Lecture 10: *nformato* Surface And Access Control

Security Problems: All vulnerabilities share commonalities.

- Caused by software flaw (programming errors)
- Can be anywhere in a program
- Exploits are carried out through some interface accessible to an adversary (e.g. command line input)
- Typically we look for software flaws. In most cases they are benign and not security problems

Attack Surface: A bunch of entry points that are accessible to an adversary. That is, they are untrusted inputs that can come into a program

Example 1: A remote attacker trying to compromise a target system with an http server

- Goal is to execute code remotely

Example 2: A local attacker (no privilege like project 3) trying to compromise a web server running on the same system.

- An attacker can find software vulnerability in the http server. Establish a connection with the http server and interact with the web server through that entry point to compromise the server.
- An attacker may also find a software flaw within the OS kernel and compromise it through the entry point of system calls.
- An attacker can compromise services that the http server depends on. Those services like daemon and file server can have entry points accessible to attackers.
- Goal: change the behavior of the HTTP server

Threat Model: Describes the capability of an attacker. How powerful is the adversary. Is it an adversary that already has an account on the system? Or is the adversary remote and has no access to the machines in the internal network?

The attack surface is relative to the threat model.

Relative Attack Surface Metric (system)

- A relative attack surface quotient (RASQ) metric is used to compare systems which has larger relative attack surface
- The metric lists a set of entry points that you should be concerned about minimizing as a system distributor
- Example: open (TCP/UDP) sckets, open RPC endpoints, open named points, services, services running by default, … etc
- Can count number of unsafe instances

Individual Program Attack Surface

- Identify relevant subset of system resources that can be used in an attack (are or could be controlled by an adversary)
- Identify when such resources may be used by the program (program entry points)

Program Entry Points

- Programs obtain information from external sources (e.g. files and network sockets), and program statements that access such external sources are entry points

Example: System calls provide sources for gaining most external information. But for attack surfaces, we focus on the statements that a program makes to the individual library/system calls

System Calls As An Attack Surface

- Why use system calls for the attack surface?
- Program has to use library calls to access external resources
- Wrappers in libraries e.g. fopen() vs. open()
- Program statements that invoke each call (only a subset of these may be adversary accessible)
- ALL system calls constitute a program's attack surface. At some point, a system call may access adversary-controlled data. So test them all.

Example: Consider an apache web server. A user can create a webpage. Program attack surface (would be accessible to an adversary) would be the html page and the http. The attacker can create a complex html page and exploit a bug with the html parser. The attacker can also supply malicious network requests.

Example: E-voting. Which commands may be threatened (attack surface)?

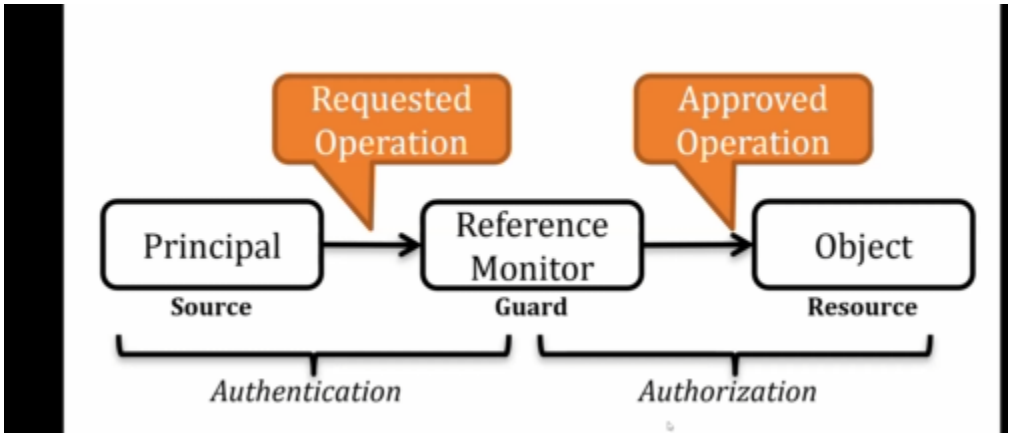
- Start program (by admins): You can use command line arguments to the program and use a buffer overflow attack
- Submit a vote (by voter): Voter has incentive to cheat. Amplify vote or change existing votes. Entry point could be a touch screen. But other possible entry points like bluetooth. Some devices may have bluetooth interfaces left on by default. Voter could issue network requests over bluetooth interface and try to find a vulnerability and compromise it
- Count votes (by tallier): Interface to allow tallier to manipulate the votes

Lecture 11: Access Control, BLP and Biba Policies

Access Control

- Give users permissions access to certain resources
- E.g. linux file system access control. Access control bits when you do ls -l
- Attack surface computed with respect to threat model and access control policy
- Trying to minimize attack surface with attack control

Reference Monitor: A guard that mediates access to resources. Any requested operation goes through the reference monitor.



Policies and Mechanisms

- Policy says what is, and is not allowed
- Mechanisms enforce policies
- Compositions of policies: if policies conflict, discrepancies may create security vulnerabilities

Example:

- Policy says file should be readable by only the root user
- Mechanism: Kernel checks through the file open() syscall

State of a system

- A collection of the current values of all memory locations, storages, registers, etc.
- A subset of this collection that deals with protection is the protection state of the system

Protection States

- An abstraction that focuses on security properties
- Primarily interested in characterizing safe states
- Goal is to prove that all operations in the system preserve security of the protection state

- **Access Control Matrix** is our first protection state model

Access Control Matrix Model
- Access Control Matrix A
- Subjects S
- Objects O
- Rights R
- State: (S, O, A)

Safety
- Let R be set of generic rights of system
- When generic right r is added to an element of an access control matrix not already containing r, that r is said to be leaked.
- If a system can never leak right r, the system is called safe with respect to the right r

Other Types of Security Policies:
- Confidentiality policy and integrity policy

Confidentiality Policy
- Goal: prevent the unauthorized disclosure of information
- Deals with information flow
- Extensive redundancy in military makes integrity/availability less of a problem
- Example: BLP Model (Bell-Lapadula Model)
- Security levels arranged in linear ordering
- Information flows up, not down
- Write up allowed, write down disallowed
- Top secret, secret, confidential, unclassified

Lattice Model
- Expand notion of security level to include categories
- Security level is (clearance, category set), or formally (L, C) where L is the clearance level and C is the set of categories

Example:
- Alice (TOP SECRET, {CRYPTO})
- Bob (SECRET, {NUC})

Integrity Policy
- MLS talks about who can “Read” a document
- Integrity is considered who can write or make change to a document
- That is, who can affect the integrity (content) of a document
- Example: You may not care who can read DNS records, but you better care who changes them

BIBA Model

- Dual secrecy for integrity
- No read down, no write up
- High integrity, some integrity, suspicious, garbage

In Reality:

- High integrity programs may need to access low integrity data
- Command line input, web server accepting data from internet, attacker controlled files: ssh daemon runs as root and reads /home/[user] directory
- Sometimes programs may expect high-integrity objects mistakenly

Example: programs require a variety of resources to function

- Files
- Inter-process communication channels
- Signals
- Inputs to OS: name, bindings (dirs, symbolic links)
- Output: resource
-

Example: confused deputy attack

Checking Low Integrity Input

- Doesn't work cause of race condition
- Adversary exploits non-atomicity in "check" and "use" of resource
- Time-of-check, time-of-use (TOCTTOU) attack

Directory Traversal:

- Trick apache server to give /etc/passwd file by maliciously crafting the http get request
- Root cause: program did not perform adequate checks

Fundamental Problem:

- When a program expects a high integrity resource but gets a low integrity one
- When a program expects low integrity resource but did not perform adequate checks
- They are all part of the attack surface

Lecture 12 - vulnerability discovery - fuzzing

Vulnerability
- Flaw: Can we find flaws in the source code?
- Accessible to adversary: Can we find what is accessible (attack surface)?
- Adversary has ability to exploit: Can we find how to exploit (highly dependent on types of flaws e.g. buffer overflow) ?

One Approach
- Run program on various low-integrity inputs (through identified attack surface)

Dynamic Analysis Options
- Regression Testing: Run program on many **normal** inputs and look for bad behavior in the responses. Typically looking for behavior that differs from expected
- Fuzz Testing: Run program on many **abnormal** inputs and look for bad behavior in the responses. Looking for behaviors that may be triggered by adversaries.
- Fuzz testing is more likely to find vulnerabilities because it's testing for uncommon inputs whereas regression testing tests common inputs. Fuzz testing finds corner cases.

Fuzz Testing
- Generate random inputs
- Run lots of programs using random inputs
- Identify crashes of these programs
- Correlate with random inputs that caused the crashes
- PROBLEMS: not checking returns, Array indices

Challenges
- Idea: Search for possibly accessible exploitable flaws in a program by running the program under a variety of inputs
- Challenge: selecting input values for the program
- When we choose input values for dynamic analysis, we want to find all exploitable flaws with fewest possible inputs

Black Box Fuzzing
- Feed the program random inputs and see if it crashes
- Pros: easy to configure
- Cons: May not search efficiently
- May re-run over the asme ptah over again (low coverage)
- May be very hard to generate inputs for certain paths (checksums, hashes, restrictive conditions)
- May cause program to terminate for logical reasons

Mutation-Based Fuzzing

- Supply a well-formed input: Generate random changes to that input
- No assumptions about input. Only assumes that variants of well-formed input may be problematic
- Easy to setup and not dependant on program details
- May be strongly biased by the initial input
- Still prone to problems: may re-run same path over again, may be very hard to generate inputs for certain paths (checksums, hashes, restrictive conditions)

Generation-Based Fuzzing

- Generate inputs “from scratch” rather than using an initial input and mutating
- To overcome problems of naive fuzzers, they often need a format or a ptotocal spec to start
- Cumbersome bc you need a fuzzer spec for every input you are fuzzing
- Pros: more complete search, more accurate. Values more specific to the program operation. Can account for dependencies between inputs
- Cons: More work. Need to get the specification, write the generator which is done ad hoc, need to do for each program

Grey Box Fuzzing

- Also called coverage-guided
- Instrument program to track the paths run
- Save inputs that lead to new paths
- “State of practice” at this time
- Pros: finds flaws, but still does not understand program, much better than black box testing, no configuration, lots of crashes identified
- Cons: may not be able to execute some paths, searches for inputs independently from the program

White Box Fuzzing

- Combines test generation with fuzzing
- Rather than generating new inputs and hoping they enable a new path to be executed, compute inputs that will execute a desired path
- Goal: given program with a set of input parameters, generate a set of inputs that maximizes code coverage

Lecture 13 - vulnerability discovery - static analysis

Static Analysis

- Goal: explore all possible executions of a program. All possible inputs and all possible states
- Provides an approximation of behavior
- Run in the aggregate: rather than executing on ordinary states, finite sized descriptors representing a collection of states
- Covers all paths but may produce false alarms

Dynamic vs Static

- Dynamic:
    - Select concrete inputs
    - Obtain a sequence of states given those inputs
    - Apply many concrete inputs (i.e., run many tests)
- Static:
    - Select abstract inputs with common properties
    - Obtain abstract/approximate states created by executing abstract inputs
    - One run

Control Flow Analysis

- Compute control flow to find execution path that does not check the return value of a function
- Program statements of interest: basic blocks, conditionals, loops, calls, return

Data/Information Flow Tracking

- Can help with secrecy and integrity
- Secrecy: write programs in which all secret data is only output to authorized subjects
- Integrity: Write programs in which there is no way to access adversary-controlled data that may be used to modify unauthorized data
- Both achieved by tracking program information flows

Lecture 15: Network Security 1

A machine sends a network packet to another on the internet. Packet needs source and destination ip address. Each ip address needs a port number specified. Port numbers differentiate different services on a host. This is layer 2 and layer 3. Transport and network.

Application Layer

- Examples: HTTP, FTP have a specific packet format. The payload contains the specification for describing the data.

Transport Layer

- We use some socket API to send the application layer
- Attach TCP header. TCP header contains
- HTTP uses TCP
- Contains source and destination port number

Internet Layer

- IP header contains source and destination IP address

Link Layer

- Contains source mac address and destination mac address

Common Network Security Attacks and their countermeasures

- Packet sniffing and spoofing. Use encryption
- Finding a way into the network. Use firewalls
- Exploiting software bugs, buffer overflows. Use intrusion detection systems
- Denial of service. Ingress filtering, IDS

Common Threat Models In Network

Passive Eavesdropper: read and insert

- Suppose A and B are talking to each other
- An attacker that's eavesdropping can read what A and B are sending to each other.
- An attacker can also send/inject a packet to A or B

Man-in-the-middle

- On the communication path. Example: compromised router, government surveillance, ISP
- An attacker can read and write packets and also drop

Off-path attacker:

- Attacker NOT on communication path and cant eavesdrop
- Can only insert packets to A or B

Most Powerful to least powerful:

- Man in the middle
- Passive Eavesdropper
- Off path attacker

Passive Eavesdropper

- Coffee shop free wifi
- Can impersonate somebody/something by inserting packets

Man-in-the-middle

- ISPs, routers
- ISPs, three letter agencies control routers and can launch attack
- Can drop a packet from A to B and insert a different packet to B to change the message

Off-path

- Can spoof a message
- Pretend a message is coming from A when sending to B

IP Layer

- Responsible for end to end transmission
- Sends data in individual packets
- Maximum size of packet is determined by the networks
- UNRELIABLE: packets might be lost, corrupted, duplicated, delivered out of order

IP Address

- A field in the IP layer, most important field
- IPv4 is 4 bytes
- Each device normally gets one

Routing

- All devices need to know what IP addresses are on directly attached networks
- If destination is on a local network, send it directly there
- If destination address isn't local, most non-router devices just send everything to a single local router (gateway)
- Routers need to know which network corresponds to each possible IP address

IP Packets

- Source and destination address
- Protocol number: 1=ICMP, 6=TCP, 17=UDP
- Time to live: prevent routing loops. How many hops can packet live

Problem with IP address (off-path attack)

- Source address in a packet can be filled arbitrarily by a host (think of USPS mail)
- Lack of authentication of packet sources
- Many vulnerabilities arise because of this

UDP

- Thin layer on top of IP (alternative to TCP)
- Adds packet length + checksum (guard against corrupted packets)
- Also source and destination ports (ports are used to associate a packet with a specific application at each end)
- Still unreliable: duplication, loss, out-of-orderliness possible
- Used for DNS, VoIP, some games
- Better for when packet loss is better handled by the application than the network stack
- Better when the overhead of setting up a tCP connection isn't wanted

Off-path attack against UDP

- Need to guess the client's port number for UDP because when A and B are talking, there is some dedicated port number associated with the communication session
- Need to guess destination port number
- Can be guessed with brute force

DNS Protocol: Application Layer Protocol

- A protocol that translates human readable domain names into computer recognizable ip addresses
- 1) Type google.com in browser. Your browser issues a DNS request to a local DNS resolver
- 2) DNS resolver ask google's name server for the ip address
- 3) Google's name server responds with ip address and gives it to DNS resolver
- 4) DNS resolver forwards the ip address to client/browser
- DNS stores a cache, a copy of the mapping from google.com to its ip address. In future, when somebody asks for google.com, request will hit the cache and you wont have to go to name server again

DNS Cache Poisoning Attack

- Attacker pretends to be the name server and gives a false ip address for google.com
- DNS Resolver saves a mapping with false ip address
- All three attack types can launch this attack: off-path, man in the middle, passive eavesdropper
- Challenge is to guess the port number for the DNS resolver. It is random. Port number for Google NS is known
- Also have to consider TxID thats in the application layer. It is also 16 bit.

- Attacker has to correctly guess port number and TxID at same time, a 32 bit secret = 4 billion

TCP

- Reliable, connection-oriented, stream delivery
- Interface presented to the application doesn't require data in individual packets
- Data is guaranteed to arrive, and in correct order with no duplicates (or connection will be dropped)
- Imposes significant overheads

Applications of TCP

- HTTP, FTP, SMTP

TCP implementation

- Connections are established using a three-way handshake to make sure client and server are in sync with each other
- Data is divided up into packets by the OS
- Packets are numbered, and received packets are acknowledged
- Connections are explicitly closed (or may abnormally terminate)
- TCP Packets: Source & dest port, sequence number, acknowledgement number, checksum, various options

Off-path attack against TCP

- Need to guess port number, sequence number, and acknowledgement number
- Sequence number and acknowledgement are 32 bits each
- Destination port = 16 bit
- If you are using eavesdropping or MITM, you dont need to guess portnumber, sequence number, ack number because they are sent in plaintext

Defenses - encryption

- Without the secret key, an attacker can't read, write, or inject data
- But traffic almost never encrypted: IP, TCP/UDP, DNS, telnet

Lecture 15: Network Security 2

Firewall

- If you don't have one, then somebody on the internet can discover you
- Dangerous because attacker can scan your network remotely, find machines running, how many machines running, what OS running,
- In fact in a local network, you can see MAC addresses
- Problem is that many network applications and protocols that have security problems that are fixed over time. It's difficult for users to keep up with changes and keep host secure
- Solution: Administrators limit access to end hosts by using a firewall. Firewall is kept up-to-date by administrators
- A firewall is like a castle with a drawbridge. Only one point of access into the network
- A firewall can be hardware or software
- Firewalls look at features of a packet and decide whether they should be allowed. This has to be configured on your router.These are called packet filtering firewalls
- E.g. drop packets with destination port of 23 (Telnet)
- Why we don't just turn off Telnet? Because firewall blocks them anyway. It's more convenient to just block with one firewall sitting at the gateway rather than have to turn them off for 10,000 computers.Save network bandwidth by dropping packet at firewall before sending them to devices.

Intrusion Detection

- Used to monitor for suspicious activity on a network
- Monitor payload of packets: modern firewalls also do that (blurred definition)
- Uses intrusion signatures. Well known patterns of behavior: ping sweeps, port scanning, web server indexing, OS fingerprinting
- IDS only useful against certain forms of attacks. Can't monitor encrypted traffic and attacker can morph into different player
- Even if encrypted, you can look at packet size, timing of packets, sequence of packets

Denial Of Service

- Purpose is to make a network service unusable, usually by overloading the server or network
- Strategies:
- Attacker: small resources -> amplifying impact
- Attacker: large resource -> bruteforce

SYN flooding attack

- Send SYN packets with spoofed/bogus source address
- Server responds with SYN ACK and keeps state about TCP half-open connection

- Eventually server memory is exhausted with this state
- Can cause CPU to be a bottleneck

Botnet: many machines controlled by one bot master

Side Channel Attack Against TCP

- Threat Model: a blind off-path attacker
- Secrets: presence of a connection, sequence number in both directions (tells you how many bytes exchanged, how long you stayed on website, attacker can inject fake payload)
- An attacker can figure out if two ip addresses are communicating with each other with a TCP connection. Can see what websites a person is visiting
- An attacker can forcefully shutdown TCP connection, by injecting a spoof reset packet. Pretend that packet came from server
- The remote attacker can probe client or server and count the number of responses, and the timing or payload or header of the responses

Shared Resource for side channel attack against TCP

- Challenge ACK rate limit shared across all connections
- By default 100 per second
- i.e., in a one second window, a server is able to send 100 challenge acks across all connections to any clients
- Attacker has an ongoing TCP connection with the server. Attackers can solicit 100 challenge ACKS to itself.

Exploit The Vulnerability:

- Goal of attacker is to infer whether a client and server have an ongoing TCP connection
- 1) client sends spoofed SYN packets with client;s IP and a guessed src port. Trying to see if there is a collision with the correct src port number.
- 2) If the ip address and src port number match an ongoing connection, server sends 1 challenge ack to client. Are you sure you want to establish a new connection an abandon the current one to the client? But if 1 challenge ack is sent to client how does attacker know? Attacker can indirectly infer this information by trying to solicit 100 challenge acks to itself using nonspoofed packets. If server responds with 99 challenge acks, then attacker knows there is a connection TCP between client and server.
- Can do the same exact thing to get sequence number

Side Channels in UDP

- Security of DNS relies on randomized source port of DNS requests
- Two secrets: src port and txID, each 16 bit long
- If attacker wants to spoof a DNS response, then they have to guess both secrets correctly. Impossible to guess 2^32 in short period
- Attacker can infer the presence of a UDP session by first inferring the source port number, and the brute force the transaction ID number

Final Review

Overview

- Simplicity: less to go wrong, fewer possible inconsistencies, easy to understand
- Restrictions: minimize capability, minimize access, inhibit communication

Defense In Depth

- More restrictions better
- Add more restrictions ⇒ more complex. Thats the tradeoff

Least Privilege

- A subject should be given only those privileges necessary to complete its task

Fail-Safe Defaults

- Default actions is to deny access (firewall)
- If action fails, system as secure as when action began

Economy Of Mechanism

- No system is completely 100% secure against all attacks
- Rather systems may only need to resist a certain level of attack. No point of buying $10k firewall to protect $1k worth of trade secrets
- KISS: keep it as sample as possible

Complete Mediation

- If there are ways around a defense, it is not a complete mediation
- Check every access, usually done once on first action. UNIX: access checked on open, not checked thereafter
- If permissions change after, may get unauthorized access

Open Design

- Security should not depend on secrecy of design or implementation
- Secrecy != security
- Complexity != Security
- Security through obscurity: bad! Opposite of open design

Separation of Privilege

- Require multiple conditions to grant privilege: SMS verification. Tradeoff is usability
- Separation of duty