Penetration Testing

All of your base are belong to us
Today's Lecture

● ...is about penetration testing ("pentesting")
● Some is explicitly about when a service administrator asks you to test their security
● A lot is just about attacking in general
Pentesting

- No matter...
  - how good you are at security
  - how good your design is
  - how many people agree that it’s a good design
  - how good your implementation is
  - how many people agree that it’s a good implementation
  - ...you still could have made a mistake

- “Anyone, from the most clueless amateur to the best cryptographer, can create an algorithm that he himself can’t break.” - Bruce Schneier

What is Pentesting?

- Trying to break into a system with the goal of discovering vulnerabilities so they can be fixed
- It’s basically an attack with special parameters

“The main thing that separates a penetration test from an attack is permission.”
Parameters

● Important to set parameters
  ○ For the safety of personnel or systems
    ■ “Don’t impact other users with your testing.”
  ○ Because certain components are not considered security critical; don’t want to waste the effort
    ■ “Bugs… that prove the existence of a private repository or user [do not qualify].”
  ○ Because you may decide to accept certain risks
Parameters

- Most physical systems are vulnerable to some level of physical force
- Usually accepted that you’ll breach if…
  - You’re willing to do lots of physical damage
  - You’re willing to kill
- These are usually off limits for safety/cost
- Production or development system?
  - Facebook’s “test accounts”
Parameters

● Important parameter: what level of access?
  ○ Do you have the privileges of an employee?
  ○ Are you just a stranger off the street?
  ○ How much do you know about the security ahead of time?

● White box vs grey box vs black box
  ○ White box - you know everything about how the system works
  ○ Black box - you know nothing more than a member of the public
  ○ Grey box - somewhere in between
Dropbox

- Only server is a viable target
  - Client and network are off limits
- Fully white box
- Attacking a copy of the system, not the production system
Attacking

This section is about attacking in general - applies to more than just pentesting
Attacking Overview

- Three stages
  - Collect information
  - Search for potential vulnerabilities
  - Exploit
Collect information

● You want to know as much as possible about the target
  ○ More likely you’ll spot vulnerabilities
  ○ More likely to notice potentially vulnerable interactions between components
  ○ If you find a potential vulnerability, more likely to be able to figure out how to exploit it
Collect Information - Steps

- First, explore any documents you have - design documents, code, etc
  - These will be the most useful, since they’re meant to describe how the system works.
  - Trivial if it’s a white box test, but even if not, you may still be able to find documents. Try Google!
Collect Information - Steps

- Then (in no particular order):
- Look for information that the service exposes in order to operate
  - For a web service
    - What URLs is the web page requesting?
    - What information is sent to the server?
    - What cookies are stored?
  - For a company/building
    - Do they list a shipping address?
    - Publicly listed phone numbers?
    - “Employee portal” or similar?
Collect Information - Steps

- **Scan/probe**
  - If you have a domain name, look up IP address(es)
  - If you have IP address(es)
    - Scan for open ports (**nmap** is a great tool)
    - Scan other nearby IP addresses - might be owned by the same entity (but be careful not to attack somebody else!)
    - Fingerprint - you can often infer what services are running on a machine based on certain characteristics (try **nmap -O**)
  - If your target is a web site
    - Crawl links to enumerate URLs
Search For Vulnerabilities

- Using the information you’ve collected, construct as accurate and clear a picture as you can of how it works
  - What services are running? (ie, Apache, MySQL, etc)
  - How do components/services interact with one another?
Search For Vulnerabilities

● First, look for obvious vulnerabilities
  ○ Search for known vulnerabilities in particular services
    ■ Unpatched vulnerabilities
    ■ If they’re running an older version of a service, vulnerabilities that were not patched until later versions
  ○ Try common vulnerability classes
    ■ ie, for web sites, XSS, CSRF, etc
    ■ Attack libraries are very useful for this
      ● OWASP Top Ten
      ● MITRE CAPEC
Search For Vulnerabilities

- Second, look for security-critical components
- Then, if you know a lot about the architecture/have the code:
  - Look at the implementation of security functionality
  - Look for:
    - Complex or subtle functionality - easy to mess up
    - Functions that don’t seem to match their documentation
    - Signs of laziness or incompleteness
      - Commented out code
      - “TODO” comments
Search For Vulnerabilities

- Try to figure out if any of these components will behave in incorrect ways
  - Even if “incorrect” doesn’t seem like a security problem
- If a component has a bug, there’s a good chance it can be turned into a vulnerability!
Search For Vulnerabilities

- Third, *fuzz*
- Fuzzing is a technique in which you provide many randomly-generated inputs to a system to see if you can cause unexpected behavior
  - Unexpected behavior often indicates there’s a bug
  - Bugs can often be turned into vulnerabilities
- Try to focus on components that will be vulnerable to certain crafted inputs
  - For example, if you have a URL, example.com/user.php?uid=<uid>, fuzzing the page (user.php) is probably less fruitful than fuzzing uids
Search For Vulnerabilities

- Focus on components that may perform some kind of input validation or sanitization
- Try inputs that are unlikely to occur in practice - more likely that developers won’t have considered/tested
- If you break something, carefully inspect what went wrong - you may have discovered a vulnerability
Exploit

- Once you have a list of potential vulnerabilities, you need to figure out how to exploit them
- For each vulnerability, you want to know:
  - How can I access this vulnerability?
  - What can I do with this vulnerability?
Exploit

- For each vulnerability, you want to know:
  - How can I access this vulnerability?
    - What components could cause the vulnerable code to be executed? Do they pass it input?
    - What sorts of inputs can I cause the code to be executed with?
    - Can I figure out how to invoke the vulnerable code directly, bypassing input sanitization or validation?
  - What can I do with this vulnerability?
Exploit

- For each vulnerability, you want to know:
  - How can I access this vulnerability?
  - What can I do with this vulnerability?
    - What components can I affect?
    - What actions can I take? Can I access information? Make modifications?
    - If I can access information, can I use it to give myself more access (ie, steal passwords and use them to log in)?
    - If I can make modifications, can I use them to give myself more access (ie, change passwords to let me log in)?
    - Can I use it to access other vulnerabilities to exploit them?
Disclosure

- Without explicit permission, pentesting is very illegal
- Computer Fraud and Abuse Act (CFAA)
  - It is a crime to “intentionally access without authorization or in excess of authorized access”
  - Very broad definition - includes violating terms of service!
  - Large prison sentences - usually between 5 and 20 years
Disclosure

- Make sure you are absolutely sure you have permission
- Get everything in writing, signed
- Consult a lawyer
- Once you’ve done all this, make sure to stay in scope!
  - If I agree to let you test system X, I can still sue you for attacking system Y
  - If I agree to let you test system X, but I don’t own system X, whoever does can still sue you
Disclosure

- Sometimes, no explicit agreement, but a security policy
  - “We will not take legal action against you if you play by the rules.”
- Full disclosure
  - You publicize it
- Responsible disclosure
  - You tell the company privately, give them time to fix it, then publicize
- Coordinated disclosure
  - You tell the company privately, you both work together to fix and then publicize
Disclosure

- People don’t agree on what disclosure model is best!
  - Bruce Schneier on full disclosure
  - Microsoft on coordinated disclosure
  - OSVBD criticizing coordinated disclosure
Bug Bounties

- Selling exploits can be **profitable**
- Companies want to incentivize reporting vulnerabilities
  - Compete with incentive to sell exploits
  - For many companies, exploits are very bad press
- Many companies offer “bug bounties”
  - Google (up to $20k)
  - Facebook
  - Mozilla (up to $3K)
  - GitHub
  - HackerOne joint program