Department of Computer Science

Computer Security

Exam 3

December 2, 2024

Solutions & Discussion

Prepared with tender loving care by Paul Krzyzanowski

100 POINTS – 25 QUESTIONS – 4 POINTS EACH – For each statement, select the *most* appropriate answer.

1. What is the primary purpose of using *packers* in malware?
   (a) To infect executable files.
   (b) To encrypt files in a system in a ransomware attack.
   (c) To evade detection by anti-virus software.
   (d) To exfiltrate the victim's data to a remote server.

Packers compress or encrypt malware code to make it harder for anti-virus software to analyze and detect. This technique obscures the malware's content, allowing it to bypass signature-based detection systems.

(a): Packers do not infect executable files; they are used to conceal malware but do not spread it.
(b): Encrypting files in a system is a characteristic of ransomware, not packers.
(d): Packers do not exfiltrate data; this is typically the function of spyware or trojans.

2. Which of the following best describes *polymorphic* malware?
   (a) Malware that exploits vulnerabilities across multiple network layers.
   (b) Malware that infects multiple file types.
   (c) Malware that works across different operating systems.
   (d) Malware that changes its code to evade detection.

Polymorphic malware modifies its code or appearance each time it infects a system, making it difficult for traditional anti-virus software to detect using signature-based methods.

(a): Polymorphic malware does not exploit vulnerabilities across network layers; it focuses on evasion.
(b): While polymorphic malware might infect various file types, its defining feature is code modification for evasion.
(c): Polymorphic malware is not inherently cross-platform; it targets specific systems but disguises its code.

3. A *rootkit* is:
   (a) Software designed to hide certain processes or programs from detection.
   (b) Software that installs itself in a bootloader to run before the operating system boots.
   (c) A framework for building viruses and worms.
   (d) Malicious software that allows the attacker to gain admin (root) privileges.

A rootkit's primary function is to conceal the presence of malicious software or unauthorized activity by modifying system tools and processes, making detection and removal challenging.

(b): This describes a bootkit, which targets the bootloader or MBR, not rootkits in general.
(c): A rootkit is not a framework for building malware; it is a type of malware that focuses on stealth.
(d): While rootkits may grant administrative (root) privileges, their defining characteristic is their ability to hide malicious processes.

4. What distinguishes a *Trojan* from other types of malware?
   (a) It self-replicates across networks.
   (b) It encrypts user files and demands a ransom.
   (c) It disguises itself as legitimate software while performing malicious actions.
   (d) It exploits vulnerabilities in network protocols to provide remote access to attackers.

A Trojan appears to be legitimate software, tricking users into installing it. Once executed, it performs malicious actions, such as stealing data or creating backdoors.

(a): Self-replication is characteristic of worms or viruses, not Trojans.
(b): Encrypting files and demanding a ransom is specific to ransomware, not Trojans.
(d): Exploiting network protocol vulnerabilities is more aligned with worms or targeted exploits.

5. How does *spear phishing* differ from regular phishing?
   (a) It targets multiple victims simultaneously.
   (b) It relies on exploiting vulnerabilities rather than social engineering.
   (c) It uses tailored messages for specific individuals.
   (d) It cannot exfiltrate data.

Spear phishing involves crafting personalized messages aimed at specific individuals to increase the likelihood of success.

(a): Regular phishing targets multiple victims simultaneously, often with generic messages.
(b): Spear phishing relies heavily on social engineering, not technical vulnerabilities.
(d): Spear phishing can exfiltrate data if the target is deceived.

6. Which of the following is a limitation of *signature-based* malware detection?
   (a) High false positive rate.
   (b) Dependence on network firewalls.
   (c) Inability to detect unknown malware.
   (d) Overhead in sandboxing applications.

Signature-based detection relies on known patterns or signatures. It cannot identify new, unknown, or polymorphic malware.

(a): Signature-based detection generally has a low false positive rate, though not always.
(b): Network firewalls are unrelated to the signature detection process itself.
(d): Signature-based detection does not involve sandboxing, so there is no overhead related to it.

7. How did the example in Ken Thompson's *Reflections on Trusting Trust* paper demonstrate the persistence of a malicious backdoor?
   (a) By modifying the source code of an operating system.
   (b) By introducing vulnerabilities through linked libraries.
   (c) By exploiting runtime execution flaws in compiled programs.
   (d) By embedding a self-propagating backdoor into the compiler itself.

Ken Thompson's example involved modifying a compiler to insert a backdoor into code it compiled. This backdoor was self-propagating because the modified compiler would also embed the backdoor into subsequent compilers it created.

(a): The source code of the OS was not directly modified; the backdoor was introduced via the compiler.
(b): The backdoor was not introduced through linked libraries but through the compiler itself.
(c): Runtime execution flaws were not the mechanism demonstrated in this example.

8. What is the primary goal of a CAM overflow attack?
   (a) To force a switch to broadcast traffic to all ports.
   (b) To generate high network traffic so legitimate packets will be dropped.
   (c) To disable all VLAN configurations on a switch.
   (d) To redirect traffic to an attacker's MAC address.

A CAM overflow attack floods the Content Addressable Memory (CAM) table of a network switch with fake MAC addresses. This forces the switch to behave like a hub, broadcasting traffic to all ports, which enables attackers to intercept data.

(b): CAM overflow does not aim to generate high traffic volume for dropping legitimate packets; that is more characteristic of a DDoS attack.
(c): CAM overflow does not directly disable VLAN configurations.
(d): The attack does not redirect traffic to a specific MAC address but causes the switch to broadcast traffic to all devices.

9. What is the purpose of SYN cookies in defending against SYN flooding?
   (a) To encrypt SYN packets.
   (b) To verify the client's IP address.
   (c) To block unauthorized connections from untrusted IP ranges.
   (d) To validate incoming SYN-ACK packets without storing state information.

SYN cookies mitigate SYN flooding by encoding connection state information in the sequence number of SYN-ACK packets. This allows servers to handle SYN requests without keeping track of every pending connection, reducing resource exhaustion.

(a): SYN cookies do not encrypt SYN packets; they modify the sequence number for validation.
(b): They do not verify the client's IP address directly; they validate the connection request itself.
(c): SYN cookies are not designed to block connections from specific IP ranges; firewalls typically handle this.

10. How can attackers use DNS to covertly communicate with a command-and-control (C&C) server?
   (a) By redirecting all DNS traffic to a public DNS server.
   (b) By embedding encrypted commands in DNS queries and responses.
   (c) By using DNSSEC to bypass firewalls.
   (d) By poisoning DNS cache entries on the target system.

Attackers can exploit DNS by embedding commands or data in the payload of DNS queries and responses. Since DNS traffic is often trusted and less scrutinized, it provides a covert channel for communication with a C&C server.

(a): Redirecting DNS traffic to a public server does not inherently establish covert communication.
(c): DNSSEC enhances security and does not bypass firewalls.
(d): Poisoning DNS cache entries alters traffic flow but is unrelated to covert communication channels.

11. How does a *DNS rebinding attack* bypass the same-origin policy?
   (a) By using short TTL values to later change the IP address of a domain name.
   (b) By embedding malicious JavaScript into a trusted webpage.
   (c) By exploiting DNSSEC vulnerabilities.
   (d) By modifying the client's `hosts` file.

DNS rebinding attacks exploit the browser's trust by initially resolving a domain name to an external IP address, and then, after a short TTL expires, rebinding it to an internal (private) IP address. This enables access to internal network resources despite the same-origin policy.

(b): Embedding malicious JavaScript in a webpage is not related to DNS rebinding but to XSS.
(c): DNSSEC does not directly create vulnerabilities but helps authenticate DNS records.
(d): Modifying the client's hosts file could redirect traffic, but it is not how DNS rebinding works.

12. Which of the following is an effective defense against *DNS rebinding* attacks?
   (a) Implementing a firewall to block external DNS requests to private IP addresses.
   (b) Using DNSSEC to authenticate DNS records.
   (c) Decreasing the TTL of DNS responses.
   (d) Using randomized query IDs for DNS requests.

Firewalls can prevent DNS rebinding by blocking requests that attempt to access private IP addresses from external domains, thus protecting internal resources.

(b): DNSSEC prevents DNS cache poisoning but does not stop DNS rebinding.
(c): Decreasing TTL does not prevent rebinding; in fact, short TTLs are exploited in such attacks.
(d): Randomized query IDs protect against cache poisoning but are not relevant to rebinding.

13. What does a *BGP hijacking attack* do?
    (a) Floods the network with spoofed BGP packets.
    (b) Redirects traffic to malicious servers.
    (c) Exploits a vulnerability in a router to modify its routing table.
    (d) Creates new IP prefixes in routing tables.

BGP hijacking involves announcing false IP prefix routes, causing internet traffic to be misrouted to malicious servers controlled by attackers.

(a): Flooding with spoofed BGP packets describes a different type of attack and does not define hijacking.
(c): Exploiting a vulnerability in a router may play a role, but hijacking is achieved via false route announcements, not direct modifications.
(d): Creating new IP prefixes in routing tables is unrelated to BGP hijacking.

14. What is the main difference between the *transport* and *tunnel* modes in IPsec?
    (a) Transport mode encrypts only the payload, while tunnel mode encrypts the entire packet.
    (b) Transport mode encrypts only the IP header, while tunnel mode encrypts the entire packet.
    (c) Transport mode requires pre-shared keys, while tunnel mode requires digital certificates.
    (d) Transport mode is used for multicast traffic, while tunnel mode is used for unicast traffic.

In **transport mode**, only the data payload is encrypted, leaving the IP header intact. In **tunnel mode**, both the payload and the entire original IP packet, including the header, are encrypted and encapsulated in a new IP packet.

(b): Transport mode does not encrypt the IP header; tunnel mode does.
(c): Both modes can use pre-shared keys or digital certificates, so this is not a distinguishing feature.
(d): IPsec modes are not specifically tied to unicast or multicast traffic types.

15. What is the primary purpose of *TLS* (Transport Layer Security)
    (a) To provide encryption, authentication, and integrity for application-layer communication.
    (b) To establish a secure connection for all traffic between networks.
    (c) To create a private network tunnel over a public network.
    (d) To secure routing tables for internet traffic.

TLS is designed to secure application-layer communications (e.g., HTTPS) by encrypting data, authenticating the communicating parties, and ensuring message integrity.

(b): TLS secures specific application-layer traffic, not all network traffic.
(c): Creating private tunnels is the role of VPNs, not TLS.
(d): TLS does not secure routing tables; it focuses on securing communications.

16. How does *TLS* differ from VPNs in terms of functionality?
    (a) TLS operates at the network layer, while VPNs operate at the application layer.
    (b) TLS encrypts IP headers, while VPNs encrypt only the payload.
    (c) VPNs rely on public key cryptography, while TLS relies on symmetric encryption only.
    (d) VPNs secure all network traffic between endpoints, while TLS secures specific application communications.

TLS is used for securing specific protocols (e.g., HTTPS), while VPNs encrypt all traffic between the user and the VPN server, providing network-wide security.

(a): TLS operates at the transport layer, not the network layer.
(b): TLS does not encrypt IP headers; it encrypts application data.
(c): Both TLS and VPNs use public key cryptography during key exchange; this is not a distinguishing feature.

17. What is the primary reason organizations adopt a *zero trust* model?
    (a) To simplify network configurations.
    (b) To assume that internal networks are not automatically secure.
    (c) To eliminate the need for endpoint security tools.
    (d) To replace encryption with other security mechanisms.

Zero trust assumes that no user or device, whether inside or outside the network, is trusted by default. This requires verification of all access requests.

(a): Zero trust does not simplify network configurations; it often adds complexity.
(c): Zero trust does not eliminate the need for endpoint security tools; it complements them.
(d): Encryption remains an integral part of zero trust models and is not replaced.

18. Which of the following is a limitation of *stateless packet filtering* firewalls?
    (a) They do not track connections, making them vulnerable to spoofing.
    (b) They cannot block packets based on IP addresses.
    (c) They cannot filter packets based on the transport layer protocol.
    (d) They require deep packet inspection for effectiveness.

Stateless firewalls filter packets based on predefined rules (fields in IP and TCP/UDP headers) but do not maintain information about active connections, leaving them vulnerable to spoofed packets.

(b): Stateless firewalls can block packets based on IP addresses.
(c): They can filter packets based on transport layer protocols like TCP or UDP.
(d): Deep packet inspection is not required for stateless firewalls; it is a feature of more advanced firewalls.

19. How does a *reflection amplification* attack increase the effectiveness of a DDoS attack?
    (a) By sending large volumes of ICMP packets to the target.
    (b) By targeting the target's routers with malformed packets.
    (c) By directly accessing the target's resources with repeated HTTP requests.
    (d) By leveraging remote UDP-based services to generate responses to spoofed requests.

Reflection amplification attacks exploit services like DNS or NTP that respond with large amounts of data to small queries. By spoofing the source IP address, attackers redirect these amplified responses to the target.

(a): Sending large volumes of ICMP packets describes an ICMP flood, not a reflection amplification attack.
(b): Malformed packets target vulnerabilities in routers but do not involve reflection or amplification.
(c): Repeated HTTP requests describe HTTP flooding, which is unrelated to reflection amplification.

20. Which of the following is considered the *same origin* according to the same-origin policy?
    Assume `www.example.com` and `example.com` share the same IP address.
    (a) `http://example.com` and `https://example.com`
    (b) `http://example.com` and `http://www.example.com`
    (c) `http://example.com:8080` and `http://example.com`
    (d) `http://example.com/page1` and `http://example.com/page2`

The same-origin policy considers two resources to share the same origin if their protocol, domain, and port are identical. Both URLs in (d) meet these criteria.

(a): The protocols differ (http vs. https), so they are not considered the same origin.
(b): The subdomain www makes the origins different.
(c): The port numbers differ (8080 vs. default 80), making them distinct origins.

21. What does the `Secure` flag on a cookie do?
    (a) Encrypts the cookie before transmission.
    (b) Restricts access to the cookie by client-side scripts.
    (c) Prevents the cookie from being shared across domains.
    (d) Ensures the cookie is only sent over HTTPS.

The Secure flag ensures that cookies are transmitted only over encrypted HTTPS connections, protecting them from interception over insecure HTTP channels.

(a): The Secure flag does not encrypt cookies; it only restricts their transmission method.
(b): Restricting client-side script access is achieved by the HttpOnly flag, not the Secure flag.
(c): Preventing cross-domain sharing involves SameSite or domain-specific attributes, not the Secure flag.

22. What makes *CSRF* attacks possible?
    (a) Users reusing passwords across multiple sites.
    (b) Browsers automatically sending cookies with requests.
    (c) Exploiting software vulnerabilities in the operating system.
    (d) Using outdated DNS records.

Browsers automatically attach cookies for the target site when sending requests, even if the request is crafted by a malicious third-party website. This behavior enables CSRF attacks.

(a): Reusing passwords is unrelated to CSRF attacks.
(c): CSRF does not exploit OS vulnerabilities; it abuses browser behavior.
(d): Outdated DNS records are unrelated to CSRF mechanisms.

23. What is the main purpose of *CORS* (Cross-Origin Resource Sharing)?
    (a) To prevent XSS attacks.
    (b) To ensure cookies are only sent over secure channels.
    (c) To allow controlled access to resources across domains.
    (d) To enforce the same-origin policy.

CORS is a protocol that allows servers to specify which origins are permitted to access their resources, relaxing the restrictions of the same-origin policy in a controlled manner.

(a): CORS does not prevent XSS attacks; it manages cross-origin resource sharing.
(b): Ensuring cookies are sent securely is related to the Secure flag, not CORS.
(d): CORS provides mechanisms to bypass the same-origin policy, not enforce it.

24. Why might a *combosquatting* attack be more effective than typosquatting?
    (a) It uses misspellings that may be hard to notices.
    (b) Parts of the domain name refer to a legitimate site.
    (c) It combines typographic attacks with cross-site scripting.
    (d) It injects malicious JavaScript into pages from legitimate websites.

Combosquatting combines legitimate domain names with misleading terms (e.g., paypal-login.com), making the domain appear more trustworthy to users than misspellings typically used in typosquatting.

(a): Using misspellings describes typosquatting, not combosquatting.
(c): Combosquatting does not inherently involve typographic attacks or cross-site scripting.
(d): Injecting malicious JavaScript is not a defining characteristic of combosquatting.

25. Which of the following *violates* the same-origin policy?
    (a) An embedded CSS file from a third-party domain.
    (b) A JavaScript script reading a resource from another domain.
    (c) An image loaded from a different domain.
    (d) A cookie set by the server with the `HttpOnly` flag.

The same-origin policy restricts JavaScript from accessing resources on different origins (different protocol, domain, or port) to prevent cross-origin attacks.

(a): Embedding CSS from third-party domains does not violate the same-origin policy, as static resources like stylesheets are not restricted.
(c): Loading images from different domains is also allowed under the same-origin policy.
(d): The HttpOnly flag prevents client-side scripts from accessing cookies but is unrelated to the same-origin policy.

— The end —