Computer Security

08r. Pre-Exam 2 Review

Paul Krzyzanowski

TAs: Fan Zhang, Shuo Zhang

Rutgers University

Fall 2019

This is not a review of everything we covered in the past 4 lectures

It's just a listing of some of the major concepts you should know

You should be familiar with these topics

If you some of this material is not familiar, this this indicates areas you need to focus on in preparing for the exam

We will not review everything from four lectures in 50 minutes!

Malware

Worms & Viruses

- Worm
  - Standalone software
- Virus
  - Requires a host program:
    a virus attaches itself to another piece of software
- Components
  - Infection mechanism: how does it spread?
  - Payload: what does the malware do?
  - Trigger (logic bomb): when will the payload run?

Some places where malware resides (1)

- File infector
  - Virus is part of an executable program
- Bootloader
  - Boot process invokes the malware
- Flash drive
  - Malicious software on the drive
  - or modified malicious firmware makes the drive send commands
  - or lost drive causes allows data to be stolen by someone else
- Macros
  - Office documents, editor files, PDF files
- Trojan horse
  - Useful program but also has a covert purpose

Some places where malware resides (2)

- Backdoors
  - Program with some undocumented mechanism to allow a user to log in or execute commands
- Rootkit
  - Modify the operating system, libraries, and/or commands to hide the presence of malware

Social engineering

#1 way of getting malware onto a system

- Get users to do something that isn't in their best interest
- Examples:
  - Phishing: email masqueraded to come from someone you trust with links or attachments you're asked to click on
  - Spear phishing: personalized phishing
  - Deceptive web sites: web sites that masquerade as legitimate companies or services
    - Phishing requests often take you there
    - Goal: steal login credentials
  - Deceptive content on web sites
    - Ads on file sharing sites often look like download links

Defenses against malware

1. **Signature-based scanning**
   - Scan the file system and download streams for malware
   - Signature = small part of the malware the (we hope) is unique to it
   - Accurate but requires knowledge of sample bytes of the malware
2. **Behavior-based monitoring**
   - Detect abnormal behavior
   - Difficult to detect what is an anomaly but works on new malware

Countermeasures taken by malware

- Use a **packer** to obscure the payload when it's in the file system
- **Polymorphic** viruses
  - Modify the code each time the virus propagates
- **Social engineering**: ask a user to override permissions

Cryptographic Systems

Ciphers

- **Symmetric**
  - Same key for encryption & decryption
- **Asymmetric** (public key)
  - Two related keys:
    encrypt with one, decrypt with the other
  - Based on **trapdoor functions**
- **Kerckhoffs's Principle**
  Use publicly known algorithms for cryptography

Classic cipher types

- **Monoalphabetic substitution cipher**
  - Replace one character (bit pattern) with another
  - Caesar cipher: the substitution alphabet is the alphabet shifted by *n* positions
  - Vulnerable to frequency analysis: certain letters are more likely than others
- **Polyalphabetic substitution cipher**
  - Change the substitution alphabet based on the position of the character
  - Still vulnerable to frequency analysis but more difficult
- **One-time pad**
  - Key = long set of random characters
  - Each key character encrypts one plaintext character
  - Problem
    - Key must be (1) truly random, (2) as long as the message, (3) never reused
- **Transposition cipher**
  - Scramble the data instead of substituting it

Ciphers

- **What is perfect secrecy?**
  - Ciphertext contains no information about the plaintext
- **Stream cipher**
  - Approximation of a one-time pad
  - Instead of a random stream of bits for the long key, create the stream using a pseudorandom number generator (**keystream generator**)
- **Block ciphers**
  - Encrypt a chunk of data at a time
  - Most ciphers we use are block ciphers – usually based on key size
    - Symmetric ciphers: **AES** (32- or 64-byte blocks), **DES** (8-byte blocks)
    - Public key ciphers:
      RSA (typically 8- or 16-byte blocks)
      ECC – **Elliptic Curve Cryptography** (typically 28-, 32-, or 64-byte blocks)

Using block ciphers

- **Electronic code book (ECB)**
  - Each block of plaintext is encrypted individually
- Problem
  - Common parts of plaintext will produce identical ciphertext
- Solution
  - **Counter mode**
    - An incrementing count is encrypted with the key for each block
    - Result is XORed with the block of plaintext to create ciphertext
  - **Cipher block chaining (CBC) mode**
    - Encryption of each block is a function of the previous blocks

Secure communication

Secure communication

- Symmetric cryptography
  - Encrypt and decrypt with a shared secret key
- Public key cryptography
  - Encrypt with the destination's public key
  - They decrypt with their private key
- Hybrid cryptography
  - Public key cryptography is *really slow …* and generating keys takes time
    - Use public key cryptography to send a random ***session key***
    - Then communicate with symmetric cryptography

Key Exchange

Key exchange

- **Diffie-Hellman** key exchange
  - Not an encryption algorithm: only key exchange
  - Forward secrecy
    - Means that even if you steal someone's private key, you cannot decipher their past communications
    - Requires single-use (ephemeral) keys
- **Needham-Schroeder** algorithm
  - Use a trusted 3rd party to send a session key
    - Use **nonces** (random bit strings) to guard against replay attacks
  - Denning-Sacco protocol: add timestamps to guard against replay attacks even if the attacker knows an old session key
  - Otway-Rees protocol: Use a random session ID for each message … also to guard against replay attacks if an old key is known
- **Kerberos**
  - Authentication, authorization, & key exchange
  - Essentially the Denning-Sacco protocol

Integrity

Integrity

- **Hash function** = strong checksum
- **Hash pointer** = pointer and *hash*(data pointed to)
  - Blockchain = linked list
  - Merkle tree = binary tree
- **MAC** = Message Authentication Code
  - *hash*(message, shared key) – cannot validate unless you know the key
  - Forms: HMAC, CBC-MAC
- Digital signature
  - Encrypted hash – using public key cryptography
- Digital certificates
  - { name, public key } signed by the issuer (the Certification Authority)

Authentication

Authentication Factors

Three factors:

1. Something you have (access token, card)
2. Something you know (password, PIN)
3. Something you are (biometrics)

Combining two or more of these factors leads to multi-factor authentication

Password Authentication Protocol

- Most commonly used authentication protocol
- Send a name & password
- Host system checks that the password is correct for the name

Keeping passwords secure

How do we keep passwords hidden on the host?

- Store the hash of the password

- Dictionary attack & brute force attack
  - Try either common or word-based passwords – or – all possible passwords
- If an attacker gets a list of hashed passwords
  - Building up a table of precomputed hashes makes it easy to do a reverse lookup to find the original password
  - Add salt to the password hash – a random string stored with each user
    - Hash(salt, password) makes it not practical to precompute hashes

One-time passwords

Three forms

1. Sequence-based: password = *f*(previous password)
   - Example: S/Key
1. Time-based: password = *f*(time, secret)
   - Example: Google Authenticator, RSA SecurID token
   - Also, Token-based: password = *f*(sequence#, secret)
     - Example: Yubikey
1. Challenge-based: *f*(challenge, secret)
   - Example: CHAP – challenge handshake authentication protocol

Pubic key authentication

- Alice has Bob's certificate
- How can Bob identify himself?
  1. Alice sends a random string (*nonce*) to Bob
  2. Bob encrypts it with his private key
  3. Alice decrypts it with the public key in Bob's certificate
     - If the decrypted value matches the original nonce, she is convinced Bob has the corresponding private key

Man in the middle attacks

- Someone can intercept & forward authentication messages
  - This allows them to take over communications *after* authentication is complete
- This is a problem for most authentication protocols
- Can be avoided with any of the following:
  - Send signed messages
    - Make sure *every* message comes from the right place
  - Use a trusted third party to generate session keys
    - Alice & Bob communicate and encrypt all their messages
  - Authenticate via public key cryptography (using certificates) and send a session key using public key cryptography
  - Establish a secure channel (e.g., using Diffie-Hellman) and then authenticate using public key cryptography

The end