CS 446/646 – Principles of Operating Systems

Homework 6

Due date: Tuesday, 12/10/2024, 11:59 pm

Objectives: You will read up on and research famous cases that involve issues of ethical nature with respect to software engineering and more specifically Operating Systems. You will answer a set of open-format questions with your own opinions, analyses, and assessments, after researching this content as well as additional content from external resources.

General Instructions & Hints:

- All work should be your own.
- You may work on any preferred text editor (LaTeX, Word, LibreOffice Writer, GoogleDocs) and submit a commonly formatted document file (.pdf, .doc/.docx, .odt, etc.).
- To turn in, create a folder named **`PA6_Lastname_Firstname`** and store your text document files in it. Then compress the folder into a **`.zip`** or **`.tar.gz`** file with the same filename as the folder, and submit that on WebCampus.

Background:

I. The ACM Code of Ethics and Professional Conduct

The Association for Computing Machinery (ACM) has compiled a document expressing what conscientiousness is defined as in the computing professional context. This is known as the "*Code of Ethics and Professional Conduct*", and publicly available online at https://www.acm.org/code-of-ethics.

The corresponding booklet version is uploaded on the Webcampus Homework folder. You are expected to have read the 4 major sections covering the subjects of:

1. General Ethical Principles
2. Professional Responsibilities
3. Professional Leadership Principles
4. Compliance with the *Code*

before moving on to the subsequent sections of this assignment.

II. OS Security and Impacts on Performance / Economy

Of the known Common Vulnerabilities and Exposures (CVE)s, an up-to-date list of which can be found online at https://www.cisa.gov/known-exploited-vulnerabilities-catalog, two caught the world by surprise in early 2018. These were named *Meltdown* (CVE-2017-5754 : https://en.wikipedia.org/wiki/Meltdown_(security_vulnerability)) and *Spectre* (CVE-2017-5753, CVE-2017-5715 - https://en.wikipedia.org/wiki/Spectre_(security_vulnerability)), and they both stemmed from the very microarchitectural design of most popular processors at the time (including Intel, AMD, ARM-based, and IBM). The ability to perform branch prediction and speculative execution (https://en.wikipedia.org/wiki/Speculative_execution) integrated within these architectures managed to yield increased execution performance, but due to leaving observable side-effects (leftover data caches of branch mispredictions could be extracted) constituted a **significant security risk** that affected almost every single modern processor at the time.

As soon as patches were available, Data Centers and Cloud Service Providers had to be restarted in order to apply them, with potential interruptions in large-scale provided services. Even worse, the patches necessarily were at the level of processor microcode and OS kernel functionalities, and to

mitigate the security flaws **a significant impact on performance had to be incurred**. This meant impacting clients' tight-timing-critical operations (think Financial Service sector & High-Frequency Trading systems). Even more, updated patches had to be rolled-out through OS-level updates even years after (Sept 2020, Nov 2020, Jan 2021) by the OS vendors/maintainers, effectively creating a cycle of the same situation for clients. And this is just one of the most famous instances.

III. OS Security and Impacts on Human / Environmental Safety

A historically-infamous malicious attack is Stuxnet (https://en.wikipedia.org/wiki/Stuxnet), which was effectively a rootkit targeting the MS Windows operating system. Its most potentially catastrophic impact was the disruption of the centrifuges performing the Uranium enrichment process in Iran; in this case, the Stuxnet variant was designed to use the OS as the vector to infect industrial Programmable Logic Controllers (PLC)s through the PLC management software that was running on the OS (Siemens Step7). The infected devices and OS-running software would use incorrect commands while misreporting correct operation; the impact of this was a brief disruption of the enrichment process, but in that period of time a serious nuclear accident occurred involving the shutdown of the centrifuges.

Nuclear accidents being the threat they are for human and environmental safety locally but also globally, this is one of the most world-renowned instances that an OS zero-day flaw was involved.

IV. OS Security and Legal Questions

Another historically significant case was the Apple – FBI encryption-related legal dispute of 2016 (https://en.wikipedia.org/wiki/Apple%E2%80%93FBI_encryption_dispute), wherein the company was asked to facilitate the extraction of encrypted information from iPhone devices that were running their proprietary iOS7 version Operating System. The cellphones were company-issued devices to individuals who had conducted terrorist attacks.

The proprietary nature of the device's software meant that the company alone had the ability to unlock such access at the time; Apple declined the request due to its policy against undermining their own security features. This led to long litigation and more importantly public visibility for the case. Eventually a "third-party" (non-concretely identified to this day) performed this task, also using a zero-day flaw in the OS. Interestingly, later iOS versions used new encryption methods to ensure that not even the company could facilitate such government requests.

While this case left the public eye, a number of questions remain as to what the OS flaw was and/or the nature of the "third-party" used. The very proprietary nature of this core-and-crucial piece of software provides little transparency to potential back doors / vulnerabilities that existed at the time, and/or have carried on to today.

V. OS Development Process

A last prominent case that made the headlines in 2018 was the reveal that Linux Torvalds, the "mastermind" behind Linux, one of the most globally-successful Operating Systems, had a long history of exhibiting insulting behaviors against contributors to the open-source kernel's development, especially when proposing problematic changes with potential impacts that could "break the code". This led to significant public exposure, and Linus himself temporarily stepped down, announcing that he was going to focus on working to improve his ability to appropriately respond to people within such contexts. Linus, who has been named a "benevolent dictator", remains in a leading role until this day.

Intricate (and critical) software such as Operating Systems, require a deep understanding of multiple levels of functionalities and their interplay (https://en.wikipedia.org/wiki/Linux_kernel - Map of

the Linux kernel), necessarily imposing some form of "gate-keeping" for decisions spanning from the micro-scale, to the integration of entire new feature support. Behaviors such as colleague/worker abuse are and should remain inexcusable on one hand. At the same time, open-source development implies that necessarily, strict control over development teams and their active members is challenging if not impossible.

Centralized decision-making at some level is required, and can take place from the early design phase (prescribing specific tasks and methods to use for desired new features), or can be in the form of "gate-keeping" that controls which development efforts get integrated, and which ones get trashed (irrespective of the time dedicated). In this boarder context, another important notion to consider, especially for OS that are such critical pieces of software, is consistency when leadership change necessarily happens.

General Directions:
Read the *ACM Code of Ethics and Professional Conduct*, and research the above-mentioned cases that involve issues of **ethical nature**. You are also invited to further expand your conceptual understanding of the widespread ethics-related questions associated to software engineering, and more specifically for critical software such as Operating Systems, by researching online more cases with similar profiles. Then, answer the following questions that are related with the 4 aforementioned cases, with **your own**, open-form discussion and analysis. For cases, online articles, etc. that you wish to reference, you have to include **citations** at the end of your document.

1) What is the responsibility of an OS vendor/distributor in the cases of where significant hardware flaws are discovered. What if their product is especially affected when compared to other competitors? What role do you believe an OS software engineer (OS developer/employee/manager) has to play in such a circumstance?

2) What is the responsibility of an OS vendor/distributor when their platform is exploited (unspecified whether knowingly or not) to launch industry attacks that may geopardize more than just financial assets? What is the role of the OS engineer?

3) What is the responsibility to an OS vendor/distributor with respect to safeguarding privacy? If your answer is situation-specific, elaborate. What is the role of the OS engineer? What concerns do you have regarding whether the OS and its safety features are proprietary or open-source?

4) What do you believe are the ethics-related questions and principles that apply to an OS-development engineer, from the single-feature development roles, all the way to high-level management? Describe a form of integration between these levels that you believe is long-term sustainable.

Note: There is no word count / size restriction for each of the 4 questions. Your answers will however be assessed by whether they demonstrates that you are accounting for the previously discussed cases (or your own researched ones), as well as whether ethical principles discussed within the *ACM Code of Ethics and Professional Conduct* are taken into account (i.e. make sure they are not chatbot-generated blanket statements without specific nuance).

Submission Directions:
When you are done, create a directory named **`PA6_Lastname_Firstname`**, place text document(s) providing your answers to the above questions into it, and then compress it into a **`.zip`** or **`.tar.gz`** with the same filename as the folder.
Upload the archive (compressed) file on Webcampus.

Early/Late Submission: You can submit as many times as you would like between now and the due date.
A project submission is "late" if any of the submitted files are time-stamped after the due date and time. Projects that are up-to 24 hours late will receive a 15% penalty, ones that are up-to 48 hours late will receive a 35% penalty, ones that are up-to 72 hours late will receive a 60% penalty, and anything turned in 72 hrs after the due date will receive a 0.

Verify your Work:

After you upload your archive file, re-download it from WebCampus. Extract it, and re-read your document(s) to verify your latest submission is as intended.