|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Software security   |  |  | | --- | --- | | Software Quality | Software Security | | accident | By attacker/不會被測試到 |   Handling Program Input:  size(overflow), interpretation(fail:ret too much data)  Injection Attacks:  $user=$q.para(“user”)  Print(‘/usr/bin/finger -sh $user’)  // User=‘xxx; echo attack success; ls –l finger\*  // unless($user=~/^\w+$/)  PHP 2 feature: declare global variable in url / PHP can include url  SQL Injection ExampleSol. Replace(“ ’; ”)  Cross-site Scripting (XSS) Attacks  **Input provided to a program by one user that is subsequently output to another user/** Script code may need to access data associated with other pages / Assumption : all content from one site is equally trusted and hence is permitted to interact with other content from that site  Attacks exploit this assumption and attempt to bypass the browser’s security checks / **Involving the inclusion of script code in the HTML content of a Web page displayed by a user’s browser**  XSS Reflection 留言+script code Prevent input要檢查  Validating Input Syntax 使用input前檢查/建立白黑名單/正規  Input Fuzzing randomly generated data as inputs to a program  **PROS** Simplicity and freedom from assumptions / low cost  用template→有多做假設  **CONS** very simple, but identifying only simple types of faults  (e.g., only triggered by a small number of very specific input values)  Correct Machine Instructions for Algorithm machine code搞你  Correct Data Interpretation data type/C can allocate memory(小心)  Preventing Race Conditions with Shared Memory  Program搶os資源→**[Sol]** correct selection and use of appropriate synchronization primitives **But,** deadlock can be still an issue Attackers may trigger the deadlock to launch DoS  Interacting with the OS and Other Programs  Environment variables/ Using appropriate, least privileges/ Systems calls and standard library functions/ Preventing race conditions with shared system resources/ Safe temporary file use  System Calls and Standard Library Functions  Optimizations can conflict with program goals  Ex: Securely Delete a File  System will write the new data to same disk blocks/ Data are written immediately to disk/ When the I/O buffers are flushed and the file is closed, the data are then written to disk  Handling program output Programs must identify what is permissible output content/ Filter any possibly untrusted data to ensure that only valid output is displayed | Out of band Attacks Data are retrieved using a different channel, **e.g.**, email instead of web pages/ Used when there are limitations on information retrieval/ **But,** outbound connect from the data server lax  Inferential Attacks Reconstruct the information by sending particular requests and observing the resulting behavior of the Website/database server **→Illegal/logically incorrect queries/ Blind SQL injection**  Database Access Control  Cascading Authorizations GRANT/REVOKE  Role Based Access Control (RBAC) app owner / end user / admin  Inference 用可以access的data去infer sensitive data  Detection altering the database structure / Eliminate an inference channel violation during a query time  Database Encryption CONS: inflexibility(hard to search)/ key manage   |  |  | | --- | --- | |  | More flexible：index | |  |  |   Cloud Security  Malicious Software  Advanced Persistent Threats (APTs) target selection, persistent, and stealthy  Virus dormant propagation trigger execute  用目標分類Boot/ file infector/ Macro infector(PDF WORD)/multipartite  用策略分類encrypt/ stealth(hide anti)/polymorphic/ metamorphic(全變)  Worm multiplatform/ multiexploit/ ultrafast spread/ polymorphic/ metamorphic/ transport vehicle/ **Zero-day** exploit  Clickjacking UI redress  **Worm** propagates itself and activates itself  **Bot** is initially controlled from some central facility (use IRC server)  Backdoor(bypass security)Difficult to implement OS controls for backdoors in apps Maintenance hook: a backdoor used by programmers to debug  Rootkit 偷偷取得root privilege→alter system’s standard function  Generic decryption (GD) detect virus decrypt itself  Host based Behavior Blocking software cause harm before it be detected  **Intruder Behavior**  Target acquisition and information gathering/ Initial access/ Privilege escalation/Information gathering or system exploit(找你要的data)  **Maintaining access**→Installing backdoors rootkits / Covering tracks  Anomaly detection collect data→analyze   |  |  |  | | --- | --- | --- | | **Statistical** | **Knowledge based** | **Machine learning** | | simplicity low compu-  tation cost lack of assumptions about behavior expected | robustness and  flexibility | flexibility adaptability , and ability to capture  interdependencies  between factors | | difficulty in  selecting suitable mtrics ,and not all behaviors can be modeled | difficulty/time required to develop high quality knowledge rules | requiring significant  time and computational  resources |   **Limit** cannot detect unknown data  Signature/Heuristic detection Using a set of known malicious data  patterns (signatures) or attack rules **/ Rule based heuristic identification**  Intrusion Prevention (IPS) block or prevent detected malicious activity  HIPS sandbox **Pros:** various tools work closely together  Compile time defenses Aim to harden programs to resist attacks in new programs  Choice of Programming Language cost in resource use  Safe Coding Techniques  Language Extensions & Safe Libraries Handling dynamically allocated memory: more problematic / The size information is not available at compile time/ Requiring an extension and the use of library routines  **Cons**  Generally, there is a performance penalty/ Programs and libraries need to be recompiled with the modified compiler/ Feasible for new OSes, but likely to have problems with third party apps  Stack Protection Mechanisms  Canary Add function entry and exit code to check stack for signs of corruption **CONS** All programs needing protection need to be recompiled/ The structure of the stack frame has changed: causing problems with programs, e.g., debuggers  **Return Address Defender (RAD)** Do not alter the structure of the stack. Run time defenses Aim to detect and abort attacks in existing programs Function **entry code:** writing a copy of the return address to a safe region of Memory/ Function **exit code**: checking the return address in the stack frame against the save copy  RunTime Defenses Aim to harden programs to resist attacks in new program  Can be deployed as OS updates to provide protection  Compile time approaches: usually require recompilation of existing p0rograms/ Involving changes to the memory management  Executable Address Space Protection Block the execution of code on the stack/ Against the attacks: copying machine code into the targeted buffer and then transferring execution to it / Tag pages of virtual memory as being nonexecutable  Address Space Randomization Guard Pages between critical memory  Heap Overflow no easy transfer of Control/ May have function pointers to be exploited Or manipulate management data structures  Global Data Overflow |
| DBMS Architecture  **Security:** beyond the capability of typical OS based security  →OS: typically control read and write access to entire files  **Primary key**→Uniquely identifies a row  **Foreign key**→Links one table to attributes in another  **SQLi** Hacker injects an SQL command to a database sending the command to the Web server  →Modify or delete data/ Execute OS commands/ Launch DoS  **e.g.** Redmond’; DROP table OrdersTable -- / name: 1’ OR ‘1’=‘1  X\_FORWARDED\_FOR :127.0.0.1' or 1=1#  username: XXX ' OR username='JANE // second order inject  →已知資訊再搞更深入的cookie  In band attacks use the same communication channel for injecting SQL codes and retrieving  **Tautology**: condition always true  **End of line commen**t/ **Piggybacked querie**s  Host Intrusion Detection (HIDS) can detect both external, internal intrusions  Data Sources and Sensors (Cont.) System call traces/ **Audit (log file) records**(不須其餘軟體/可能沒有needed data)/ **File integrity checksums**(Cons: generate and protect the checksums, difficult to monitor changing files)/ **Registry access**  Signature or Heuristic HIDS Efficient at detecting known malware, but not capable of detecting zero day attacks  NIDS: examines packet traffic toward potentially vulnerable systems on a network HIDS: examines user and software activity on a host  **NIDS becomes to not function well:** encrypted packet   |  |  | | --- | --- | | Inline sensors | Passive sensors | | Pros: no additional separate hardware devices are needed  Cons: negative impact on network | more efficient and doesn’t contribute to packet delay | | Low interaction honeypot | High interaction honeypot | | Emulating particular IT services or systems well enough to provide a realistic initial interaction, but does not execute a full version Providing a less realistic target Often sufficient for use as a component of a distributed IDS to warn of imminent attack | A real system, with a full OS, serv-ices and applications, which are instrumented and deployed where they can be accessed by attackers A more realistic target that may occupy an attacker for an extended period However, it requires signi- ficantly more resources If comp- romised, could be used to initiate attacks on other systems |   Firewall  **Limit** An improperly secured wireless LAN may be accessed from outside/ Devices infected outside are attached and used internally  Packet Filtering Firewall Applying a set of rules to each incoming and outgoing IP packet(rule based) **PROS** simple transparent fast  **Cons** Cannot prevent attacks that employ app specific vulnerabilities or functions/ Limited logging functionality/ Don’t support advanced user authentication, due to the lack of upper layer functionality/ Vulnerable to attacks on TCP/IP protocol issues/ Susceptible to security breaches caused by improper configurations  Possible attacks IP address spoofing(**Countermeasure**: discarding incoming packets with an inside source address)/ Source routing attacks(**Cm:** discarding all packets that use this option)/ Tiny fragment attacks(**Cm:**enforcing the first fragment of a packet to contain a predefined minimum amount of the transport header)  Tradition PF→Weakness Doesn’t take into consideration any higher layer context / Client port>1024→vulnerbility  Stateful Inspection Firewalls Keeping track of TCP sequence number  Application Proxy Firewall A relay of app level traffic/ Must have proxy codes for specific apps/ May restrict supported app features  **Pros:** more secure than packet filters **Cons:** additional processing overhead on each connection  Circuit level Proxy Firewall Splitting a TCP connection  Doesn’t examine the contents/ Reduce the overhead of the app level proxy(Inbound: app level proxy, outbound: circuit level proxy)  SOCKS A framework for client server apps in TCP/UDP domains to conveniently and securely use the services of a network firewall  Bastion Host A hardened system identified by the firewall administrator as a critical strong point in the network’s security  Host based (**Server** based) Firewalls **Pros** Filtering rules can be tailored to the host environment/ Protection is provided independent of topology/ Providing an additional layer of protection  Personal Firewalls(router)  Much less complex than server based and stand alone firewalls/ to deny unauthorized remote access  DMZ (Demilitarized Zone) externally accessible but need some protections  Virtual Private Networks (VPN)  Using encryption and authentication in the lower protocol layers to provide a secure connection through an insecure network |