# Quiz 2

| Question | Parts | Points |
|---|---|---|
| 1: Lab 2: Bad Randomness | 3 | 3 |
| 2: Architecting a Secure System | 2 | 2 |
| 3: Isolation | 2 | 2 |
| 4: Software Trust | 3 | 3 |
| 5: Lab 3: Timing Attack | 1 | 1 |
| 6: Lab 3: SSH | 1 | 1 |
| 7: Lab 3: Signatures | 2 | 2 |
| 8: CPU Timing Attacks | 2 | 2 |
| 9: Client Device Security | 2 | 2 |
| 10: Software Security | 3 | 3 |
| 11: Lab 4: Isolation | 2 | 2 |
| 12: Lab 4: WebAssembly | 4 | 4 |
| 13: Vulnerability Disclosure Programs | 2 | 2 |
| 14: Privilege Separation | 2 | 3 |
| 15: Runtime Defenses | 2 | 2 |
| 16: TLS | 2 | 2 |
| 17: Zero-knowledge proofs | 2 | 2 |
| Total: | | 38 |

Name: _____

- This is an open book exam: you can use your notes or any material released by us this term. You cannot use the internet.
- Any form of collaboration is *strictly* forbidden.
- If you find a question ambiguous, be sure to write down any assumptions you make.

**This quiz exam is printed double-sided!**

**Problem 1.** [3 points] **Lab 2: Bad Randomness** (3 parts)
Recall that in Problem 1 of Lab 2, you exploited a bug in how the 256-bit ECDSA keypair was generated. In `ecdsa/keygen.py`, the seed for signing and verification generations was computed by `SHA256(current time)`.

(a) [1 point] Why is this key generation scheme insecure? (Select all that apply)

- `SHA256` is deterministic
- An adversary could find a collision in the key generation process with non-negligible probability
- `current time` does not have sufficient entropy
- The generated public verification key reveals information about the signing key

(b) [1 point] Describe how this issue breaks the existential unforgeability (EUF-CMA) property of ECDSA.

(c) [1 point] How can you fix this key generation scheme to obtain 128-bit security? (Select all that apply)

- Use a hash function with larger output space
- Use a hash function with smaller output space
- Set the seed to be a random 64 bytes
- Randomly generate a 32 byte signing key

**Problem 2.** [2 points] **Architecting a Secure System** (2 parts)
Answer the following multiple choice questions:

(a) [1 point] Apps on a phone are isolated

- physically
- by the Operating System (OS) Kernel
- by the Virtual Machine Monitor (VMM) or hypervisor
- by the language runtime

(b) [1 point] Which of these are a nice feature of capabilities? Select the most appropriate answer below.

- They are easy to delegate
- They are precisely scoped
- Both of the above
- None of the above

**Problem 3.** [2 points] **Isolation** (2 parts)
Answer the following multiple choice questions:

(a) [1 point] Which of these properties or mechanisms protects the metadata of a process?

- Non-leakage
- Delegation
- Non-interference
- Scoping

(b) [1 point] Which of these is NOT an isolation mechanism?

- Naming
- Time-multiplexing
- Compilation
- Delegation

**Problem 4.** [3 points] **Software Trust** (3 parts)
Answer the following multiple choice questions:

(a) [1 point] Provide one **positive** feature of reproducible builds of binary executables from source code.

(b) [1 point] Provide one **negative** feature of reproducible builds of binary executables from source code.

(c) [1 point] (T/F) In a measured boot process, all software that is loaded needs to be signed by a trusted source verifiable by the boot ROM public key.

**Problem 5.** [1 point] **Lab 3: Timing Attack** (1 part)
Recall in Lab 3, a bad server ran the following code, which allowed us to extract the password through a timing attack.

```
s = request.password
for i in range(len(self._password)):
    if len(s) <= i:
        return api.VerifyResponse(False)
    elif s[i] != self._password[i]:
        return api.VerifyResponse(False)
```

Now imagine the server stores the hash of the password and runs the following code (differences: hash the password and check equality from the stored hash):

```
s = hash(request.password)
for i in range(len(self._password_hash)):
    if len(s) <= i:
        return api.VerifyResponse(False)
    elif s[i] != self._password_hash[i]:
        return api.VerifyResponse(False)
```

Are we susceptible to the same timing attack as before? Why?

**Problem 6.** [1 point] **Lab 3: SSH** (1 part)

Recall the attack from Lab 3 part 3b, which we tamper with a ssh packet to change a `ls ./files/*` to `rm -r /`. In this attack, we are observing all of the communication between the server and the client. Imagine the adversary now starts observing the communication between the server and the client after the connection has already been established. The client will send many commands to the server instead of only the `ls ./files/*`, but it is guaranteed that the client will send `ls ./files/*` after the adversary starts listening. However, the adversary does not know where in the stream of commands they start listening at. The communication will look like:

1) Client establishes connection to SSH server 2) Client starts sending commands 3) Adversary starts listening 4) Client continues to send an unknown number of commands 5) Client sends `ls ./files/*`

How can the adversary perform the attack from the lab in this case? Assume the adversary can handle both the client requests and server responses.

**Problem 7.** [2 points] **Lab 3: Signatures** (2 parts)
Imagine Bob is trying to infer what signing algorithm a server is using. Select the best answers using the information from Lab 3.

(a) [1 point] Looking at the key length only, which algorithm would be the most identifiable?

- RSA
- DSA
- ECDSA

(b) [1 point] If Bob can't identify the algorithm from the key length, what information should he look at next?

- The time it takes to send a signature over the network
- The signature signing and verification times
- The signature length

**Problem 8.** [2 points] **CPU Timing Attacks** (2 parts)

(a) [1 point] Running a cache timing attack on a fully associative cache is _____ on a direct-mapped cache.

- more difficult than
- less difficult than
- the same difficulty as

(b) [1 point] (T/F) The Linux dm-crypt attack would still work if the cache were large enough to fit the entire AES key schedule.

**Problem 9.** [2 points] **Client Device Security** (2 parts)

(a) [1 point] (T/F) Downgrade prevention on the Apple iPhone requires that the ECID of each device be kept secret from the attacker.

(b) [1 point] (T/F) Data protection on the Apple iPhone requires that the UID of each device be kept secret from the attacker.

**Problem 10.** [3 points] **Software Security** (3 parts)

(a) [1 point] Consider the following C code:

```
void f() {
    char buf[128];
    gets(buf);
}
```

Which of the following statements is the most correct?

- If the stack grows in opposite direction of the overflow, then the adversary can overwrite the return PC of `gets()`
- If the stack grows in the same direction of the overflow, then the adversary can overwrite the return PC of `f()`
- Neither of the above is true
- They are both true

(b) [1 point] (T/F) The stack canary is a random value determined when the program is compiled into an executable.

(c) [1 point] Android apps shipped with zip files were corruptible because

- it was easy to find collisions in the hash scheme used prior to computing the signature
- Only a subset of the data in each file was checked
- non-unique file names resulted in signatures being computed and checked on different files
- the Android zip utility was susceptible to a buffer overflow attack

**Problem 11.** [2 points] **Lab 4: Isolation** (2 parts)
In Problem 1 from Lab 4, you likely took advantage of the garbage collection system to deploy your attack – no worries if not, the question provides all the background you need.

A garbage collector is a memory management tool that automatically reclaims memory occupied by objects no longer in use, preventing memory leaks. It tracks object references, identifies those that are unreachable, and deallocates their memory, freeing developers from manual memory management tasks.

The attack involved using `gc.inspect()` to see details currently tracked by the garbage collector, then finding the secret from the objects tracked by the garbage collector.

(a) [1 point] (Y/N) If the garbage collector deallocates memory as soon as possible, would this attack still work?

(b) [1 point] Why do garbage collectors not deallocate memory as soon as possible?

- To allow the system to allocate more memory for new objects
- To minimize performance overhead
- Because it cannot detect unreferenced objects reliably
- To optimize memory usage in specific regions

**Problem 12.** [4 points] **Lab 4: WebAssembly** (4 parts)
You also had the opportunity to work with WebAssembly in the lab. In this question, we will look at why exactly WebAssembly is such a big deal and how it revolutionized web applications.

(a) [1 point] What is WebAssembly?

- A framework for building server-side applications
- A binary instruction format for running code on the web with near-native performance
- A JavaScript library for improving browser rendering speed
- A security protocol for encrypting web communication

(b) [1 point] How does WebAssembly improve security in web applications?

- It provides a framework for encrypting browser data
- It ensures all code is written in a type-safe language
- It prevents unauthorized access of the network
- It executes in a sandboxed environment, isolating it from the host system

(c) [1 point] Which languages can be compiled into WebAssembly?

- Only Javascript, Typescript, and Python
- Only markup languages like HTML and CSS
- Any language that supports the LLVM compiler framework, such as C or C++
- None of the above

(d) [1 point] What is the **main** advantage of WebAssembly (i.e. why do so many people use it)?

- It provides enhanced security through a sandboxed execution environment.
- It allows execution of precompiled code with near-native performance
- It simplifies web design by replacing JavaScript
- It runs independently of any operating system or browser

**Problem 13.** [2 points] **Vulnerability Disclosure Programs** (2 parts)
This question relates to the presentation from the BU students about VDPs.

(a) [1 point] Bob is performing vulnerability research on an online programming website, SandSquare, which allows people to write and run arbitrary code on the web. Bob knows that for this to be secure, the developers must implement a sandbox. Bob discovers, however, that the developers tried to enforce isolation through python (think about Lab 4). Bob extracts all of the secrets running in the main program. What law is Bob breaking?

- Electronic Communications Protection Act
- Computer Fraud and Abuse Act
- Unauthorized Computer Access Act
- Computer and Internet Hacking Act

(b) [1 point] Bob realized he forgot to check the VDP to see if he was able to perform this research. What provision should Bob be looking for to see if he is able to perform vulnerability research?

- Ethical Hacking Provision
- Get Out of Jail Free Provision
- Security Research Provision
- Safe Harbor Provision

**Problem 14.** [3 points] **Privilege Separation** (2 parts)

(a) [1 point] Which of these is a reasonable set of APIs for the App $\rightarrow$ Logger, Logger $\rightarrow$ DB, and App $\rightarrow$ DB? $A \rightarrow B$ means that $A$ is calling a function running inside $B$. App is the application, Logger is the logging utility, and DB is the database. Read and Write below are random access.

- Read, Write, Read, Write, Read, Write
- Read, Append, Read, Write, Read, Append
- Read, Append, Append, Delete, Read
- Read, Write, Append, Read

(b) [2 points] Which of these is a reasonable set of APIs for Launcher $\rightarrow$ Network, Launcher $\rightarrow$ Time Service, and Network $\rightarrow$ Time Service? Launcher is the main program of the Network Time Protocol service running on the system, Network provides the current time according to some server on the Internet, and Time Service controls the time for the system. $A \rightarrow B$ means that $A$ is calling a function running inside $B$. Resetting a clock sets the time of day to be 0 without changing the day, setting and shifting the clock each take an argument.

- Launch, Shift Clock, Set Clock
- Launch, Set Clock, Reset Clock
- Launch, Launch, Shift Clock
- Launch, Reset Clock, Set Clock

**Problem 15.** [2 points] **Runtime Defenses** (2 parts)

(a) [1 point] (T/F) A non-executable stack requires compiler support to restrict addresses of data regions from being jump targets.

(b) [1 point] ASLR works at the level of

- Physical memory
- CPU caches
- Virtual memory
- CPU registers

**Problem 16.** [2 points] **TLS** (2 parts)
Answer the following multiple choice questions:

(a) [1 point] In TLS, which of the steps uses only public-key cryptography?

- Handshake
- Record Protocol

(b) [1 point] In TLS, how do the parties agree on a secret key without ever meeting?

- They run a key exchange protocol.
- They use a trusted party to generate a secret key for them.
- They don't exchange a secret key, instead only agreeing on public-key cryptography.

**Problem 17.** [2 points] **Zero-knowledge proofs** (2 parts)
Answer the following multiple choice questions:

(a) [1 point] A common way to convert a 3-round identification scheme into a signature scheme is by:

- Send the verifier's message ahead of time
- Set the verifier's message to be the hash of the prover's first message and the message to be signed
- No need to reduce interaction, since ZK proofs are non-interactive to begin with

(b) [1 point] Zero-knowledge proofs-of-knowledge guarantee which one of the following?

- If the verifier is convinced it must learn the witness.
- If a (possibly cheating) prover succeeds in convincing the verifier then the prover must know a witness.