Computer Security

2018 Exam 2 Review

Paul Krzyzanowski

Rutgers University

Spring 2018

Question 1

Explain why *hypervisor rootkits* are more difficult to detect than user-mode or kernel-mode rootkits.

The presence of a hypervisor is **invisible** to the operating system and any programs.

A hypervisor can **intercept** & monitor all I/O operations transparently

Every bit of code and data in the OS, libraries, applications, files, and memory can look untouched.

Not:
"Runs before the OS"

Question 2

Explain why it is not feasible to use hash pointers in a doubly-linked list structure.

It's a circular dependency problem.

If A→B, you want to store hash(B) in A.

But B contains B→A, which contains hash(A). So you need to recompute hash(A) and update B.

But that changes hash(B) in A, so you need to recompute that … but then hash(A) changes …

Not: "hashing is a one-way operation"

Nothing to do with circular lists or having to re-hash the entire list

Question 3

How does a *digital signature* differ from a Message Authentication Code?

Both are keyed hashes. You need to use a key to create them and to check them.

A MAC uses a shared key.

A digital signature uses a private key for signing and a public key for validation.

Not: Signatures are used to establish origin while MACs are used for message integrity.

Note: messages are not encrypted with either MACs or digital signatures

Question 4

What makes a *digital certificate* unforgeable?

What does it mean to *forge* a signature?

(a) Take someone else's certificate and change the public key in it so that you have the corresponding private key.

You cannot do this because the certificate is signed by the CA and you do not have the CA's private key, which you need to change the signature.

(b) Change your name (and other ID) to associate it with someone's public key.

You cannot do this because the certificate is signed by the CA.

(c) Create a totally new certificate where you impersonate someone else.

You can do this but only if the CA is not trustworthy. A legitimate CA will insist on validating your identity in some way.

Not: certificate chaining
Note: the certificate does not contain any private keys and nothing in the certificate is signed with your private key.

Question 5

Which confinement technology would work best to restrict an application to only be able to open text files?
(a) Capabilities.
(b) Containers.
(c) Sandboxes.
(d) Virtual machines.

(a) No. That limits root escalation operations.

(b) No. That's a combination of capabilities, control groups, and namespaces.

(c) Yes – checks can be done on parameters to system calls.

(d) No. Like containers, it allows broad partitioning of namespaces (put files in different systems) but not per-file validation

Question 6

Unlike a virtual machine, multiple *containers* on one system:
(a) Cannot see each other.
(b) Share the same system hardware.
(c) Share the same operating system.
(d) Share the same set of libraries.

(a) True, but that's just *like* a virtual machine

(b) True, but VMs share the same hardware too.

(c) Each VM runs its own copy of the operating system. Containers to not.

(d) Containers were designed to package dependencies, such as libraries and supporting applications, into an isolated container.

Question 7

Application *sandboxing* primarily works by:
(a) Running code via an interpreter.
(b) Creating a virtual machine for each application.
(c) Requiring user authentication prior to running an application.
(d) Intercepting system calls.

(a) No code is interpreted. System calls may be intercepted but code runs natively.

(b) No. That's not a property of the sandbox. Some, like Java, use a processor virtual machine, but that's because of Java and not because of the sandbox.

(c) No. There's no requirement that users need to authenticate themselves prior to running an app.

(d) Yes – system calls are examined to see if the requested operations conform to the rules of the sandbox.

Question 8

A problem with running a sandbox at the *user level*, as Janus does, is:
(a) Keeping state synchronized with the operating system can be challenging.
(b) Security controls are limited to what the operating system offers.
(c) It is easy for a process to bypass the sandbox.
(d) A system cannot support multiple sandboxes concurrently.

(a) Yes – the system needs to replicate the state of the operating system: failures, side-effects, asynchronous events.

(b) No. Sandboxes enforce their own additional constraints.

(c) Generally, no. It may be possible but that's a bug and not easy.

(d) Sure it can. There's no limitation on the # of processes running in sandboxes environments.

Question 9

A restriction with the Chromium Native Client (NaCl) is that applications:
(a) Run in an interpreted environment.
(b) Can only be written in JavaScript.
(c) Must interact with the system through NaCl libraries.
(d) Run within their own virtual machine.

(a) No. NaCl was designed to run native applications directly for maximum performance.

(b) No. NaCl has nothing to do with JavaScript.

(c) Yes. The libraries ensure that the apps don't interact with the operating system directly.

(d) No. There are no VMs here.

Question 10

The Java Virtual Machine (JVM):
(a) Uses a hypervisor.
(b) Virtualizes a processor architecture.
(c) Uses containers for isolation.
(d) Intercepts system calls made from Java programs.

(a) No. It simulates a processor that can execute Java code to run a single process.

(b) Yes. It simulates a Java Machine.

(c) No containers are used.

(d) Not really. System calls are not intercepted. They're inside classes that use native methods (e.g., compiled C code) to invoke system calls, bypassing the JVM.

Question 11

In contrast to a native virtual machine, a *hosted* VM:
(a) Is cloud-based instead of local.
(b) Can emulate a non-local processor architecture.
(c) Runs within a container.
(d) Directs requests to an installed operating system that is not running under a hypervisor.

(a) No – Clouds – or networks – have nothing to do with this.

(b) No – neither native nor hosted VMs emulate processors

(c) No – VMs don't run in containers.

(d) Yes. A hosted VM means there is an OS that is in charge of the underlying hardware rather than the hypervisor.

Question 12

A *virus* differs from a worm in that:
(a) It is malicious while a worm is benign.
(b) It exists as part of some other software rather than a separate process.
(c) It is local while a worm is delivered via a network.
(d) It is designed to propagate.

(a) Either one can be malicious or benign.

(b) Yes – a virus, by definition, is tied to some other software. A worm can propagate and run on its own.

(c) No.

(d) Viruses are also usually designed to propagate.

Question 13

A *zero-day* attack is:
(a) Based on a previously undisclosed vulnerability.
(b) An attack that takes place on the same day the software is released.
(c) An attack that takes place in the early morning hours when it is unlikely to be detected.
(d) A spontaneous attack that disappears quickly.

(a) Yes. It take advantage of a vulnerability that administrators, and usually authors, were not aware of and didn't fix or guard against.

Question 14

A *macro virus*:
(a) Takes advantage of scripting capabilities built into some programs.
(b) Combines a sequence of operations into one program.
(c) Refers to any attack that is deployed via social engineering.
(d) Targets the entire system, while a micro virus targets a single application.

(a) Yes. It's a virus that runs within the scripting capabilities of certain programs. Microsoft Word & Excel have been common targets since they support VB scripting. Text editors (e.g., vim, emacs) have also been targets.

Question 15

A *Trojan horse* is:
(a) Malware that appears to come from someone you know.
(b) Software that runs undetected on your system but creates a backdoor for attackers.
(c) Software that disguises itself as a legitimate service.
(d) Any horse breeds that originated from the Balkan Pony.

(a) It might but doesn't have to.

(b) Its purpose may be hidden but the software isn't. It doesn't have to create a backdoor; it may send data, for example.

(c) Yes, by definition. A trojan has an *overt* function and a *covert* function.

(d) The Balkans includes parts of Turkey where Troy was located but there's no such collection of horse breeds.

Question 16

Backdoors rely on:
(a) Social engineering.
(b) Malicious installation of hidden software.
(c) The ability to execute code from within documents (e.g., PDF files, Microsoft Office documents).
(d) Bypassing normal authentication checks.

(a) No. They are often built into software for honest purposes, such as software updates or acquiring diagnostics.

(b) See (a). They aren't necessarily tied to malware.

(c) No.

(d) Yes. They provide a "back door," a way to get into the software or system even without legitimate access.

Question 17

Spear phishing refers to:

(a) An attack that is directed to specific targets, using information customized to those targets.

(b) An online attack that appears to come from a legitimate organization.

(c) Any email that contains URLs to malicious sites.

(d) A fraudulent website that attempts to extract personal information from users.

(a) Spear phishing is a deception attack like phishing but uses customized personal information to convince the target that the message/site is legitimate.

(b) This applies to phishing too.

(c) It might contain URLs to malicious sites but this isn't a good definition.

(d) This applies to phishing too.

Question 18

A *virus signature* is:

(a) A hash of a virus used by virus checkers to identify a virus.

(b) A sequence of bytes that a virus checker believes is unique to a virus.

(c) An encrypted hash of a virus used by malware to ensure it is not modified by virus eradication software.

(d) A digital signature attached to software to enable detection of whether it has been modified by a virus.

Virus signatures have nothing to do with cryptographic signatures.

They're just a set of bytes in the virus that are used as a pattern to identify the malware.

Question 19

Kerckhoff's Principle tells us that:
(a) Symmetric ciphers are more secure than public key ciphers.
(b) Longer keys provide exponentially greater security.
(c) Ciphertext should be indistinguishable from random data.
(d) The encryption algorithm does not need to be secret.

Kerckhoff stated that the entire secrecy of an encryption has to be in the key.

Question 20

Suppose it takes you one hour to test all 4-byte keys. How long will it take you test all 5-byte keys?
(a) 1.25 hours.
(b) 2 hours.
(c) 8 hours.
(d) 256 hours.

Each bit doubles the search space.

1 extra byte = 8 extra bits => 2^8 = 256 times longer

Question 21

Which ciphers are not vulnerable to *frequency analysis attacks*?
(a) Monoalphabetic substitution ciphers.
(b) Transposition ciphers.
(c) Polyalphabetic substitution ciphers.
(d) Shift ciphers.

Transposition ciphers will have the same frequency distribution of characters in the ciphertext – they're just scrambled.

You'd need to look at the frequencies of digraphs, trigraphs, etc. to get insights.

Question 22

One-time pads, created in 1882, are impractical because:
(a) Key distribution is difficult.
(b) They are not as secure as newer algorithms, such as AES or ECC.
(c) They are computationally inefficient.
(d) They do not work with binary data.

(a) Yes – the key has to be as long as the message and cannot be reused. We replaced the problem of transmitting a message securely with the problem of transmitting an equally-long key securely.

(b) The one-time pad is the only provably secure cipher.

(c) They are insanely efficient: c[i] = p[i] ^ k[i]

(d) Sure they do – just do xors.

Question 23

A *Feistel cipher* differs from a normal block cipher because:
(a) It goes through several rounds of substitutions and permutations.
(b) Each round only permutes half of the data in the block.
(c) It does not require multiple rounds.
(d) It supports different keys for encryption and decryption.

(a) Normal block ciphers have rounds of S-P operations.

(b) Yes – the SP network only takes ½ the block per round

(c) Yes it does.

(d) No – it's a symmetric cipher.

Question 24

Cipher Block Chaining (CBC) is used to:
(a) Encrypt each block of data with a different key.
(b) Add a hash pointer to each successive block of cipher text.
(c) Allow a message stream to be encrypted with a series of encryptions for increased security.
(d) Make the ciphertext of one block a function of the ciphertext of the previous block.

(a) No. One key is used for the entire message stream.

(b) No hash pointers are used.

(c) Just one encryption per block takes place.

(d) Yes. A block of plaintext is XORed with the previous block of ciphertext prior to encryption.

Question 25

A *hybrid cryptosystem* uses:
(a) A double layer of encryption for extra security.
(b) Multiple encryptions to support multiple recipients.
(c) A combination of message encryption and digital signatures.
(d) A combination of message encryption and key exchange.

(a) No. Everything is encrypted just once.

(b) No. It has nothing do to with multiple recipients.

(c) No digital signatures are used.

(d) Yes. It uses public key cryptography to send a symmetric key and symmetric cryptography to encrypt data.

Question 26

To send a message securely to Alice, Bob would encrypt the message with:
(a) Alice's public key.
(b) Alice's private key.
(c) Bob's public key.
(d) Bob's private key.

Bob encrypts the message with Alice's public key since only Alice has the corresponding private key to decrypt the message.

Question 27

Collision resistance in a hash function means:
(a) It is difficult to find two messages that hash to the same value.
(b) Two different messages can never hash to the same value.
(c) The hash value is recomputed with different parameters if it is found to hash to the same value as another message.
(d) Each hash pointer is distinct from every other one in a system..

(a) There is no known way of finding two messages that hash to the same value.

(b) It's possible – pigeonhole principle $2^{256} << 2^{\text{megabytes}}$

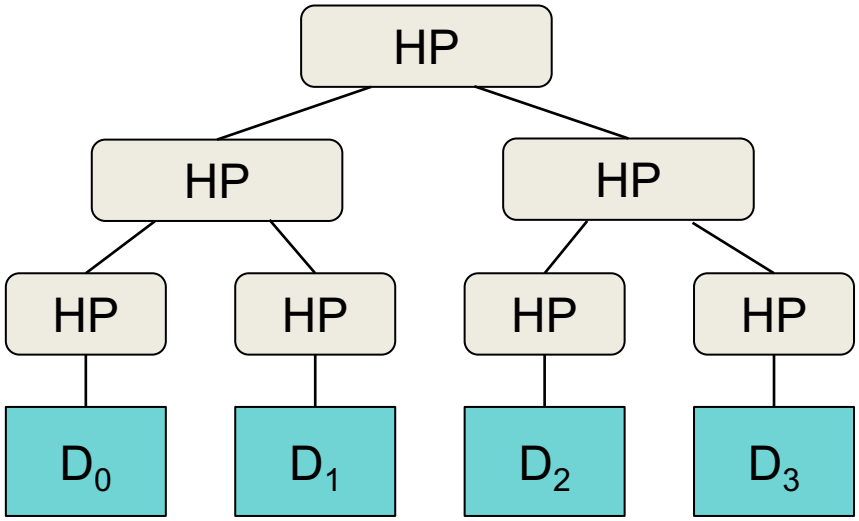
(c) Nope.

(d) Nothing to do with hash pointers.

Question 28

Suppose you have a Merkle tree with 32 data blocks (leaf nodes). How many hashes need to be recomputed to modify the data in one leaf node?

(a) 5
(b) 6
(c) 31
(d) 32



Suppose we have 4 blocks

Need 4 + 2 + 1 hash pointers
= 3 levels of pointers

In general,
$1 + 2 + 4 + \ldots + 2^n$ hash pointers

Depth = $\lceil \log_2(n) \rceil + 1$

For 32 blocks, we need
1 + 2 + 4 + 8 + 16 + 32 pointers

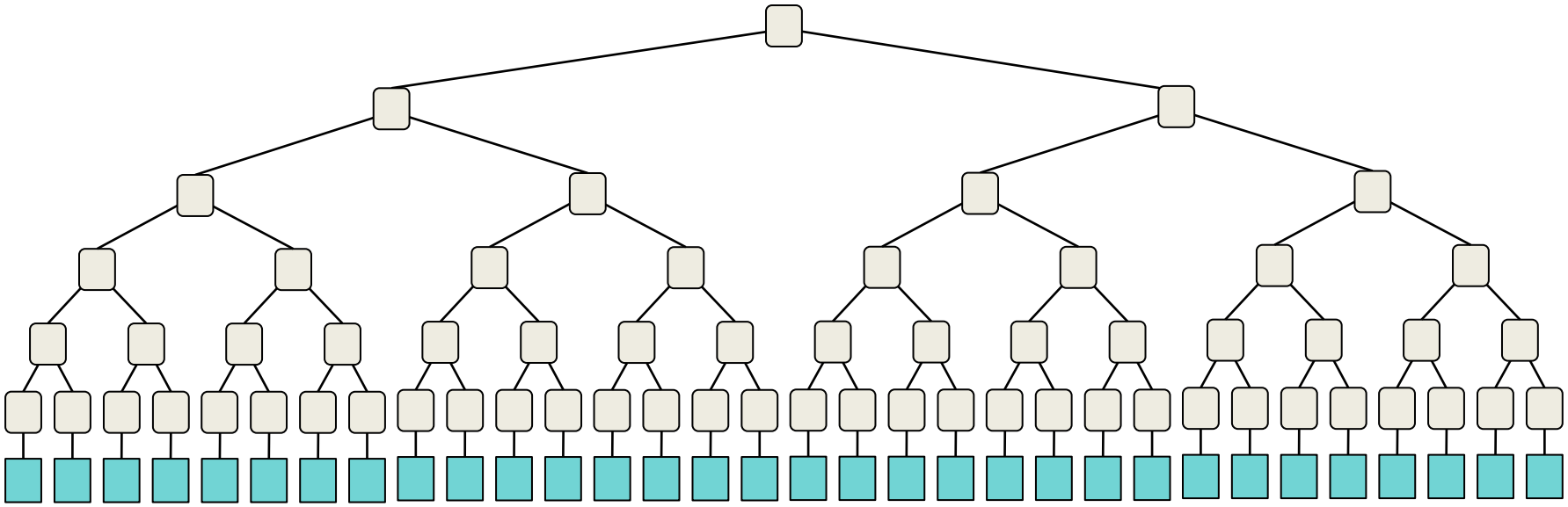
Depth = 6

Question 28

Suppose you have a Merkle tree with 32 data blocks (leaf nodes). How many hashes need to be recomputed to modify the data in one leaf node?

(a) 5
(b) 6
(c) 31
(d) 32

You received 2 out of 3 points for (a)



Question 29

Which of these were *not* a modification to the Needham-Schroeder protocol designed to help avoid replay attacks?
(a) Timestamps.
(b) Digital signatures.
(c) Use the of a trusted third party.
(d) Session IDs.

(a) Timestamps were added to allow a recipient to detect a replay attack.

(b) Digital signatures were never a part of Needham-Shroeder-based key exchange protocols.

(c) The original algorithm already used a trusted third party BUT the trusted third party has nothing to do with replay attacks.

(d) Session IDs were added as an alternative to timestamps.

Question 30

When Alice gets a Kerberos *ticket* to talk to Bob, it contains:
(a) Alice's public key.
(b) A shared session key.
(c) Signed authorization information from Kerberos.
(d) Bob's public key.

(a) Public key cryptography is not used in Kerberos

(b) Yes.

(c) The message is encrypted with a shared key but there is no signed authorization.

(d) See (a)

Question 31

Kerberos avoids *replay attacks* via the use of:
(a) Timestamps.
(b) Digital signatures.
(c) A trusted third party.
(d) Session IDs.

(a) Yes.

(b) No signatures or public key cryptography is used.

(c) Yes, but this doesn't help with replay attacks.

(d) No.

The end