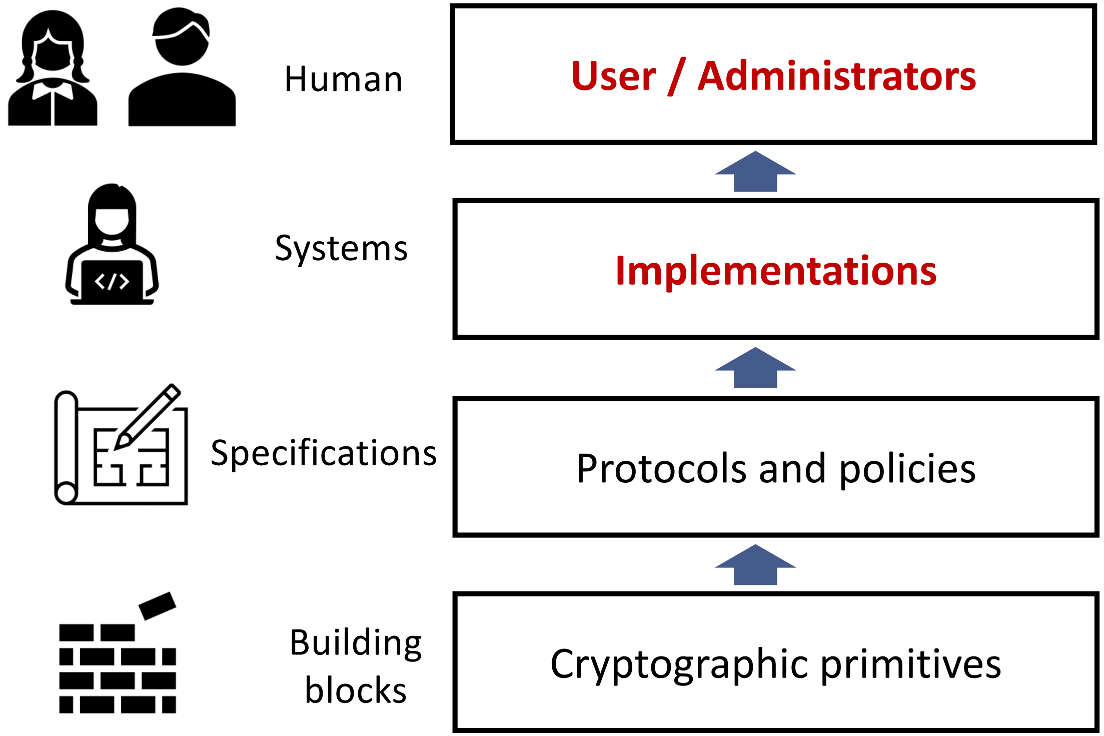
IT-Security

Chapter 10: Malware and Binary Exploitation

Prof. Dr.-Ing. Ulrike Meyer

Overall Lecture Context

- **So far, we mainly looked at**
  - Secure cryptographic building blocks
  - Design of security protocols
  - Users / Administrator when it comes to password selection
- **Now we look at**
  - **Implementation** vulnerabilities
  - **Social engineering**



Overview

Malware Types by Spreading

- Viruses, Worms, Trojans

Initial Infection

- Software Vulnerabilities
- Misconfigured access controls
- Vulnerable Authentication
  - Weak passwords
  - Protocol weaknesses
- Social engineering

Botnets

- C&C Infrastructures
- Taking down Botnets

Typical Payloads

- DDoS Engines
- SPAM Engines
- Phishing Engines
- Information Stealing
- Miners

Definition

Malware = Malicious Software

- According to NIST SP 800-83:

“A program that is inserted into a system, usually covertly, with the intent of compromising the confidentiality, integrity or availability of the victim’s data, applications, or operating system or otherwise annoying or disrupting the victim”

Owner of the system and victim do not necessarily coincide

Motivation to Write Malware

- Experimenting how to write malware
- Testing own programming skills
- Get famous
- Vandalism
- Fighting authorities
- Direct Financial gain
- Corporate Espionage
- Combatting crime and terrorism
- Cyberwar

Simple Example for Malicious Code

- **Attacker writes a small shell script on a UNIX system:**

```
cp /bin/sh /tmp/.xyz
chmod u+s,o+x /tmp/.xyz
rm ./ls
ls $*
```

- **Attacker saves this script in a file called "`ls`" and tricks a victim user into executing it**
- **This leads to a copy of the shell in a hidden file** `.xzy`
- **Shell executable by anyone with the userid set to who-ever-executed-the-script**
  - If who-ever-executed-the-script acted as root, shell will be a root shell executable by anyone
- **To the victim user, the result will look the same as result of real `ls`**
  - Script removes itself

Malware Types with respect to Spreading

Trojan Horse

- Program with an
  - overt purpose known to the user
  - covert purpose unknown to the user
- Typically installed by the user itself

Worm

- Program that actively seeks for machines to infect
- Infects new machines by exploiting one or more software vulnerabilities
- Uses network connections, shared media email,…to spread from one machine to another

Virus

- Software fragment that attaches to an existing executable
- Can replicate itself from one infected executable to another

Overview

Malware Types by Spreading

- Viruses
- Worms
- Trojans

Botnets

- C&C Infrastructures
- DGAs
- Sinkholing

Initial Infection

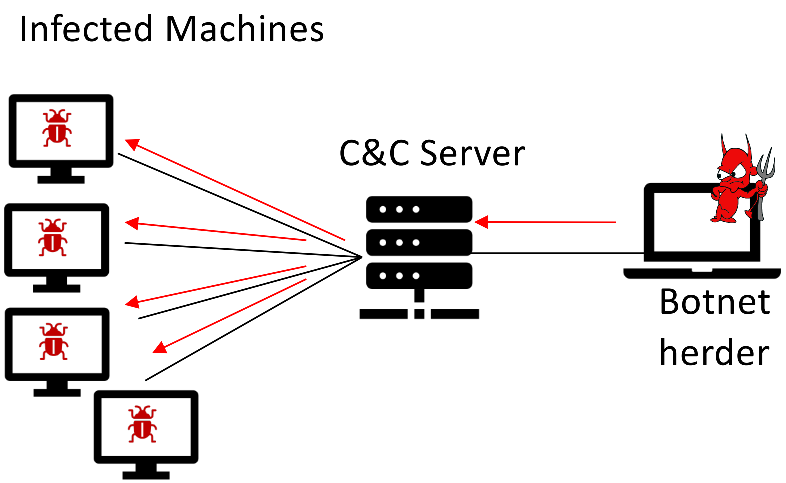
- Malicious Attachments
- Installing malicious Applications
- Software Vulnerabilities
- Misconfigured access controls
- Social engineering

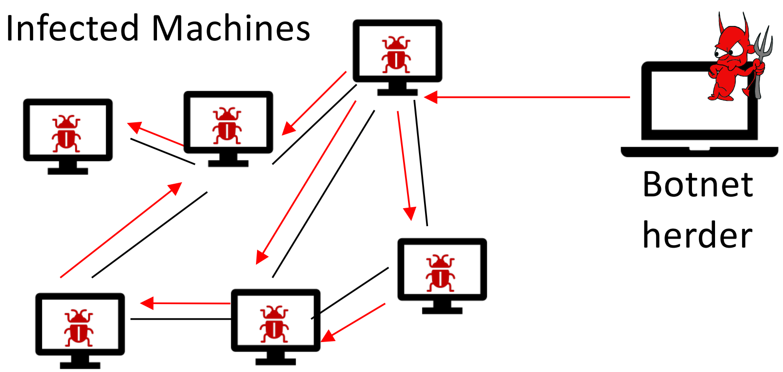
Typical Payloads

- DDoS Engines
- SPAM Engines
- Phishing Engines
- Information Stealing
- Miners

Command and Control Techniques

- **Centralized**
  - Attacker operates central infrastructure to distribute commands to the victim machines
  - Two main techniques used
    - IRC Servers: commands are pushed to connected clients
    - HTTP Servers: commands are pulled by victim clients
- **Decentralized**
  - The victim machines form a P2P network
  - Commands of an attacker are distributed from P2P directly
- **Many of today's bots are hybrid**

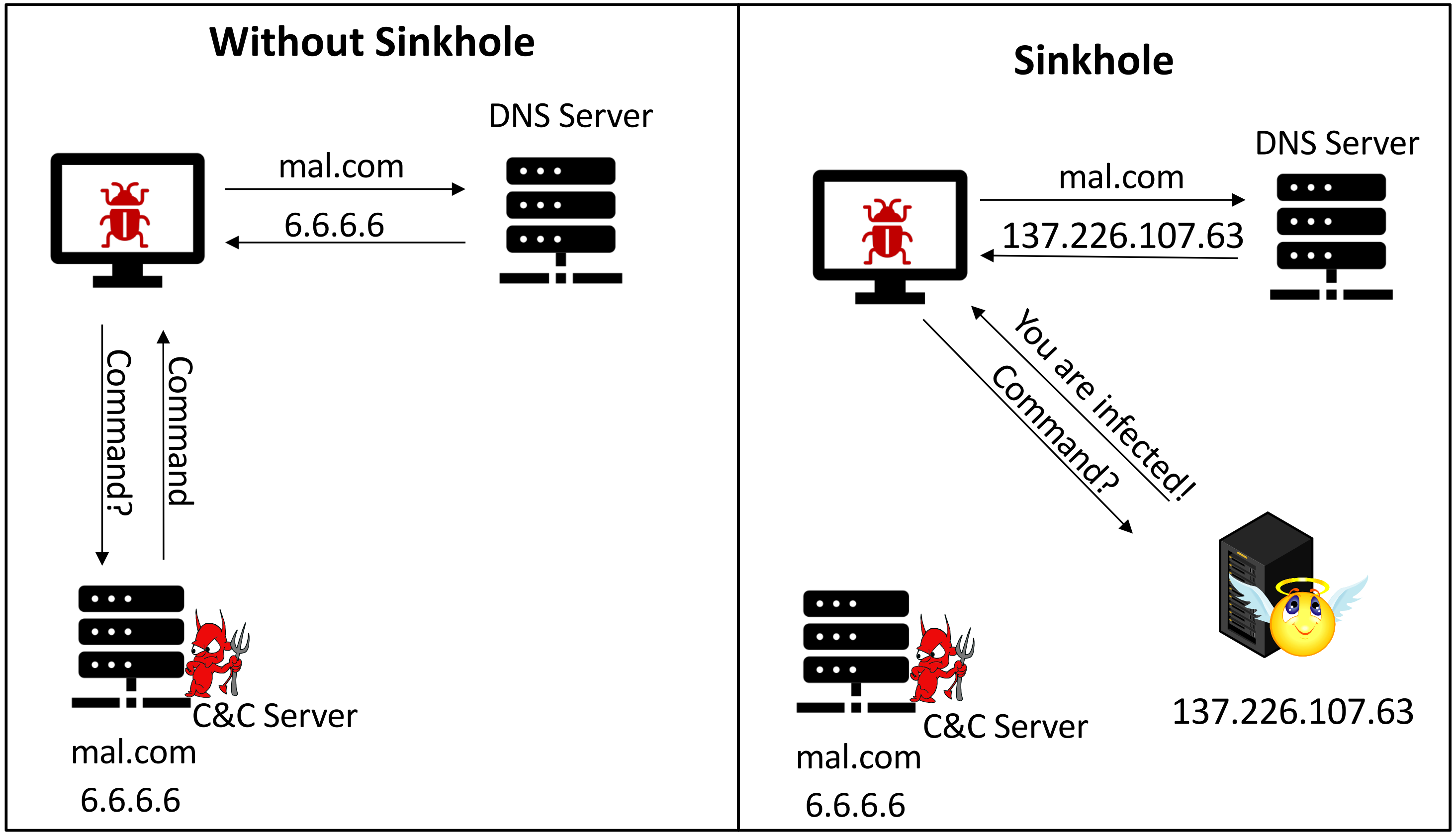




Taking Down a Centralized C&C Infrastructure

- **Locate C&C servers and take them down**
  - Analyze network traffic of infected machines
  - Analyze bot malware itself by reverse engineering the code
  - If it is C&C server is a compromised machine, contact legitimate owner
- **Make C&C server impossible to contact**
  - Block domain name in DNS
  - Block IP range of C&C infrastructure
  - Disconnect rogue hosting companies
- **Find out which devices in your network are infected by**
  - **Sinkholing** the corresponding domain names and see who connects
  - Automatically warn users of infected machines

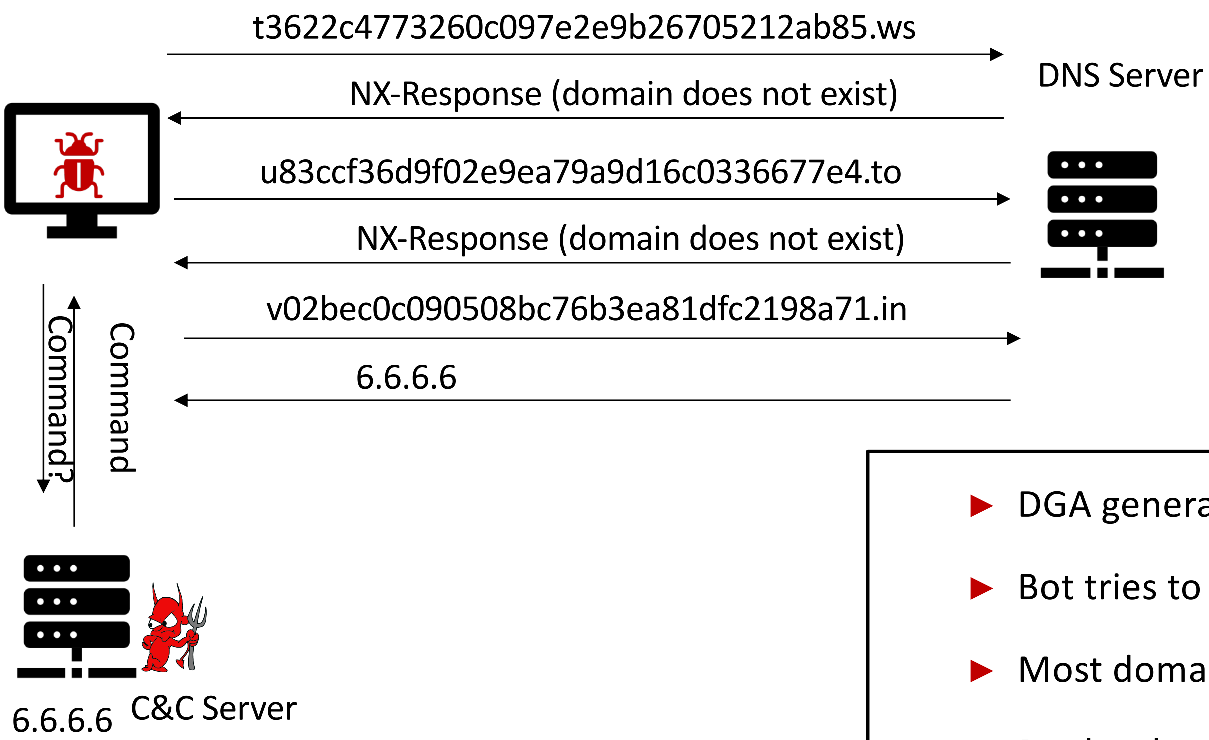
DNS Sinkholing of known Malicious Domains



Hiding the IPs of C&C Servers to Impede Take Down

- **Use of Domain Generating Algorithms (DGAs)**
  - Change domain name of machine queried for commands e.g. by an HTTP-bot based on a DGA using a seed (e.g. time stamp, twitter post,…) as input
  - Domain names queried change frequently
  - Attack has to register the queried domain names in order to be able to distribute commands
  - If DGA and seed are known domain names can be blocked in local DNS
- **Use of Fast Flux in DNS**
  - Multiple IP addresses associated with a single domain name, no one server to take down
  - IP addresses quickly changed by changing DNS records
  - IP addresses typically belong to compromised servers
  - Still domain name can be blocked locally at DNS server on the victim's network

Hiding C&C Server by DGA



- DGA generates domains
- Bot tries to resolve domains
- Most domains are not registered
- Bot herder registers one or more domains per day
- Bot connects to C&C server and asks for commands

Malware Terminology

Name	Description
Advanced Persistent Threat (APT)	Sophisticated malware directed at specific business or political targets applied persistently and effectively
Adware	Advertising integrated in software, often results in pop-up ads or redirection of a browser to a commercial site
Attack kit	Set of tools for generating malware, including propagation and payload mechanisms
Auto-rooter	Malicious hacking tool used to remotely break into machines
Backdoor	Any mechanism that bypasses a security check, allows unauthorized access to functionality in a program or system
Downloader	Code that installs other items on a machine, e.g. loads a larger malware packed after initial infection
Drive-by-downloads	Uses code in a compromised web site that exploits a vulnerability in the browser or browser plugins
Exploit	Code specific to exploiting a single vulnerability or set of vulnerabilities
Flooder (DoS engine)	Generates large volume of data, e.g. to carry out denial of service attack

Malware Terminology

Name	Description
Key logger	Captures keystrokes on the infected system
Logic bomb	Code inside a malware, triggers when a specific condition is met
Macro virus	Uses macro or scripting code, typically embedded in document
Mobile code	Code that is portable between different platforms
Rootkit	Set of hacker tools used to hide the malware and gain root access
Spam engines	Used to send large volumes of unwanted email
Spyware	Collects information from a computer and transmits it to another system (e.g. key strokes, screen shots, network traffic…)
Trojan horse	Appears to be useful but also has a secondary malicious purpose
Virus	Tries to replicate itself into executable of script code when executed
Worm	Runs independently and propagates copies of itself, typically uses software vulnerability
Bot (Zombie)	Activated on an infected machine to gain remote control to launch attacks on other machines

Overview

Malware Types by Spreading

- Viruses
- Worms
- Trojans

Botnets

- C&C Infrastructures

Initial Infection

- Malicious Attachments
- Installing malicious Applications
- Software Vulnerabilities
- Misconfigured access controls
- Social engineering

Typical Payloads

- DDoS Engines
- SPAM Engines
- Phishing Engines
- Information Stealing
- Miners

Malicious Attachments

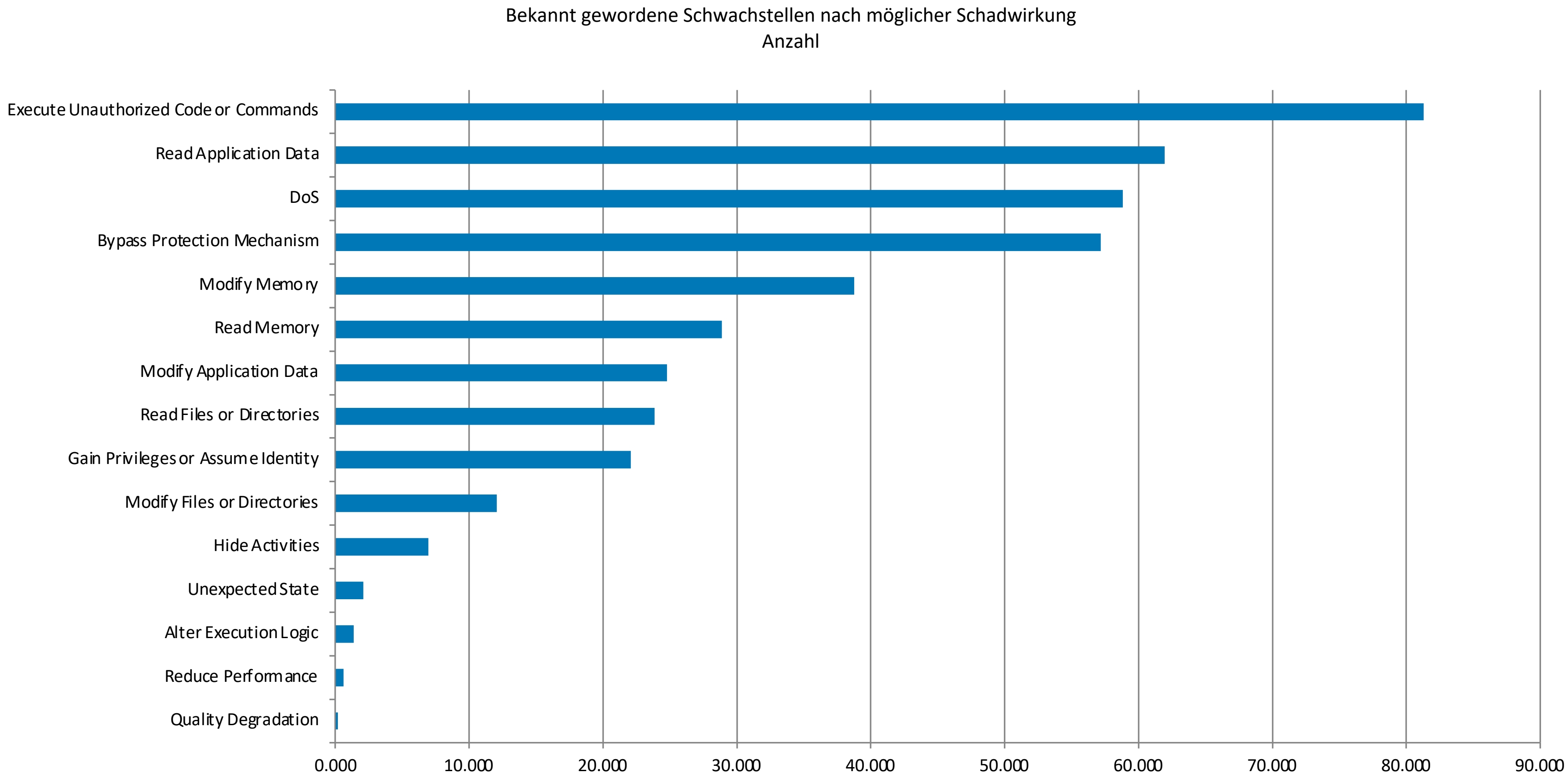
- **Spread mostly over Emails but also over Instant Messengers and SMS**
- **May contain executable code or files with macro viruses**
- **Often used in connection with social engineering, e.g.,**
  - Email pretending to be from some reputable business
    - Pretending to contain an order confirmation, tax information, bill,…
  - Email pretending to answer to job advertisements or call for bids,…
    - Pretending to contain application papers, offers,…
  - Emails pretending to alert users of security breaches etc.
    - Pretending to contain cleaning software that urgently needs to be run,…

- **E.g., according to BSI-Lagebild 2022:**
  - 34 000 emails **per month** filtered in German government networks

Installing Malicious Applications

- **Trojans are typically deliberately installed by users**
- **User tricked into installing them by claimed functionality**
  - Free versions of games
  - Free anti-virus products
  - …
- **Most common strategy used to infect mobile devices still**

Software Vulnerabilities



Example for Execution of Unauthorized Code: Buffer Overflow – Definition by NIST

Buffer Overflow: A condition at an interface under which more input can be placed into a buffer than the capacity allocated for it, overwriting other information. Attackers exploit such a condition to crash a system or to insert specially crafted code that allows them to gain control of the system

Example for a Basic Buffer Overflow in C Code

```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
        valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

Copies some expected tag value into str1

`gets()` does not do any length checking!

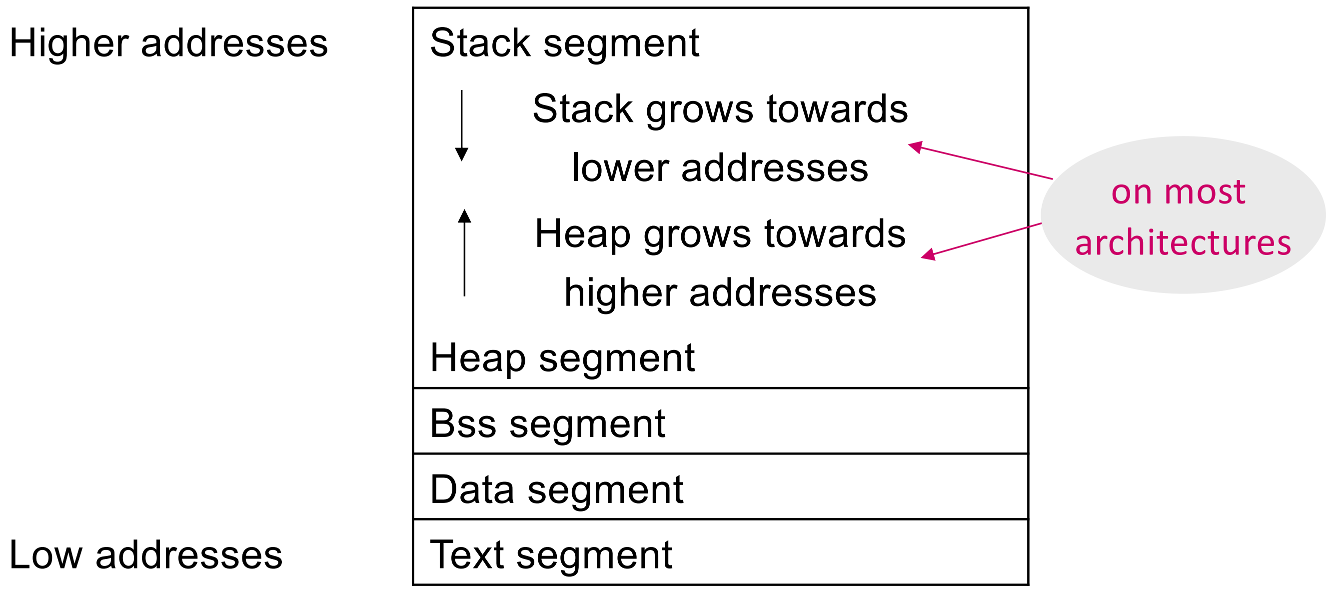
Assume tag is START

```
$ cc -g -o buffer1 buffer1.c
$ ./buffer1
START
buffer1: str1(START), str2(START), valid(1)
$ ./buffer1
EVILINPUTVALUE
buffer1: str1(TVALUE), str2(EVILINPUTVALUE), valid(0)
$ ./buffer1
BADINPUTBADINPUT
buffer1: str1(BADINPUT), str2(BADINPUTBADINPUT), valid(1)
```

Basic Buffer Overflows

- **The simple example on the last slide results in a variable corruption**
  - Overly long input data overwrites memory location of another variable
  - This may already result in a serious attack
    - E.g., if `next_tag` contained a password to which the input (`str2`) is to be compared before access to some system resources are granted
- **More sophisticated buffer overflows target corruption of program control addresses in order to alter the flow of execution of the program**
- **To exploit any type of buffer overflow vulnerability an attacker needs to**
  - Identify a buffer overflow vulnerability in some program that can be triggered using externally sourced data under the attacker's control
    - E.g., by inspecting the source code of a program or using fuzzing tools
  - Understand how that buffer will be stored in the processes memory and can thus be used to corrupt adjacent memory locations (architecture and compiler dependent)
    - Recap on memory segmentation

Executable Program's Memory Segments



- **Compiled program's memory is divided into five segments**
  - text, data, bss, heap, stack
  - Text, data and bss segments are of static size,
  - Heap and stack shrink and grow dynamically during program execution

Stack Buffers

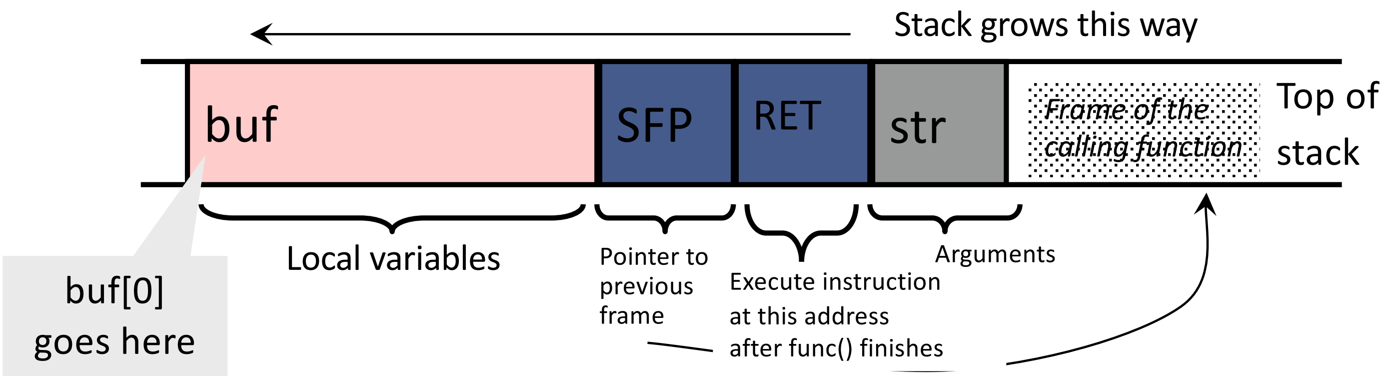
- **Suppose Web server contains this function**

```
void func(char *str) {
     char buf[126];
     strcpy(buf,str);
}
```

Allocate local buffer (126 bytes reserved on stack)

Copy argument into local buffer

- **When this function is invoked, a new frame with local variables is pushed onto the stack**



What If Buffer is Overstuffed?

- Memory pointed to by str is copied onto stack…

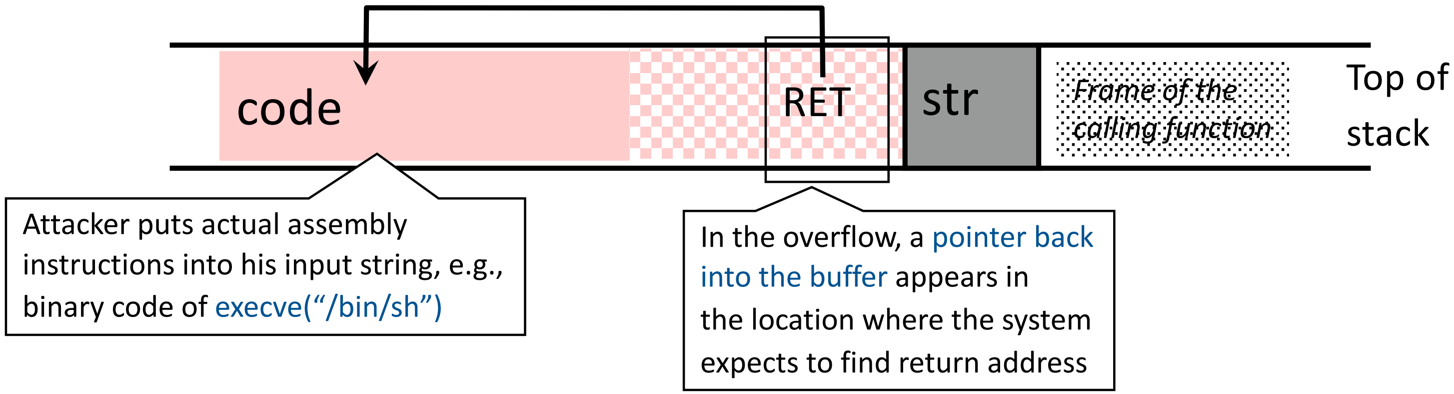
```
void func(char *str) {
     char buf[126];
     strcpy(buf,str);
}
```

strcpy does NOT check whether the string at *str contains fewer than 126 characters

- If a string longer than 126 bytes is copied into buffer, it will overwrite adjacent stack locations

Executing Attack Code

- **Suppose buffer contains attacker-created string**
  - For example, *str contains a string received from the network as input to some network service daemon



- **When function exits, code in the buffer will be executed, giving attacker, e.g., a shell**
  - Root shell if the victim program is setuid root

Cause: No Range Checking

- **strcpy does <u>not</u> check input size**
  - strcpy(buf, str) simply copies memory contents into buf starting from *str until “\0” is encountered
  - ignores the size of area allocated to buf
- **Many C library functions are unsafe**
  - strcpy(char *dest, const char *src)
  - strcat(char *dest, const char *src)
  - gets(char *s)
  - scanf(const char *format, …)
  - printf(const char *format, …)
  - …

Does Range Checking Help?

- **strncpy(char *dest, const char *src, size_t n)**
  - If strncpy is used instead of strcpy, no more than n characters will be copied from *src to *dest
    - Programmer has to supply the right value of n
- **strncat(char *dest, const char *src, size_t n)**
  - If strncat is used, then the first n characters from *src will be appended to *dest

- **Potential overflow in htpasswd.c (Apache 1.3):**

```
… strcpy(record,user);
        strcat(record,":");
  strcat(record,cpw); …
```

Copies username ("user") into buffer ("record"), then appends ":" and hashed password ("cpw")

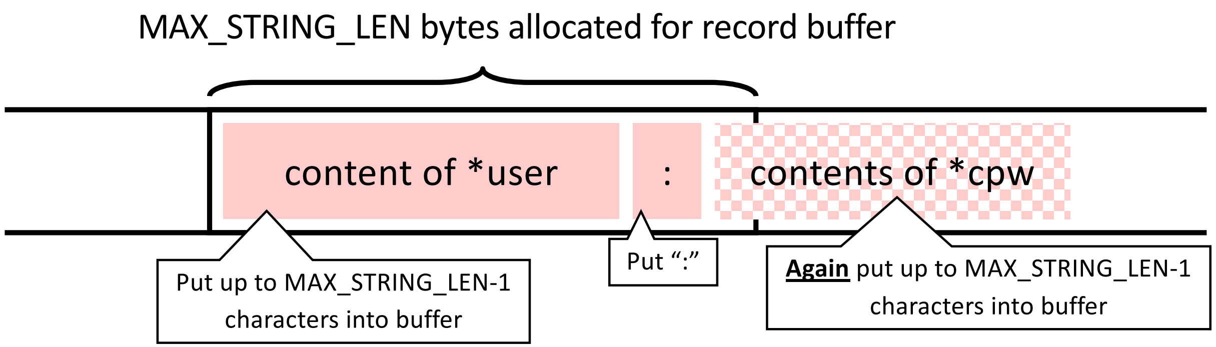
- **Published "fix" (do you see the problem?):**

```
… strncpy(record,user,MAX_STRING_LEN-1);
  strcat(record,":");
  strncat(record,cpw,MAX_STRING_LEN-1); …
```

Misuse of strncpy in htpasswd “Fix”

- **Published “fix” for Apache htpasswd overflow:**

```
… strncpy(record,user,MAX_STRING_LEN-1);
   strcat(record,":");
   strncat(record,cpw,MAX_STRING_LEN-1); …
```



Defense against Buffer Overflows

- **Defense Mechanisms can broadly be divided into**
  - Compile time defenses, which aim to harden new programs to resist attacks
  - Run-time defenses, which aim to detect and abort attacks in existing programs
- **Compile-time defenses**
  - Choose a high-level programing language that does not permit buffer overflows
    - Programs may still be vulnerable if existing system libraries are used
    - Disadvantage: direct access to some instructions and hardware resources lost
  - Encourage safe coding standards
  - Language extensions and use of safe standard libraries such as libsafe
  - Include additional code at compile time to detect corruption of the stack frame at runtime
    - E.g. gcc extensions such as Stackguard, Stackshield, and Return Address Defender

Run-Time Defenses – Executable Address Space Protection

- **Typical memory exploit involves code injection**
  - Put malicious code at a predictable location in memory, usually masquerading as data
  - Trick vulnerable program into passing control to it
    - Overwrite saved EIP, function callback pointer, etc.
- **Idea: Make stack and other data areas non-executable**
  - Needs to be supported by the processor‘s memory management unit
    - Tag pages of virtual memory as non-executable
  - Some useful functionality also uses executable code on the stack, e.g., nested functions in C, Linux signal handlers,…
- **Support has become standard in most modern operating systems**
  - Protects against classic overflows, where shellcode is included in stack buffer
- **Consequence:**
  - Newer buffer overflow exploits use more sophisticated techniques such as using code already existing on the target machine,…

Misconfigured Access Controls

- **Examples for misconfigurations include**
  - Weak user-selected passwords
  - Weak default passwords that are not changed
  - Open port such as open ssh port
  - …

Social Engineering

- **Essential part of many already mentioned infection paths**
  - Malicious attachments
  - Installing malicious applications
  - …
- **Other examples**
  - Trick users into revealing their password
  - Trick administrators into resetting passwords of specific users
  - Trick users on the phone / via email
  - Trick users into entering account credentials into fake websites
    - Phishing
  - …

Overview

Malware Types by Spreading

- Viruses
- Worms
- Trojans

Botnets

- C&C Infrastructures

Initial Infection

- Malicious Attachments
- Installing malicious Applications
- **Software Vulnerabilities**
- Misconfigured access controls
- Social engineering

Typical Payloads

- DDoS Engines
- SPAM Engines
- Phishing Engines
- Information Stealing
- Miners

Examples for Malicious Purposes aka Payload

Ransome Ware

- Encrypt all or some files on the victim machine
- Ask for ransom to release encryption key
- Makes use of crypto currencies for payment

DDoS Engine

- Enables infected machine to participate in DDoS attacks w/o user's consent

Data Theft and Espionage

- Steal sensitive information from infected machine

SPAM or Phishing Engine

- Engines that allow to sent spam or phishing emails from the victim machine

Key Logger or general Spyware

- Logs a user's keystrokes and stores them
- Sends them off to the attacker
- Thereby steals, e.g., account credentials, credit card information…
- Turn on camera remotely to spy

Examples for Payloads and Additional Malicious Functionalities

Crypto Miners

- install a malicious program on the victim's host that helps in mining crypto currencies
- Runs in the background and typically uses computing resources while victim machine is idle
- Spreads the energy consumption and computing time over multiple victim machines

Bot

- enables attacker to remotely control an infected machine via a command-and-control infrastructure

Rootkit

- allows to maintain covert root access to the infected machine
- hides any evidence of its presence, e.g., by installing malicious versions of standard system programs such as netstat, ps, ls, du, et.

References

- **W. Stallings, Cryptography and Network Security: Principles and Practice, 8th edition, Pearson 2022**
  - Chapter 21: Network Endpoint Security
    - 21.3 Malicious Software
- **Wenliang Du, Computer Security a Hands-on Approach, 3rd edition, 2022**
  - Chapter 4: Buffer Overflows