Homework 4

▼ From the "Android Permissions Re-mystified: A Field Study on Contextual Integrity" paper:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/389bf1ac-bd13-4bef-aab2-3f11d0011abc/sec15-paper-wijesekera.pdf

▼ What is "contextual integrity?"

- Ensuring the information flows are appropriate as determined by user and how often applications access protected resources when the users are not expecting it.

▼ What kind of method have the authors tried to reduce the frequency of runtime permission requests?

- First the authors tried to prompt when only protected resources are accessed, but it still had too many requests. Then they tried to model the decision based on context. Then they tried prompt on first use, initially without visibility and then with visibility.

▼ From the "Preventing Privilege Escalation" paper:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/96f2b924-d05c-4b6a-846a-f3b5f0620300/provos_et_al.pdf

▼ When performing "privilege separation," what principle should be followed? In other words, if we treat this as an optimization problem, what should be maximized/minimized under what constraints?

- The code that requires privilege should be separate from the code that does not require privilege. If it were an optimization problem, code that requires privilege should be minimized and separated under the constraint that it can be done by the master that has all the privileges required.

▼ What kind of privileges are required by the "unprivileged child?"

- By default the unprivileged child does not require any privilege and is just a slave. However in the post authentication phase, it can have privileges to access file system. However, for special privileges, it still needs to request from privileged parent of master. Also, a child can ask different privileged operations from monitor.

▼ From the "Native Client: A Sandbox for Portable, Untrusted x86 Native Code" paper:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/da00012c-6105-4b45-8e49-740f00c57cbc/34913.pdf

▼ How does native client enforce data integrity (i.e., constrain data access for code inside the inner sandbox)?

- The Native client combines the previous work of CISC fault isolation with 0386 segmented memory and constrain data references to a contiguous subrange of virtual 32-bit address space. This allows for an effective sandbox without sandboxing of load and store instructions.

▼ How does native client enforce control-flow integrity?

- For control flow integrity, it is required to insure that all control transfers in the program text target an instruction identified during disassembly. For this, For each direct branch, target is computed statically and it is confirmed that it is a valid instruction as per constraint C6 (All valid instruction addresses are reachable by a fall though disassembly that starts at the base address). For indirect branches, the 80386 segmented memory with simplified sandboxing are combined. As per constraint C2 and C4, the CS segment is used to constrain executable test to a zero-based address range sized to a multiple of 4k bytes.

▼ From the "Kerberos: An Authentication Service for Open Network Systems" paper:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/c1e2befb-bde2-4acc-baaf-7e05aa9d82ed/kerberos.pdf

▼ What is the threat model (i.e., what attackers can do and what kind of attacks this work aims to mitigate)?

- In an open environment, the user and host both can be used as attacker and both can be the victim. Thus, both of them need to authenticate and prove their identity. The attackers can impersonate a real user, forge some credentials to appear to be real user or replay the stolen messages and authenticated certificates to appear to be the real user in order to gain access. The paper aims to mitigate this by introducing Kerberos, a third party authentication service, which along with a ticketing service, will be able to prevent such attacks and validate authentic users.

▼ Authentication can be done based on what you know (e.g., password) and what you have (e.g., a secret), explain how a server authenticate the client, and how the client authenticate the server.

- When a user requests a service, their identity must be established. This is done by presenting a ticket to the server. along with the proof that the ticket is from the user and not stolen. The user obtains credentials to be used to be able to request access to a service. User then requests authentication for the service. User then presents the credentials to the end server. If a user wants to authenticate the server as well, this can be done by the user specifying that it needs to authenticate the server. The server then adds 1 to the timestamp received by it from the user, encrypts it in session key and sends it back to the user. With this process, the user is sure the server is authentic and the server is sure that user is authentic. They also have a session key that only they have for communication.

▼ From the "VC3: Trustworthy Data Analytics in the Cloud" paper:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/3c926201-ef25-41c1-8209-eb281214abf4/vc3-MSR-TR-2014-39.pdf

▼ 1. Section 5 describes a technique called "self-integrity." What kind of security threats does self-integrity aims to mitigate?

- Read outside enclave
- Write to null pointer or outside enclave
- Hijacking of control flow by code outside of enclave

▼ 1. Section 6.2 describes the key exchange protocol of VC3, where a secret key \`k_w\` is exchanged. Explain why only the client and the enclave knows \`k_w\`, and the untrusted cloud provider (including administrators, OS, and the hypervisor) cannot know \`k_w\`.

- The untrusted cloud can duplicate, drop, manipulate or leak the data. This is why they cannot know the secret key.