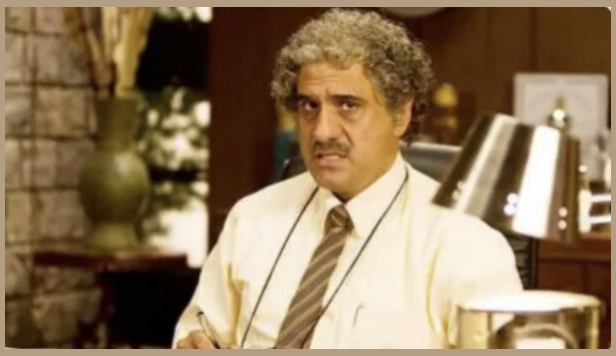
Malicious Code and Application Attacks

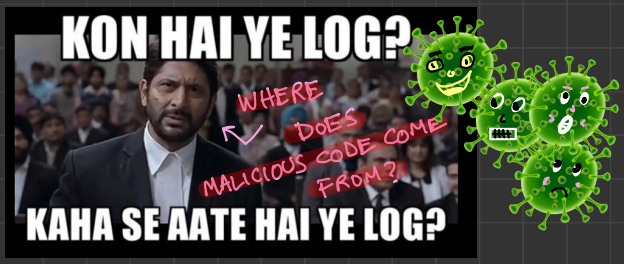
chapter 21 Review



Rashid

24/09/21

Malware – Malicious Software



Like biological virus, computer virus has two main fxn → Propagation
↓
Payload Execution

Virus propagation techniques

- Master Boot Record Viruses
- File Infector Viruses
- Macro Viruses
- Service Injection Viruse

These viruses attack the MBR
A small portion of code is stored in MBR, and the rest in storage media. MBR viruses act by redirecting the system to an infected boot sector, which loads the virus into memory before loading the OS from the Legitimate boot sector

- File Infector Viruses → infect different types of executable files and trigger when OS attempts to execute them
  - Companion viruses → self contained executable files. Take filenames of legitimate files (similar filenames)
- Macro Viruses → Exploits Macros. Restricting the use of untrusted macros to run without explicit user permission contains them
- Service Injection Viruse → Malicious code injects itself into trusted runtime process of the OS, such as svchost.exe, winlogon.exe, explorer.exe

Virus Technologies

Multipatriate viruses
↓
Use more than one prop technique

Stealth viruses
↓
Hides by actually tampering OS to fool AV that everything is fine.

Polymorphic Viruses
↓
Modify their own Code as it travels from one system to another

Encrypted viruses
Use cryptographic techniques to hide from detection

Logic Bombs

→ Malicious code objects that lie dormant until they are triggered by the occurrences of one or more conditions like time, program launch, website logon, certain keystroke etc.

Trojan Horses → A software program that appears benevolent but carries a malicious behind the scene payload.

RATs – Remote Access Trojans → Opens backdoor into a system.

Trojans and other Malwares that perform Cryptocurrency mining are also known as Crypto malware

Worms – Propagate themselves w/o any human intervention.

Spyware – monitors action and tx important information to remote systems ← User might consent to install, but they carry out function that user do not wish to authorise

↑ → Adware – display advertisements on infected computers

Potentially unwanted Programs (PUP) takes adv of 3rd Party plugins. (web Browser)

Ransomware → Use encryption as a weapon to ransom

Malicious Scripts → Taking advantage of existing automation and scripts

Fileless malware → Run entirely in memory, do not create any file/log on disk, hence undetected

Zero-Day Attacks → Exploits Zero-Day vulnerability

Delay b/w discov of a new malicious code and issuance of Patches

Slowness in applying updates

Malware Prevention

Anti malware softwares → Eradication → Disinfects the affected file and restore the machine

→ Quarantine → User examines quarantine files manually

→ Deletion → If the file exceeds a predefined danger threshold

Signature Based

Heuristic mechanism

Integrity Monitoring → Tools designed to monitor file modifications. Maintains a database of hash values for all files stored on the system

Endpoint Detection and Response → Analyse endpoint memory, filesystem and network actvt

↓

Analytic focus on endpoint

Automatically isolate potential malicious actvt

Integration c̄ threat intelligence source to obtain real time insight into malicious behavior elsewhere on internet.

UEBA - User and Entity Behavior Analysis

↓

Analytic focus on enduser

Integration c̄ other incident response mechanism

Application Attacks

Buffer Overflow → vulnerability exist → Lack of validation for user input

↳ Remediation → Value size should not be longer than the buffer space

→ Variable type should match

→ Value should be within the parameter

Timing → STATE Attack ← System state Transition

STATE Attack → Data flow control

TOCTTOU – Time of Check to Time of Use – Race Condition

↳ Replace the original object in the time difference b/w TOC & TTU

Require in-depth knowledge of program and system under attack

Backdoors → Undocumented cmd sequences that allow individuals w/ knowledge of the backdoor to bypass normal access restrictions

Privilege Escalation and Rootkits

↑ are used to achieve ↵

Injection Vulnerabilities

A specific example of

↓ Code Injection Attack

SQL Injection Attack → Blind Timing based SQL Injection

Amount of time required to process a query can be used as a channel for retreiving info from a database

↓

Blind Content based SQL Injection

Prepetrator sends input to the web application that tests whether the application is interpreting injected code before attempting to carry out an attack

Code injection Attacks

Any env that inserts user-supplied input into code written by an app developer may be vulner to Code injection attacks

→ LDAP injection

→ XML injection

→ DLL injection

Input validation, input escaping and defensive coding are essential to eliminate these threats.

Command Injection attacks → Application code reaches back to the OS to execute a command, causing OS level changes.

Exploiting Authorisation Vulnerabilities

Insecure Direct Object Reference – Lack of specific authorisation for object access

Directory Traversal — A misconfiguration/vulnerability ch̄ allows users to navigate thē directory structure and access files that should remain secure.

File Inclusion – Next level of Directory Traversal Attack. File inclusion attack actually execute thē code contained within a file.

File Inclusion Vuln is exploited by Web Shell.

- Local File Inclusion Attack
- Remote File Inclusion Attack

Exploiting Web Application Vulnerability

Cross-Site Scripting Attack → when web application allow an attacker to perform HTML injection.

- **Reflected XSS**
  Application allows reflected input.
  – Remediate using input validation, input pattern matching, Output encoding.
  → Transforms potentially dangerous content into safe form
- **Stored/Persistent XSS**
  Stores XSS code on a remote server

Some XSS attack work by modifying Document Object Model (DOM) environment within thē user's browser. Thesē attack don't appear in HTML code.

Request Frogery → Exploits trust relationship and attempt to have users unwittingly execute commands against a remote server

- **CSRF/XSRF**
  Cross-Site
  → Tricking a user
- **SSRF**
  Server-side
  → Tricking a server

XSS attack exploits the TRUST that a user has in a website to execute code on the user's computer

VS.

XSRF attack exploits the trust that remote site have in user's system to execute command on user's behalf.

Reasonable assumption – users are often logged into many different websites at the same time.

Attackers then embed code in one website that sends a command to another website

XSRF Protection → Secure tokens, verify refering URL that it is originated from their own site and received from end user only.

SSRF → These attacks are possible when a web application accepts URLs from a user as an input and then retreives information from the URL.

Session Hijacking → Communication Interception and session takeover and assuming identity of the authorised user.

- Making Attacker as a server
- Authentication capturing
- Using Cookie Data

Remediate using
- anti-replay authentication technique.
- Cookie Expiry within reasonable time.

Application Security Controls

Metacharacters

' " [ ] \ ; & ^ $. | ? * + { } ( )

Input Validation – Input whitelisting / Allow list

- Should occur at server side
- Client side validation is useful for providing users c feedback on their input.

Input Blacklisting – Restrictions of HTML tags, SQL commands

Parameter Pollution – depends on defects in web platforms that don't handle multiple copies of the same parameter properly

Web Application Firewalls – Works at application layer of OSI Model
Does input validation – whitelisting / Blacklisting

Database Security

Parameterised Queries → Developer prepares a SQL statement, and then allows user input to be passed into that statement as carefully defined variables that do not allow the insertion of codes

Protects Applications against Injection Attacks ←

Stored Procedures → Here SQL code is not contained within the application, but is stored on the database server.
- The client does not directly send SQL code to the database server
- Client sends arguments to the server, which then inserts those arguments into a precompiled query template

↓

Protects against injection attacks, and improves database performance as well

Obfuscation and Camouflage

- Data minimization: Collect only data needed
- Tokenization: Replaces PII ε a unique identifier
- Hashing: Replace sensitive information ε salted hash

Code Security

Code Signing – Digitally signing the code as a proof of code's authorship legitimacy and integrity.

Code reuse – SDKs

Software diversity – Avoid SPOF

Code Repositories – Centralised location for storage and mgmt of source code
- Version control, promotes code reuse, help avoid the problem of dead code

Integrity measurement – using cryptographic hash to verify code release

Application Resilience

Scalability

- Vertical scaling - Scaling UP
- Horizontal scaling – Scaling out
- Capability of incremental addition of resources

Elasticity

- Automatic Provisioning and deprovisioning

Common Features of Cloud Platforms

Secure Coding Practices

Source Code Comments → Remove comments from Production web Apps
- Executable Files auto remove Comments
- Keep comments secure in a software version ch is stored securely

Error handling → Use minimum information necessary for the user to understand the problem

Hard Coded Credentials → Take care in not storing credentials ch can fxn as a backdoor
- Inclusion of the service access credentials in the code

Memory Management

Resource Exhaustion → Uncontrolled or unchecked use of computing resources by Apps

↳ Memory Leak → App fails to return memory to the system, perhaps by losing track of written objects

Pointer Dereferencing → Null pointer exception can provide an attacker access to debugging information, that may be used for reconnaissance of the application's security

May allow an attacker to bypass security controls