**EMCS2600: The Future of Cybersecurity: Technology and Policy**

Assignment: Module 4 Review Questions

Brian Russel Davis, [brian\_davis@brown.edu](mailto:brian_davis@brown.edu)

Question 1: Choose what you consider to be the 3 most interesting Hardware Security issues from the readings and explain your decisions (in at most 200 words).

The most interesting problems are always the ones related to **how the hardware handles memory** in my opinion because there is not that much collaboration between firmware engineers and software engineers to catch the issues. In on the papers by Wei Hu, Baolei Mao, Jason Oberg, and Ryan Kastner, IEEE Computer, 2016, there was a discussion about how the hardware handles plain text data and how even at the design stage a malicious actor can use the abstracted plans for the hardware to expose a vulnerability. Allowing for a sub-second lag in a protocol at this stage could be enough for an attacker to compromise all of the plain text on a machine with no course for repudiation.

This falls in line with **supply chain issues**. How can anyone be sure that what is manufactured in China has not been modified or re-engineering with alternative components that also allow for a sub-second lag when processing plain text? These types of security issues cut to the center of a serious problem which is trust. How can we create centers of trust when we are doing business with countries that do not share our values? What type of validation would we have to put in place to ensure that hardware has not been tampered with or altered, and how much would that cost. It seems like device manufacturers have opted for the cheaper cost-efficient route at the cost of security.

And last but certainly not least, **computer memory** seems to be a huge target these days. The ALSR attacks mentioned in the readings, that I first witnessed in real-time at BlackHat and Defcon two years ago are just the beginning. This year more interesting methods like Stack Bombing were all the rave as researchers hijack the pieces of code that are known for Return Object Programming ( ROP ) after a crash. For example[[1]](#footnote-0) :

**Naïve code runs and crashes:**

HANDLE t = OpenThread(THREAD\_SET\_CONTEXT | THREAD\_GET\_CONTEXT |

THREAD\_SUSPEND\_RESUME, FALSE, thread\_id);

SuspendThread(t);

CONTEXT ctx;

ctx.ContextFlags = CONTEXT\_ALL;

GetThreadContext(t, &ctx);

DWORD64 ROP\_chain = (DWORD64)ctx.Rsp; // for the 5 alertable state functions…

… // Adjust ROP\_chain based on ctx.rip (or use APC…)

… // write ROP chain to ROP\_chain memory address in the target process

ResumeThread(t); // when the current function returns, it’ll execute the ROP chain

But it’s unclear to me if these types of attacks are a vector of the hardware/firmware or one originating in the way memory is handled by the OS?

Question 2: From the readings on Software Security pick two of the four categories that you consider most important and provide a synopsis of them (in at most 200 words).

In earlier papers I railed against RBAC because of the sheer number of CVE vulnerabilities that exist because of it’s poor implementation or misconfiguration. Defining a role with “built in” privileges then trying to hide the configuration is like placing the crown jewels in the middle of Times Square under glass with a sign that says “Do Not Touch.” While access control seems to be necessary in every system, it also seems like we should be looking for new and better ways to define who has access with creation of a roadmap for attackers. In my work with NASA I described a system where Discretionary Access would be determined in real tme by AI based on deep curation of personnel. In that work since the relations between co-workers may be new with sparse data I made use of Amazon’s Deep Scalable Sparse Tensor Network Engine to map the relationships and grant access based on functional curation instead static ( hackable ) xml files sitting in active directory.

Lastly I would like talk briefly about Configuration Errors. While it’s commonly accepted that dev op engineers are blamed for configuration error, much in the same way that we would blame parents for not following the direction when building a crib ( then the crib falls apart and harms the baby ). I would argue that safe / secure software configuration should be the job software product managers. If the software doesn’t come out of the box with a safe configuration, like if you bought a 9mm Glock and it was loaded with bullets, that should be the fault of the people that made it. Yes, testing and holistic configuration management will find errors in the configuration, but the software should some with the things that are the most dangerous turn off. For example, serves and other types of networking devices that come with port 32 open, should really be boycotted because who is really using telnet these days ( except for hackers )? Testing is a big part of security and development ( I have never written a piece of code without a unit test ), but testing should be a second layer defense. The first layer of defense is proper off the shelf configuration before the software is sold.

1. Windows Process Injection in 2019, Amit Klein, Itzik Kotler, Safebreach Labs : <https://i.blackhat.com/USA-19/Thursday/us-19-Kotler-Process-Injection-Techniques-Gotta-Catch-Them-All-wp.pdf> [↑](#footnote-ref-0)