Intro

We can envision our API as pipelines delivering digital goodness to our users. Pipelines need to be strong and able to contain their contents as well as keeping outside contaminants out.

In order to work correctly the pipeline must be built correctly and well maintained. Containment failures can have serious consequences.

When attempting to build a "secure" anything remember that security is not an on/off switch

It is a spectrum with security/convenience tradeoff

Real world examples: door locks are less convenient but stop casual theft, cars have anti-theft measures. Nothing is completely secure. Goal is to raise the bar high enough so that attackers look elsewhere.

Even locks have their vulnerabilities

Security is not something that can be easily or cheaply added after the fact. You can't just sprinkle security on.

Security should be baked in/deeply embedded and should permeate the entire system. Security awareness should be a goal of all phases of design, development and operation. Build security into projects and ensure they will be able to adapt over time.

Remember you are being watched. More secure APIs can result in more stable, better documented applications. there are benefits beyond security

Transport security / cleartext - there is a reason we don't ship money around in open pickup trucks

Explanation of TLS

TLS w/ self signed certs

TLS with real certs

Free certs (startssl, letsencrypt)

Demo Milford TKD certs

TLS versions (POODLE (cypher fallback), Qualsys SSL reports)

Certificate issues (self signed certs vulnerabilities , cert validation disabling)

Better way is to use good certs and implement cert pinning)

Other certificate gotchas, certificates with private keys on client machines, Dell CA certificate w/ private key

MITM eavesdropping with disabled cert validation, MITM on open wifi

Pineapple description and threats

Demo Fiddler

Demo ShipBattle services

Authentication, how do you tell someone/something is what they say they are?

- no authentication

- basic auth with static credentials

- basic auth over HTTP (attack header)

- Oauth workflow

Authorization

- ensure authentication checks are performed

- don't trust user input (auth token + username manipulation)

Session persistence - where/how to persist, renewal tokens, tradeoffs for token expiration time versus convenience

Parameters in GET requests - insecure because of MITM, server logs, browser history and copy/paste/forward

JWT, JWT validation vulnerabilities: token validation with “none” algorithm, use public key as HMAC key

Response manipulation

- elevate privileges

Injection attacks. Describe injection attack. Mixing of command and data channels.

Why do we keep talking about injection attacks? This is why.

NoSQL injection vulnerability

NoSQL injection detailed

QA - Sempf QA

Fuzz testing

- Fiddler fuzz

API as an attack pivot point for attacking other internal systems / least privilege on both client and server

Enumeration and rate limiting defenses, GitHub API attack

Data storage

User secrets, use platform crypto with unique encryption keys (not shared or static keys)

Application secrets, reverse engineer application

OWASP Mobile top 10

Mobile vulnerability statistics

Threat modeling! / SDL tool

Questions

References

Resources

Metasploit / ZAP / Fiddler / Kali linux / Burp proxy