안

- 메모리 버퍼 오버플로 공격
- 프로그램이 실행되는 도중에 메모리 오류가 발생하여 의도하지 않는 동작이 일어나는 보안취약점

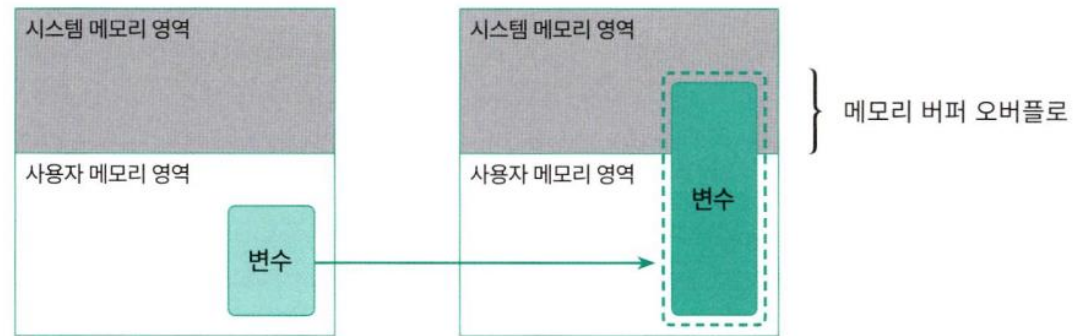


그림 10-38 메모리 버퍼 오버플로의 기본 개념

소프트웨어 개발 보안

- 보안대책 1 : 입력 값에 대한 길이(크기검증)
- 보안대책 2 : 오버플로에 안전한 라이브러리 사용
- 보안대책 3 : 정적소스 분석 도구로 미리 수정하기

```
05: if (argv[1] != NULL && strlen(argv[1]) < sizeof(buffer) )  
06:     strcpy(buffer, argv[1]);
```

표 10-6 버퍼 오버플로 관련 문자열 처리 함수

안전하지 않은 함수	안전한 함수
strcpy()	strcpy_s()
strcat()	strcat_s()
strncat()	strncat_s()
strncpy()	strncpy_s()
sprintf()	sprint_s()

소프트웨어 개발 보안

- 포맷 스트링 공격
- 포맷 문자열을 이용하여 프로그램의 특정 메모리 내용을 읽거나 쓰는 보안공격기법

```
04: printf( "%s", argv[1] ); // 보안상 안전한 코드
```

【실행 예】

```
a.out addr-%x [엔터]
```

```
addr-%x
```