**EMCS2010: Applied Cryptography and Data Privacy**

*Assignment: Cybersecurity Plan Crypto Chapter*

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

Version 1.0.0

2019-09-08T19:37:31+00:00

#### Section One

##### Scenario

PAX as an organization handles sensitive user data on a daily basis. While this data is extremely sensitive it is also extremely useful to the data science team and consequently to the software and product teams as they work to educate consumers on their consumption and provide relevant recommendations.

##### Security Objectives

Our objective is to create a secure and yet flexible system that protects the individual and collective privacy of the consumer while maximizing portability, data integrity and providing accurate and clean data points for our software engineers.

##### Scope

The scope for this solution includes all of the PAX device data, PAX pod data and usage information collected from consumers. This data includes but is not limited to usage information, pod strain and manufacturer information, PAX user account information and PAX user feedback.

##### Technical Description

Protecting user data is paramount, but can be incredibly tricky due to its depth, breadth, and complexity. Often software engineers require root access to the machines they are developing on, which if compromised, could result in their credentials being used to decrypt and exfiltrate data.

Our solution creates a network of encrypted Docker images that expose pseudo-anonymized data from a cache that is recycled and re-encrypted every 24 hours. The cache is protected by 265-bit encryption and rebuilt inside of a secure enclave by using homomorphically encrypted functions in “sister” Docker image.

The process starts with two blank sister Docker images that pair by scanning each other’s hard drives and building encrypted channels for communication. One docker image becomes a “bucket” and the other become a “scoop.” The bucket accepts a stream of data that ends with a unique blockchain hash. The stream of data starts when the image authenticates with the application controllers. Immediately the image pseudo anonymizes and shreds the data. The stream begins and ends with a block-chain inspired stamp that will match with a stream from the same user, even though the raw data from the user has been destroyed. The sister image using a homorphically encrypted API layer that was built during pairing then scoops the data into a central repository, aligning it with previous data by matching the blockchain stamps.

This system was inspired by the way the human body ingests breaks down the nutrients in food. The mouth, teeth and throat control general access but don’t have the codes to “break down” and extract the nutrients from food. In our solution we separate the control, access and algoritmic work between virtual machines with a very short lifespan to create realtime data with the risk that any one machine or system has access to a complete record.

##### Troubleshooting

There is a risk that records may be incomplete if one of the images fails. This is why the images should not create the blockchain stamp for the streamed data block until after the stream is complete. A new image would simply revive the stream after “consuming” the stalled image and get the next block of data, retaining the correct stamp even though usage data might be missing.

##### Exceptions

This system should not be used in systems that require the data to be retained or saved for the purpose of editing at a later date. This system destroys all identifying information after it comes in contact with the recieving image.

###### Glossary

**Docker** : is a set of platform-as-a-service (PaaS) products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating-system kernel and are thus more lightweight than virtual machines

#### Section Two

##### Rationale

There are many things wrong with the way we implement security, however the greatest issue that not very many organizations can overcome is the inheritant issues that invovle the way Operating Systems and Programming Langauge interact with the kernel and allocate memory. Trying to replace the current OS, kernel and compiler system is a not a small feat. Rather than trying to overcome the issues by making a bigger and better system, by making the containers that applications run in emphermal, small and limited we can limit the risk they pose to the system. The human body is not reliant on any one red blood cell to keep us healthly. And while one might argue there are organs whose failure would consequently be fatal, the human body experiences 100s of thousands of invaders every day. It requires a programatic and systemic failure to topple the body’s defenses, not just one instance of code injection or credential stuffing. By using strong encryption, then separating the scope of access among temporal processes, we effectively remove a surface for an attacker to gain a foothold in the system.