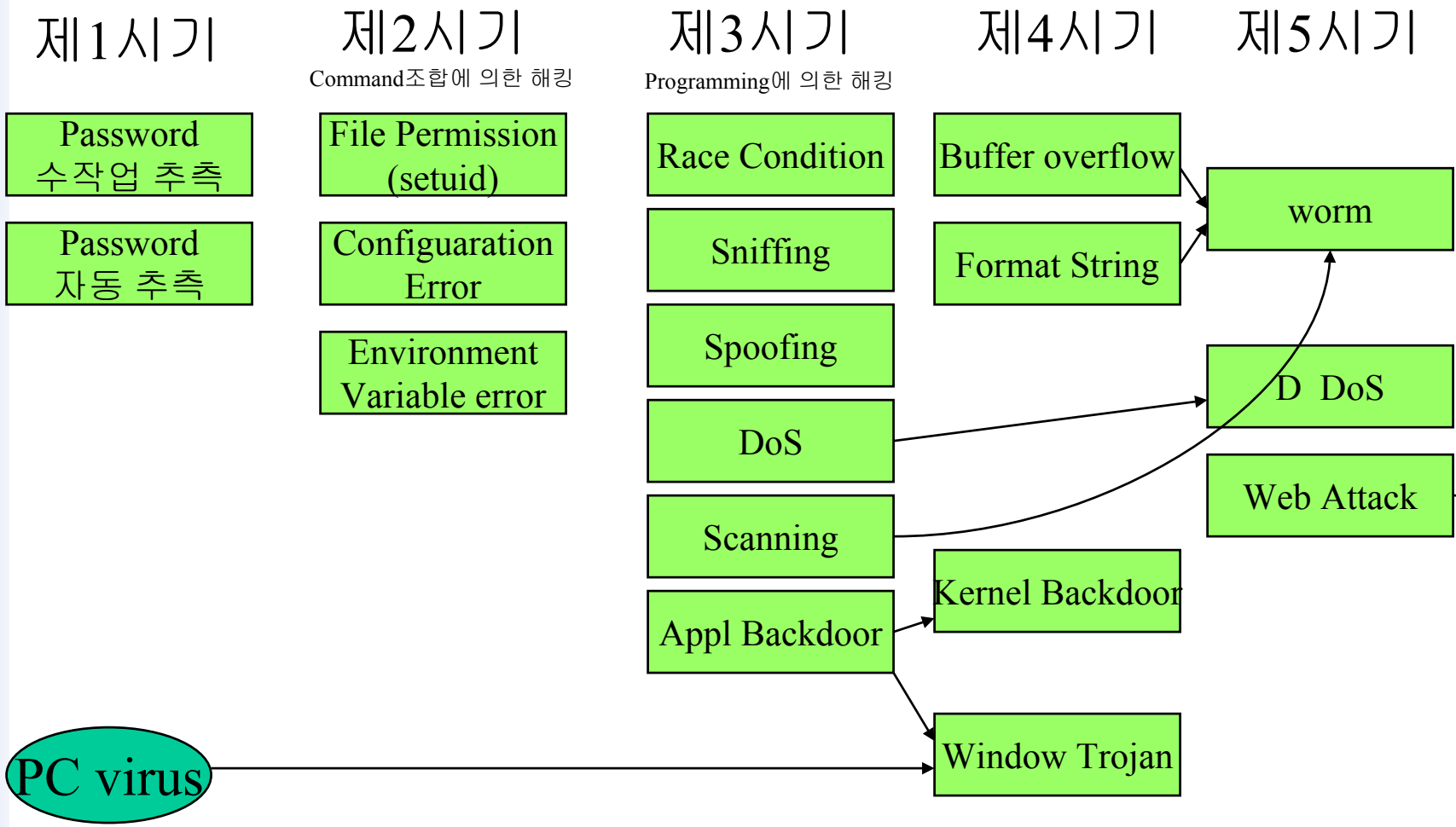
해킹 공격방법의 이해

양 정 규
amy@securesoft.com

Securesoft / Pen-Test team

목 차

공격 방법의 변천


제1시기
제2시기
Command조합에 의한 해킹
제3시기
Programming에 의한 해킹
제4시기
제5시기
Password 수작업 추측
Password 자동 추측
File Permission (setuid)
Configuaration Error
Environment Variable error
Race Condition
Sniffing
Spoofing
DoS
Scanning
Appl Backdoor
Buffer overflow
Format String
Kernel Backdoor
Window Trojan
worm
D DoS
Web Attack
PC virus

공격 방법의 변천

제6시기

- P2P messenger Attack
- Advanced BO Attack
- Wireless Attack
- Advanced Kernel Backdoor
- DB Attack

지금 시기는 제5시기에서 제6시기로 변화중
Blackhat영역의 시기와 Whitehat영역의
시기의 불일치 현상 존재

Blackhat & Whitehat간의 시기불일치 현상

- underground에서 연구하고 있는 해킹기법과 (black영역) 실제 internet현장(white영역)에서 대규모 발생하고 있는 해킹기법간에는 약 1시기의 차이가 있다.
- 현재 blackhat의 집중 연구 분야
  - wireless attack
  - DB SQL injection attack
  - Advanced Buffer Overflow Attack (Free malloc overflow)
- whitehat영역의 실제 피해 기법
  - internet worm (CodeRed, nimda, Window Trojan)
  - Distributed Denial Of Service Attack
  - Web Application Attack (홈페이지 공격)

Blackhat & Whitehat간의 시기불일치 현상

- 이러한 시기 불일치 현상의 결과
  - whitehat영역의 관심사와 blackhat영역의 관심사가 다름
  - whitehat영역의 경우
    - 이전 시기의 공격기법에 대한 대책에 관심
  - blackhat영역의 경우
    - 새로운 시기의 공격기법자체에 대해 관심
- 컴퓨터 보안 영역의 이원화
  - 보안제품 개발자, 시스템 네트웍 관리자의 경우 whitehat 영역의 기술을 다룬다
  - 취약점 점검, 보안진단, 버그분석자의 경우 blackhat 영역의 기술을 다룬다
- programming에 의한 hacking이 나오기 시작한 제3시기이후 두영역은 더욱더 이원화됨

기술 관점에 따른 공격기법

- 정보획득 기법 (Scanning)
- Network 통신 취약점을 이용한 공격 (TCP/IP오류)
- System 관리 잘못을 이용한 공격
- software 구현상의 오류를 이용한 공격 (“bug”)
- 은닉 기법 (backdoor)
- 서비스 제공 방해 공격 (DoS)
- 자동 공격 & 침투 기법 (worm, trojan)

Scanning

- Reachable IP address
  - Ping sweep
- Running TCP/UDP services
  - Port scan
- System architecture & Oeration System type
  - OS indentification
- DoS와 더불어서
  IDS 로그 발생원인의 대다수를 차지

ICMP echo request

- ICMP is defined by RFC 792.
- ICMP ECHO request (ICMP type 8) Packet 을 목적 시스템에 전송
- 그에 대한 응답으로 ICMP ECHO reply (ICMP type 0) Packet이 도달하였는지를 체크
- 목적 IP가 살아있는지의 여부를 결정

ICMP echo request

nmap –sP –n 192.168.192.255/24

Starting nmap V. 2.53 by fyodor@insecure.org (www.insecure.org/nmap/)

Host (192.168.192.0) seems to be a subnet broadcast address (returned 1 extra pings).

Still scanning it due to positive ping response from its own IP.

Host (192.168.192.1) appears to be up.

Host (192.168.192.3) appears to be up.

……..

……..

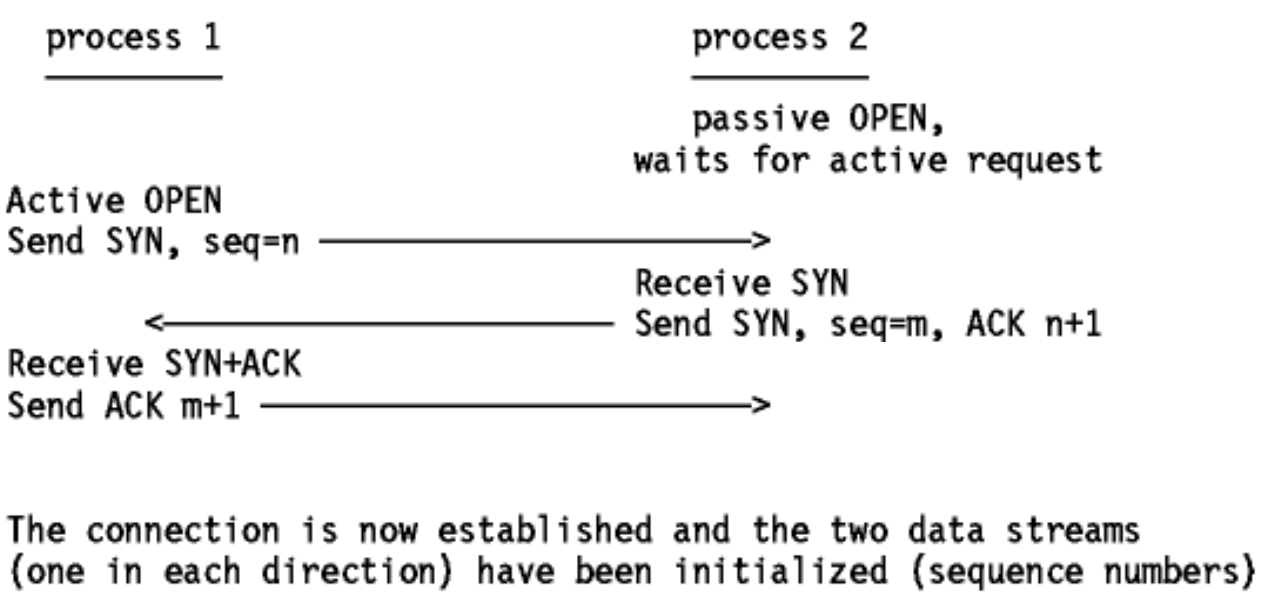
nmap run completed -- 256 IP addresses (108 hosts up) scanned in 5 seconds

Port Scan

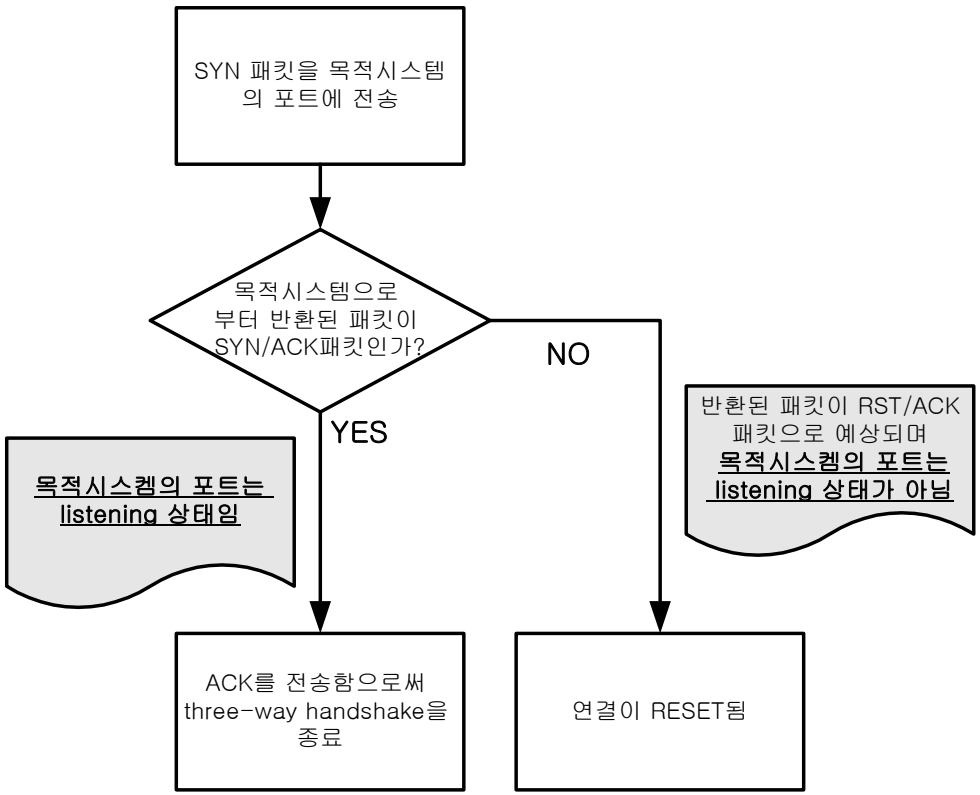
- Ping Sweep이후, 목적 시스템의 TCP나 UDP 포트에 연결을 통해서 어떤 서비스가 실행중이거나 listening상태에 있는지를 탐지
- tcp connect() scan
- tcp syn scan (half-open scan)
- stealth scan
- udp scan

TCP port scan

- TCP 3-way handshaking



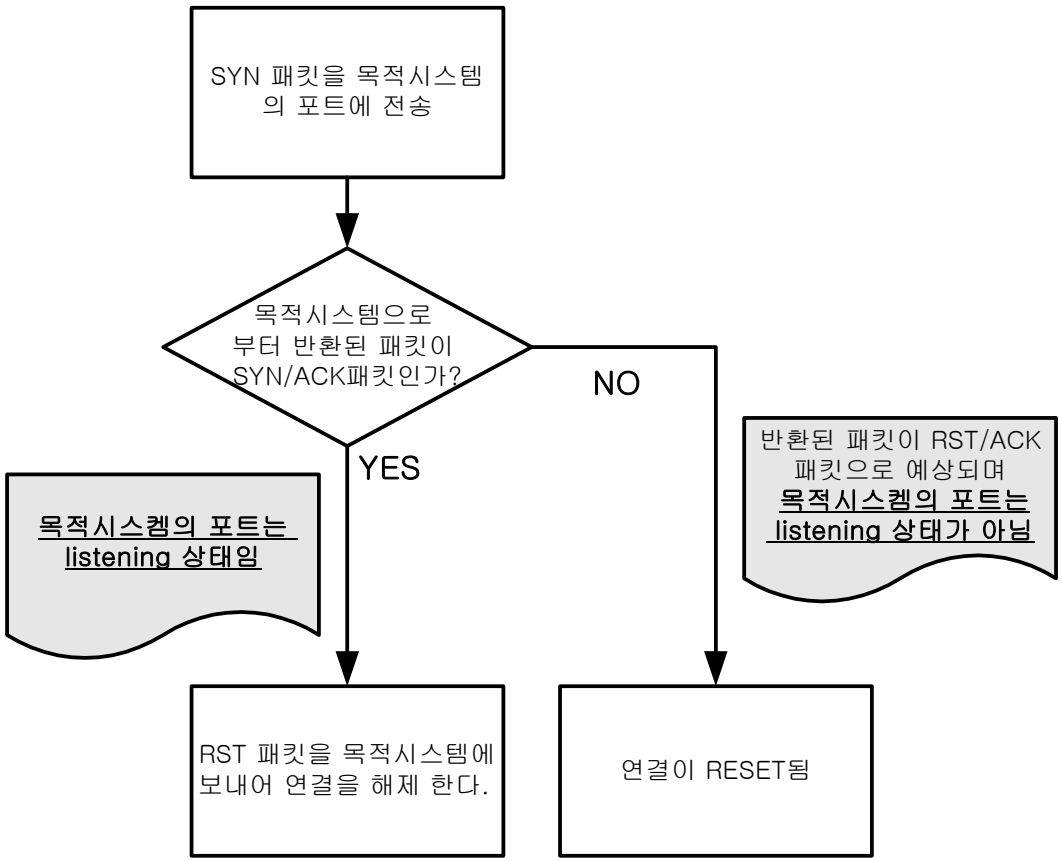
TCP connect() scan



TCP connect() scan

- Most basic form of tcp scan.
- need NOT root permission.
- scan by using many sockets in parallel. -> speedy scan.
- Using non-blocking I/O allows you to set a low time-out period and watch all the sockets at once. -> speedy scan.
- 단점은 목적시스템의 로그에 남기 때문에 쉽게 탐지될 수 있고 필터링이 가능

TCP syn scan


SYN 패킷을 목적시스템의 포트에 전송
목적시스템으로 부터 반환된 패킷이 SYN/ACK패킷인가?
NO
YES
반환된 패킷이 RST/ACK 패킷으로 예상되며 목적시스켐의 포트는 listening 상태가 아님
목적시스켐의 포트는 listening 상태임
RST 패킷을 목적시스템에 보내어 연결을 해제 한다.
연결이 RESET됨

TCP syn scan

- often referred to as "half-open" scanning
- you don't open a full TCP connection
- Return SYN|ACK indicates the port is listening.
- Return RST is indicative of a non- listener.
- You need root privileges to build these custom SYN packets.
- 로그에 잡힐 확률이 적음.

TCP Stealth scan

- 비정상적인 tcp패킷을 던져주고 그에 따른 반응을 수집하여 port scan.
- SYN/ACK, FIN, XMAS, NULL flag 패킷을 보냈을때
  - Closed port는 RESET패킷으로 반응
  - Open port는 무반응(무시)

SYN/ACK scan

- 의도적으로 TCP three-way handshake를 무시
- TCP three-way handshake의 첫 번째 단계를 생략하고 두 번째 단계인 SYN/ACK 패킷을 먼저 보내서 그 반응을 통해 정보를 획득
- Closed port
  - Target 시스템은 SYN 패킷이 전송되지 않았음을 알고 보내어진 패킷을 오류로 판단하여 RESET대답
  - 시스템의 존재와 해당포트가 closed임을 알수있음
- Open port
  - 보내어진 SYN/ACK패킷 무시(무반응)

FIN scan

- FIN 패킷을 보낸후, 시스템의 반응을 통해 정보를 획득
- Closed port
  - Target 시스템은 SYN 패킷이 전송되지 않았음을 알고 보내어진 패킷을 오류로 판단하여 RESET대답
  - 시스템의 존재와 해당포트가 closed임을 알수있음
- Open port
  - 보내어진 SYN/ACK패킷 무시(무반응)

nmap –sF 203.239.110.1

XMAS scan, NULL scan

- XMAS scan
  - URG, ACK, PST, RST, SYN, FIN 플래그를 모두 세팅한 패킷을 보내는 스캐닝기법

  # nmap –sX 203.239.110.1
- NULL scan
  - 모든 플래그를 Off 한 패킷을 보내는 스캐닝 기법

  # nmap –sN 203.239.110.1

UDP Port Scan

- UDP port (RFC 768) 이용
- send 0 byte udp packets to each port on the target machine.
- ICMP port unreachable 메시지가 return
  - closed port라고 판단
- 무반응일경우
  - open port라고 판단
- 신뢰성이 적음

Port scan의 보조기법

- Random port scan
  - 순차적인 포트 연결시도 탈피
- Slow scan
  - 룰셋 시간내 시도패킷수를 최소화
- Decoy scan
  - Source address spoof

OS Identification

- Banner Grabbing
  - ALZZA Linux release 6.0 (May the Force be with you)
    Kernel 2.2.5-22 on an i586
    login:
  - # echo 'GET / HTTP/1.0\n' | nc hotbot.com 80 | egrep '^Server:'
    Server: Microsoft-IIS/4.0
- TCP/IP Stack Fingerprinting
  - TCP 스택을 구현할 때, 각 OS vendor별로 조금씩 수치가 다르게 구현된 것을 이용하여 특정 TCP 패킷을 보내고 반환되는 응답을 관찰하여 원격시스템의 운영체제를 알아내는 기법

nmap –O 203.239.110.1

TCP Sequence Prediction: Class=random positive increments

Difficulty=5721956 (Good luck!)

Remote operating system guess: Linux 2.1.122 - 2.2.14

TCP/IP Stack Fingerprinting

- FIN probe
- Bogus flag probe
- TCP Initial sequence number sampling
- Don't Fragment Bit
- TCP Initial Window
- ACK Value
- ICMP Error Message Quenching
- ICMP Message Quoting
- ICMP Error Message Echoing Integrity
- Type of Service (TOS)
- Fragmentation Handling
- TCP options

TCP initial number sampling

- 연결요청에 응답할 때 TCP 구현에 따라 선택되어 지는 Initial sequence number의 패턴이 틀림.
- 오래된 버전의 유닉스 : 64K씩 증가
- FreeBSD, DG-UX, IRIX, Solaris : 시간에 비례하여 증가
- Linux : 완전 random

Network 통신 취약점 공격

- TCP/IP의 개방성과 효율을 위한 단순함에 기인한 문제들
  - ICMP echo reply amplifier
    - Block broadcast ping
  - TCP SYN flooding
    - TCP/IP stack tunning
  - Sniffing
    - Switch 사용으로 sniffing확률을 줄임
    - 데이터 암호화 이외에 완전한 defense 해법 없음
  - Spoofing
    - IP Spoofing, DNS Spoofing, Domain Spoofing, ARP Spoofing
    - 특수경우를 제외하고는 완전한 해법/점검방법 없음
  - Session Hijacking

System 관리잘못을 이용한 공격

- Account 도용
- Password Cracking
- File / Directory Permission Error (예: .rhosts)

- 최근에는 거의 피해사례 없음

Software 구현상의 오류를 이용한 공격

- 운영체제 또는 Application을 개발한 개발사의 프로그래머 잘못에 의한 문제점
- “bug” 혹은 “Vulnerability”를 이용한 공격
- Bug (좁은 의미의 vulnerability)
  - 1990년 이래 약 6000건 이상의 보안관련 bug가 존재 (Securityfocus Vulnerability database)
  - 발표되는 bug의 95%는 trivial
  - 실제 whitehat영역에서 피해사례로 체감할수 있는 bug는 전체 발표 bug의 5% 미만

Software 구현상의 오류를 이용한 공격

- "whitehat영역에서 피해사례로 체감"의 의미
  - Exploit가 용이
  - 많은 시스템이 이 bug에 적용
  - Bug가 많이 알려져 있음
  - 대책을 강구하기가 어려움
  - 웜형태, remote공격, rootshell획득 공격

Software 구현상의 오류를 이용한 공격

- 경계영역 문제 (boundary condtion error)
  - Buffer Overflow
    - Stack Overflow
      - Stack buffer Overflow, Frame Pointer Overflow
    - Heap Overflow
      - Heap buffer Overflow, Free/Malloc Overflow
- 접근권한 인증 실패 (access validation error)
- 입력데이터 확인 오류 (input validation error)
  - Format String Attack
  - 웹페이지 파일 업로드 공격
  - 웹페이지 중요정보 공개 공격
  - Cross Site Script 공격
- Race Condition

Software 구현상의 오류를 이용한 공격

- 예외상황 처리 오류
  - 웹페이지 중요정보 공개 오류
- 실행 환경 오류
- 패키지 설치당시의 configuration의 오류
- Object 자동 실행

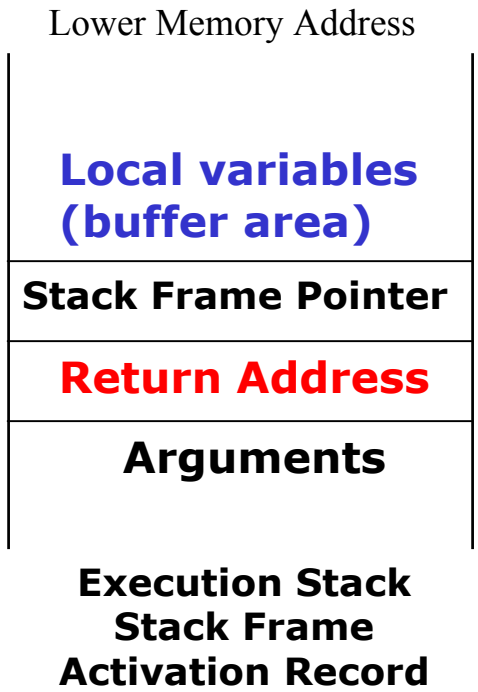
- Boundary Condition 문제
  - 프로그램을 실행중에 입력가능한 변수에, 그 변수를 위해 할당된 메모리 영역보다 더 큰 사이즈의 데이터를 입력했을때, 그 변수영역이 넘치게 되고 가까이 있는 return address가 변조되게 되어 보안상 문제가 있는 instruction을 실행
  - 소위, 버퍼오버플로우라고 알려져 있는 수많은 취약점들
  - OS vendor의 패치발표 이후 점검 툴로 점검 가능
- Access validation 문제
  - Ex)웹서버에 접속할 경우 사용자 인증에 관련된 정보를 유지하기 위해, 특정 파일에 인증정보를 저장하는데, 이 파일을 유출하거나 위조할경우, 타 사용자가 이를 이용하여 원래 사용자처럼 웹서버에 접속할 수 있음
  - 점검 툴로 점검 불가능

- Input Validation 문제
  - EX) format string bug의 경우 프로그래머가 프로그램을 개발할때 "printf" 계열 함수를 사용시 포맷argument를 지정을 제대로 하지 않은경우에, 만약 사용자가 포맷문자를 포함하고 있는 문자를 만들어 안전하지 않은 printf 함수에 사용하게 되면 메모리 참조범위를 벗어나 보안상 문제가 있는 instruction을 실행
  - Format string bug으 경우OS vendor의 패치발표 이후 점검 툴로 점검 가능
  - 웹페이지에 대한 input validation error는 점검툴로 점검 불가능
- Race Condition 문제
  - 보안상 문제가 있는 System Program과 Cracker의 Exploit Program이 Race(경쟁) Condition(상태)에 이르게 하여 System Program이 갖는 권한으로(Set-User ID가 붙은 경우 Root, Bin등 …) File에 대한 Access를 가능
  - OS vendor의 패치 발표 이후 점검 툴로 점검 가능

- Exceptional Handling 관련 문제
  - 대부분의 프로그램이 구현될때에는 어떤 기능을 사용하기 위해서는 사용자가 어떤 행위를 해야되는 식으로 구현이 되어있으나, 만약 개발자가 예측하지 않았고, 예외처리를 제대로 구현하지 않은 행위를 사용자가 할 경우 프로그램은 보안상 문제가 있는 instruction을 실행
  - OS에 딸린 실행프로그램의 경우 vendor의 패치발표 이후 점검툴로 점검 가능
  - Web홈페이지의 경우 점검툴로 점검 불가능
- Package Initial Configuration 문제
  - 패키지 설치시 초기 설정환경이 보안상 문제가 있는 경우, 설정환경을 바꿔주기 전에 해커의 공격이 있을수 있음
  - OS vendor의 패치 발표 이후 점검 툴로 점검 가능

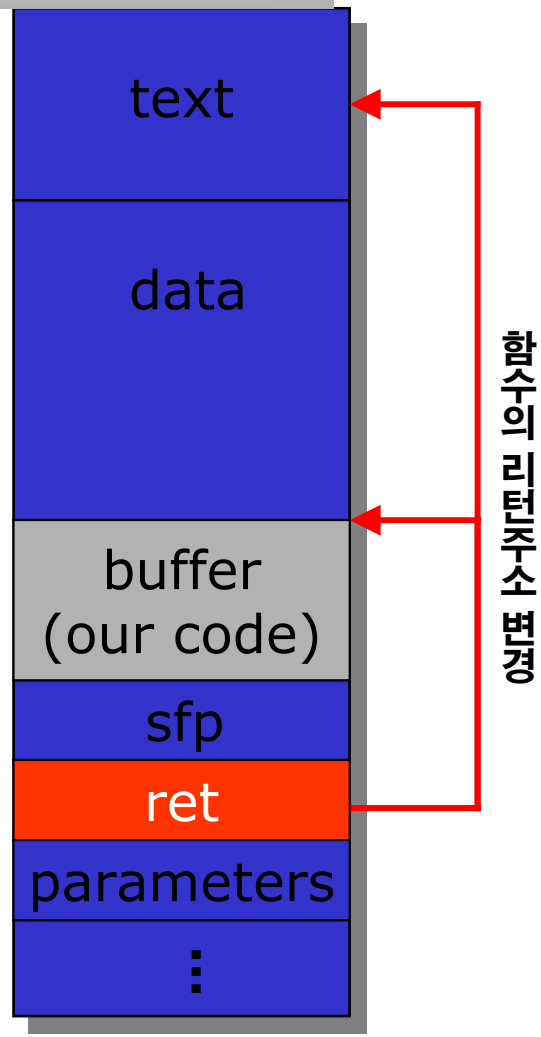
Buffer Overflow

- Buffer overflow ?
  - 컴파일러가 배열의 경계검사(Boundary Check)를 하지 않아 선언된 크기보다 더 큰 데이터를 기록함으로써 발생되는 현상
  - 운영체제가 스택이나 힙 영역에 임의의 데이터 기록 및 실행을 허용함으로써 발생되는 현상



Buffer Overflow

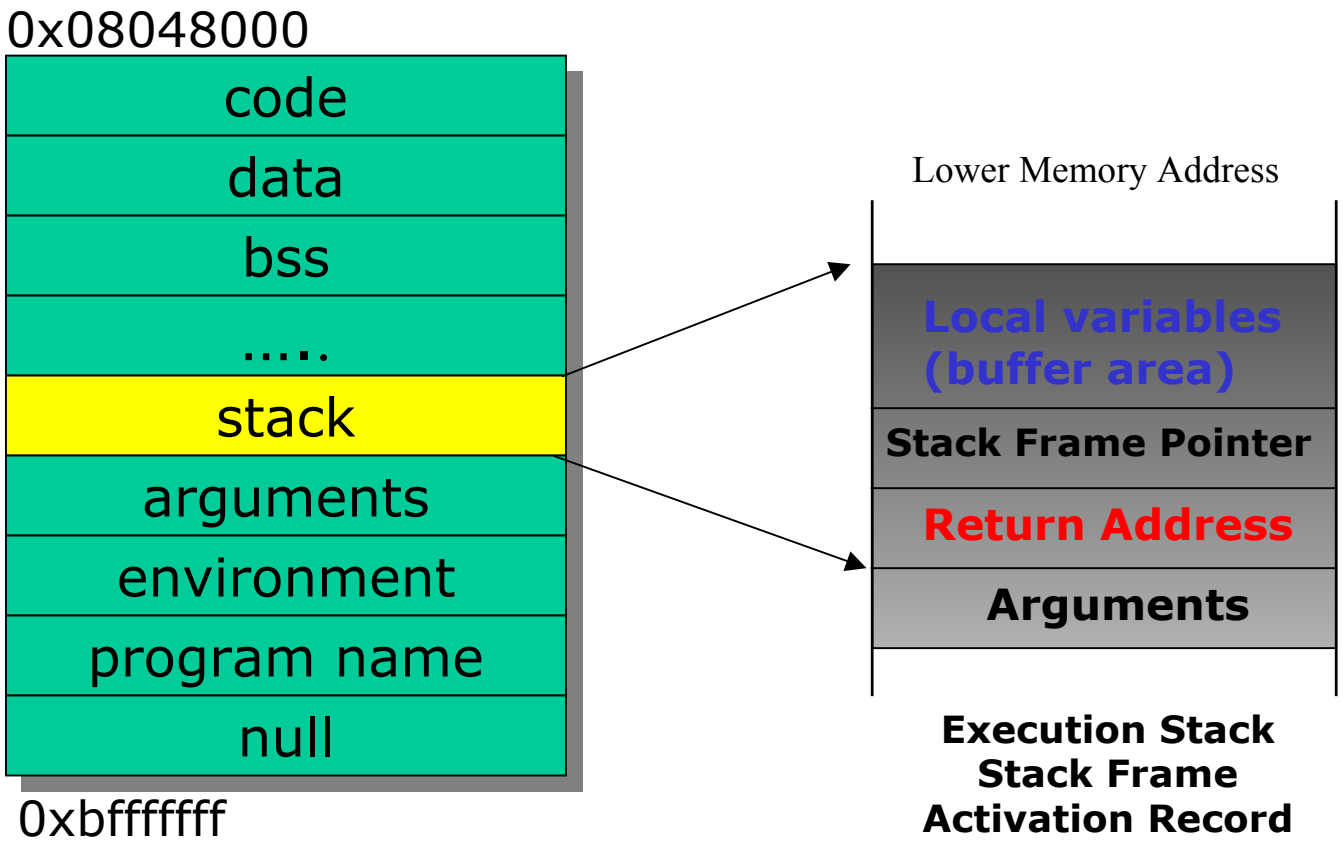
- 공격의 대상
  - 취약점이 있는 서버 데몬 및 시스템 관리 프로그램
  - 주로 root 소유의 suid 프로그램
- 공격의 절차
  - 취약점 탐지 및 정보수집
    - OS, Program, Version, etc
    - Exploit code from the well-known security portal sites
  - 혹은 직접 Exploit 프로그램 작성
    - 로컬 및 리모트 공격용 쉘 코드 작성
  - Let's exploit!



Intel Architecture32 CPU

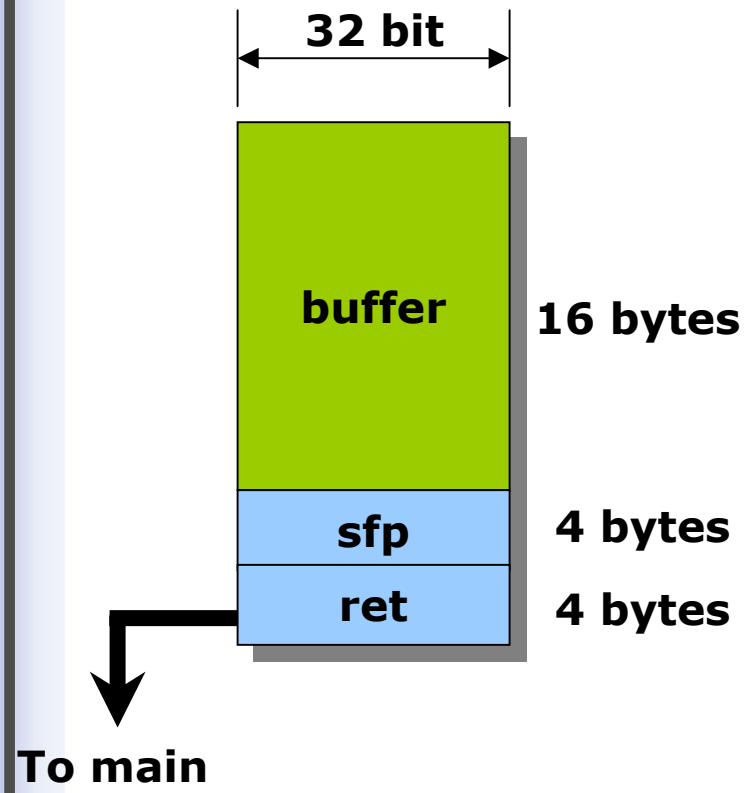
- Stack?
  - 연속된 메모리 공간으로서 Last In First Out (LIFO) 구조
  - 함수의 인자 및 결과값 전달 및 지역 변수용으로 사용
  - 메모리 영역 중에서 최상위 부분에 위치
  - 메모리 하단 방향으로 증가
  - IA-32에서는 4바이트씩 정렬
  - 따라서 1 바이트 변수도 스택 4바이트 사용
- Registers
  - ESP - 스택의 Top을 가리킴
  - EBP -호출된 procedure를 위한 스택 프레임 내의 고정reference point를 나타냄. 저장된 이전의 EBP 값을 sfp라고 함
  - EIP - 다음에 수행할 명령어의 위치를 가리킴

Stack Structure

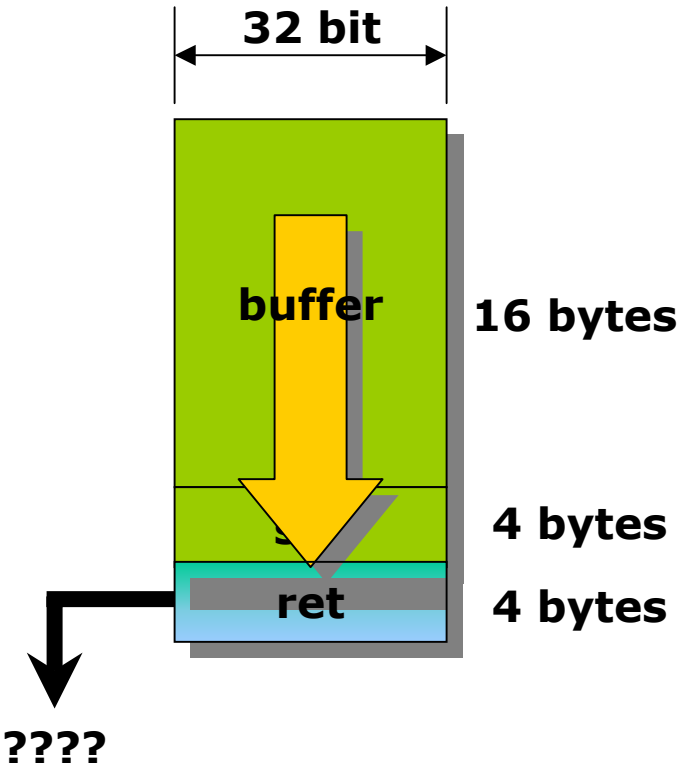


Stack Structure

Overflow 전

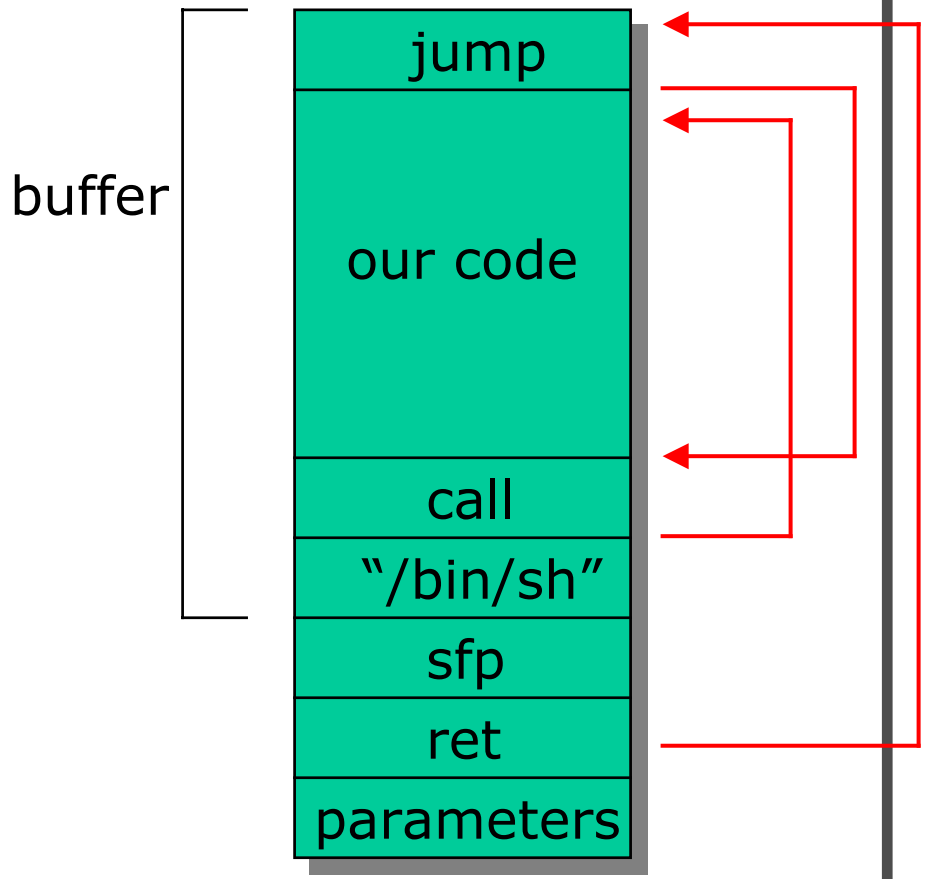


Overflow 후



Spanwing Shell Code

- shellcode란?
  - /bin/sh를 실행시키는 기계어 코드
- shellcode 특징
  - 메모리 상의 shellcode의 위치를 모르므로 상대번지만 사용
  - NULL 문자는 버퍼에 복사되지 않으므로 0x00 코드는 다른 해당 기계어 코드로 대체
  - CPU 마다 다르게 제작



Spawning Shell Code

- Shell code의 원형

```
#include <stdio.h>

void main()
{
   char *name[2];

   name[0] = "/bin/sh";
   name[1] = NULL;
  execve(name[0],name,NULL);
```

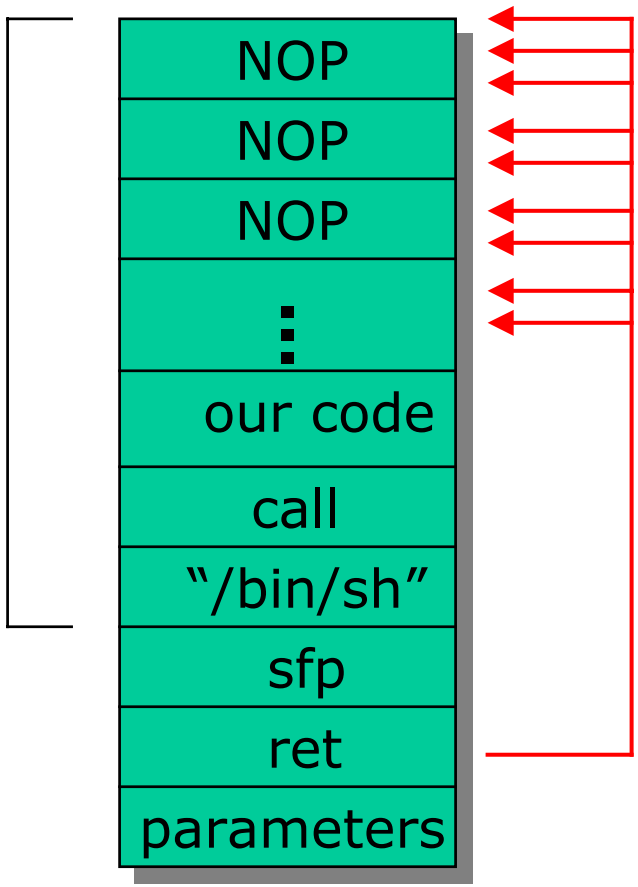
- 완성된 shell code

```
jmp     0x1f                # 2 bytes
popl    %esi                # 1 byte
movl    %esi,0x8(%esi)      # 3 bytes
xorl    %eax,%eax           # 2 bytes
movb    %eax,0x7(%esi)      # 3 bytes
movl    %eax,0xc(%esi)      # 3 bytes
movb    $0xb,%al            # 2 bytes
movl    %esi,%ebx           # 2 bytes
leal    0x8(%esi),%ecx      # 3 bytes
leal    0xc(%esi),%edx      # 3 bytes
int     $0x80               # 2 bytes
xorl    %ebx,%ebx           # 2 bytes
movl    %ebx,%eax           # 2 bytes
inc     %eax                # 1 bytes
int     $0x80               # 2 bytes
call    -0x24               # 5 bytes
.string \"/bin/sh\"         # 8 bytes
```

실전에서의 Buffer Overflow

- 정확한 ret addr을 구하기 힘들기 때문에 NOP 코드를 이용

Q: Where is our code ?
A: Maybe…



Buffer Overflow 대책

- 운영체제 설정차원에서의 대책
  - 사용자 스택 영역에 데이터 기록 금지
    - 함수로부터 복귀할 때 스택의 무결성(integrity) 검사
  - 코드 수행 금지
    - Linux Kernel patch from the Openwall Project
      - http://www.openwall.com/linux/
    - Solaris 2.6 이상 버젼: prevent and log stack-smashing attack
      - /etc/system 수정
      - set noexec_user_stack = 1
      - set noexec_user_stack_log = 1

Buffer Overflow대책

- 보안성있는 컴파일러 및 링크 사용
  - GNU GCC 2.7.2.3 버전을 패치한 StackGuard
    - http://www.cse.ogi.edu/DISC/projects/immunix/StackGuard
    - 복귀주소 다음에 “canary” word를 위치시킴
    - “canary” word가 변경되면 스택 오버플로우 공격의 시도 감지 및 보고(syslog) , 시스템 종료.
      - Random canary
        - » 실행할 때마다 canary value를 변경시켜, 공격자가 예측하지 못하도록 함
      - Null canary(0x00000000),
        - » 공격자가 버퍼에 널 문자(0x00)를 넣을 수 없다는 점을 이용
      - Terminator canary(combination of Null, CR, LF, -l)
        - » NULL 문자로 끝나지 않는 몇몇 문자열 함수의 문자열 끝문자 이용

Buffer Overflow대책

- 즉각적인 보안 패치
  - RedHat Linux http://www.redhat.com/apps/support/updates.html
  - MS http://support.microsoft.com
  - BSD 관련 http://www.freebsd.org/security/index.html
  - Sun Solaris http://sunsolve.sun.com
  - Digital ftp://ftp.compaq.com/pub
  - HP/UX http://us-support3.external.hp.com
  - IBM AIX ftp://software.watson.ibm.com/pub
  - SGI IRIX ftp://ftp.sgi.com/security

Buffer Overflow 원천방지

- 프로그래머의 관점에서의 보안 대책
  - Boundary를 검사하는 컴파일러 및 링커 사용
  - Boundary를 검사하는 함수 사용
    - 사용 자제 함수들
      - strcat(), strcpy(), gets(), scanf(), sscanf(), vscanf(), vsscanf(), sprintf(), vsprintf(), gethostbyname()
    - 사용 권장 함수들
      - strncat(), strncpy(), fgets(), fscanf(), vfscanf(), snprintf(), vsnprintf()

Buffer Overflow 원천방지

Example of Vulnerable Program

insecure.c

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(int argc, char **argv)
{
      char buffer[1024];
      if(argc > 1) {
            strcpy(buffer, argv[1]);
      }
      printf("buffer: %s\n",buffer);
}
```

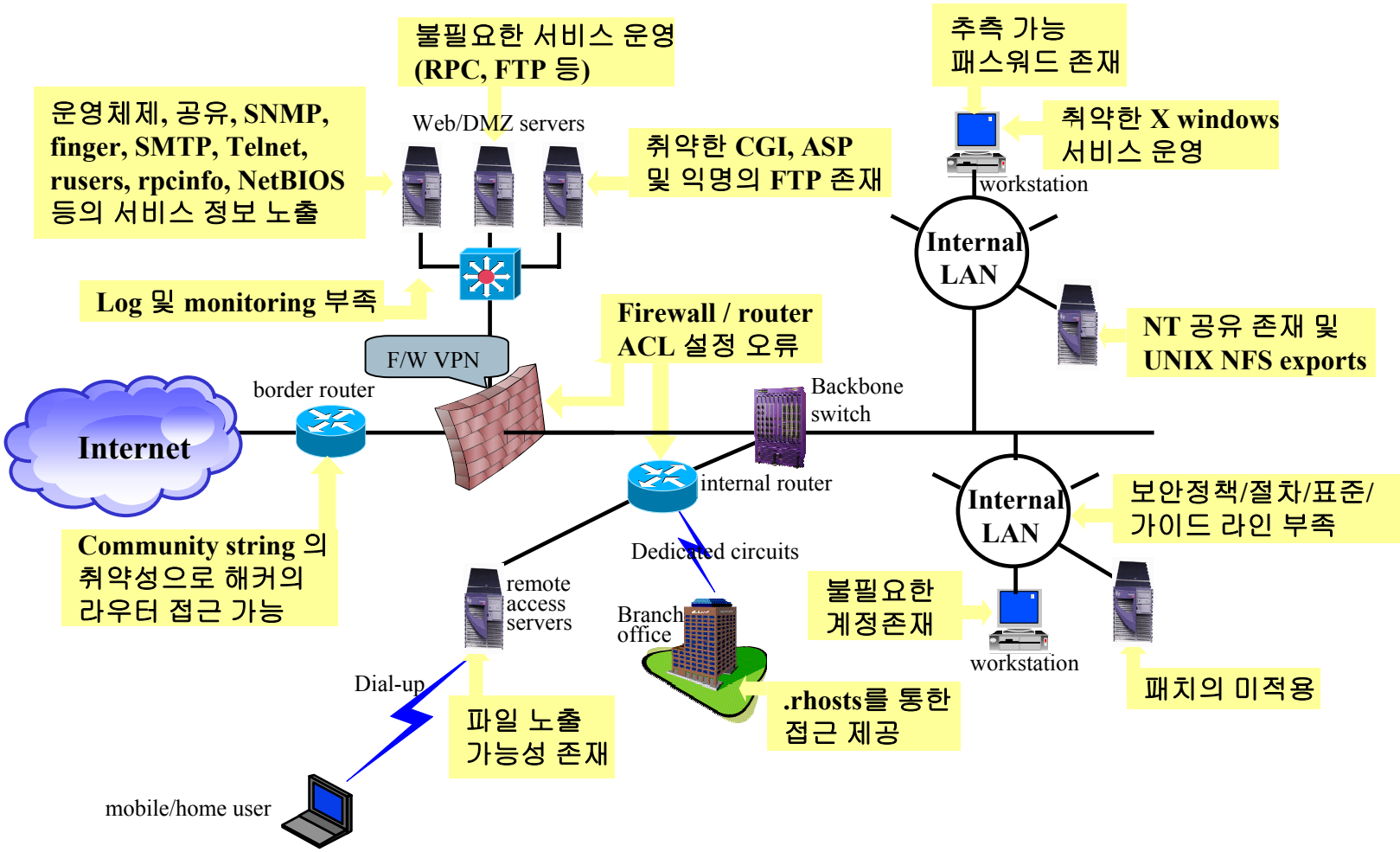
Example of Secure Program (patch)

secure1.c

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

int main(int argc, char **argv)
{
   char buffer[1024];
   int i;
   if(argc > 1) {
     if(strlen(argv[1]) >=1023) {
        fprintf(stderr,"too long\n");
        exit(0);
     }
     strcpy(buffer, argv[1]);
   }
   printf("buffer: %s\n", buffer);
}
```

Q & A


불필요한 서비스 운영
(RPC, FTP 등)
추측 가능
패스워드 존재
운영체제, 공유, SNMP,
finger, SMTP, Telnet,
rusers, rpcinfo, NetBIOS
등의 서비스 정보 노출
Web/DMZ servers
취약한 CGI, ASP
및 익명의 FTP 존재
취약한 X windows
서비스 운영
workstation
Internal
LAN
Log 및 monitoring 부족
Firewall / router
ACL 설정 오류
F/W VPN
NT 공유 존재 및
UNIX NFS exports
border router
Backbone
switch
Internet
internal router
Internal
LAN
보안정책/절차/표준/
가이드 라인 부족
Community string 의
취약성으로 해커의
라우터 접근 가능
Dedicated circuits
remote
access
servers
Branch
office
불필요한
계정존재
workstation
Dial-up
파일 노출
가능성 존재
.rhosts를 통한
접근 제공
패치의 미적용
mobile/home user