6.858
Computer System Security
Lecture Notes

Kevin Yang

2/1/22 - 5/9/22

1 Lecture 1

Secure = ”works” despite adversity

eg: only TAs can access grades.txt
Pos: Test by TAs Neg: bugs → bribe a TA, guess the password, steal the laptop

1.1 Systemic plan

Goal: only Alice can access F
Threat model: assumptions
Policy: plan/config to meet goal
Mechanism: code implements policy

1.2 Hard to get right

Iterate

→ monitor attacks

→ using well-understood parts

→ post-mortems

Defense

→ ∀ attacks

Adversary

→ one attack

1.3 Imperfect but useful

attack cost > value
attack other systems

We want to look at techniques that have payoff with relatively low cost
Enable features: VPN, sandboxing

Policy problems: mainly happens in corner problems such as administration/maintenance

- change pw
- reset 2FA
- backups
- audit logs
- updates

Password reset Matt Honen
A hacker was tryign to access his Gmail. reset Gmail → reset backup → reset Apple account → needs CC4
change email for Amazon account → needs CC#
If you buy something off of Amazon, you don't need to authenticate and allows you to save the credit card number. This allows the hacker to save his own credit card number which allowed him to access Matt Honen's Amazon account and his real credit card number.

1.4 Insecure Defaults

Default password in routers
Perms in Amazon S3
Even though you are the only user, the default settings is still very important because attackers can use the default settings as a pointer of attack.

1.5 Threat Model

secret designs: secret designs are not as reliable because if a hacker figures out a secret key or password, you can simply change the key/password but if they figure out the design, the entire design has to be remade. → security by obscurity

user behavior: dependence on the user to behavior exactly right → email phishing, 2FA codes

CAPTCHA: prevents automatic spam → hard OCR but cheaper using humans

Expected Software: protective software needs to be properly installed and the software needs to be trusted → hacker made a chinese xcode mirror that added back doors into apps with safe source codes

1.6 What to do?

Explicit: clarify weakness
Simple, general
Defense in depth: hard for one threat model is exactly right, we have a bunch of threat models

1.7 Bugs

1 bug / 1000 LoC
Bugs in policy implementation: disasterous
Bugs in any component can lead to exploits

2 Lecture 2

2.1 Security Architecture

Prevent known attacks
Prevent yet - unknown attacks
Limit damage from attacks

Need to figure out the Goals? Threats?
→Trust, isolation, authentication, security channels

2.2 Case study: Google(Cloud)

Goals

- protect user data: confidentiality and integrity
- availability
- accountability

Threats

- bugs
- insiders
- supply chain
- physical attacks
- malicious apps
- denial of service

Isolate

- VMs
- Linux users/containers
- Language-level sandbox(JS, wasm)
- Kernel sanbox
- Physical

Why?

- Assurance: if there is bug within the VM or something, there are other ways to neutralie the threat
- Cost
- Performance
- Compatibility

3 Sharing: ”reference monitor”

You start with a guard.
The guard is able to communicate/access a resource. It also has a policy that tells the guard who/what can access the resource.
The principle(outside user) sends a request to the guard to access the guard.
The guard will authenticate the principle, authorize the access and then audit the entire process
There exists an audit log for us to learn what is happening if an attack went through. We should keep the audit log on another device in order to make sure it is clean.

Principles

- Person
- Service
- Computer

Resources

- Services
- Email message
- Cluster manager

3.1 Authenticate

Person

- Password
- 2FA: send challenge code to service w/ request

Computer

- Cryptographic key: K_p
- Principle sends request with signature: Sig(K_p, req)

subsectionAuthorization f(principle, resource)
We can build a table of principle vs resources
We can store permissions by file aka rows

- ACL $\rightarrow$ who has access?

- RPC perms
- storage service

We can store permissions by principles aka columns

- capability
- end-user permission ticket: allows others who gained the ticket can impersonate the end user
- mainly short lived/termed access control for delegation

3.2 Denial of Service

Challenge: distinguish real vs fake attack requests

- Overprovision: don't want to use too many resources for a fake request
- Authenticate: get out of the challenge as fast as possible