Security in Development

Lecture will be on Zoom - so could watch it from home

Development methodologies

- Waterfall
    - Used in small teams and software consultancies
    - Slowly dying
    - Pipelines
    - Going through things step by step
    - Issue with this is that security goes in the testing phase rather than from the requirements, design and implementation section
        - Is too late because there are only vulnerabilities there
        - Problem with catching them late - things might be hard baked into the code, you would have to go back steps to fix it
        - If you see things popping up the chances are that there will be other security issues popping up
        - You might have unit tests to catch these bugs, you don't do tests for already established infrastructure
- Agile - flexible, used most often in the real world

Test driven development

- Is important to develop with security in mind from the get go
- It takes more resources but it is important - is why a lot of companies avoid this
    - Developing tests takes resources, and if spec changes then your tests and resources are gone
    - Crunch time is real - people produce something that works as opposed to something that is secure
- Bugs in testing become bugs in production

Security and Vulnerability Classification

CVE - Common Vulnerability Enumerator (thee vulnerability)

- Just gets enumerator every time a new vulnerability is listed on this NVD site

CVSS - common vulnerability scoring system - the severity
National vulnerability database (NIST NVD)
Common Weakness Enumeration - the category the vulnerability fits into
Common platform enumeration - the vulnerable software

Everytime you get a zero day and you report this to NIST then you get a CVE (looks pretty good on CV)

- If it is a CPE and it is currently being produced then you can get a CVE from it

Just because there is a CVE this doesn't mean that there is a payload associated with it necessarily

https://www.exploit-db.com/

Static vs Dynamic Software Assessment (How do you find a CVE)

Static analysis

- Statically analyze without running it
- Can get certain tools that grab assembly instructions to do it in C
- Look at things that don't get run

Dynamic analysis

- Where you run something and see what changes on your computer
- Issues with this
    - Data corruption
    - Can't read config files
    - An attacker might know you're conducting an analysis if it sends out outbound signals when you are running things (remember these are web applications so requests are always getting sent out)
    - Time consuming
    - If you had a vulnerability exclusive to an operating system your dynamic analysis might not work because it is not based on your system
    - If you have an emulator the outcome might not be the same as if it was run on a physical machine
- Virtualisation - a way of getting around corrupting your own systems
    - Will need to have infrastructure, operating system then hypervisor and then the guest OS on top of that and then the application you want to run on top of that
    - Compared to running a container - e.g. docker
        - Interfacing layer between applications and libraries in the containers and the operating system
        - Instead of installing an entire OS, anytime your app looks at a command
        - Uses the base OS and figures out what the actual function call does to apply that
        - Designed to be one application, but you can trick a container to run multiple
    - We need to know these things because they have different artifacting
        - If you are running software you can tell if you are in a container vs virtual machine

- There is more risk in container in jumping between applications
- Hypervisor - heavier but harder to break out of to get to the underlying OS

Credential Management

- Can store it in environment variables
- Can store it in a file and read from that file when needed
- Important to do this from development because vulnerabilities that start in development get pushed to prod
    - Build good habits in developing and those habits will translate
- Don't push your private key
    - Don't think that just deleting it solves your problem
    - Git blame - will tell you how has committed a pieces of code
- How to key - don't hard code your keys
    - Store your stuff in separate files and ignore them in your git ignore
    - Have different keys for testing and deployment

Gidra - similar to IDAPro but free
Cutter (better than gidra)

Going through the content of the course - review

RECON

- DNS
    - Subdomains
- Passive vs Active
- Robots.txt
- Sitemap.xml
- Source code
- Tools:
    - Shodan
    - Subbrute
    - Way back machine
- Scope and ethics
- Methodology of recon

HELP SESSIONS

- Running after Andrew's tute on thursday
    - Will have a zoom call going

    - In the week 11 lecture time slot there will be another Help session running

Next week we will run through last year's exam

Key server - is a database on a server where you store those keys

- Reason you want todo this is that if your site gets compromised your keys don't get compromised as its not on the same server as the same server as the web application
    - Mtiigates impact should you get affected

Artifacting

- Artifact is a side effect of running a program