Security Goals

Overview

The purpose of this document is to outline security goals for the entire system

Goals

There are multiple goals that need to be met to provide users with the most secure platform as possible. We will be defending against many common attacks such as:

- Session hijacking and forging
- Database leaks
- Attackers gain knowledge of the internal system
- Attackers gaining access to the internal system and stealing data

End to End Encryption

All data must be encrypted with some sort of encryption while in transport. The microservices must only accept data from HTTPS sources. Data that is going from the microservice to the database must be encrypted with TLS and data that is at rest or not in use will be encrypted. Messages from one microservice to another will also be encrypted with TLS.

Verifiable Data

Certain data should be signed by the user so other users can verify it was them. Some examples include messages, files, commits, ect.

Authentic Microservices

We should be able to verify microservices before they start that it was signed by a trusted user. Moreover, users should be able to see who has signed the microservice. This problem can be solved by using PGP keys. See the infrastructure page for more information on how it is set up

Cryptic but Useful Errors

Having general errors displayed to the user allows for people to see what went wrong and prevents bad actors from gaining inside knowledge about the system. For example, if a file can not be found we would not display:

```
Traceback (most recent call last):
File "C:\Users\sauris\Desktop\pyfile\folder-read.py", line 13, in <module>
File "C:\Users\sauris\AppData\Local\Programs\Python\Python36-32\lib\bz2.py", l
ine 96, in __init__
self._fp = _builtin_open(filename, mode)
FileNotFoundError: [Errno 2] No such file or directory: 'sampledatafile2.txt.bz2'
```

Instead, we would just display:

```
sampledatafile2.txt.bz2 was not found
```

Non-root Containers

All container/microservices should not be run as the root user. This is to prevent someone from getting access to the container and compromising it.

Post Quantum Encryption

Recently NIST published the chosen algorithms for post quantum cryptography! These will be worked in to the software as seen fit

---

Revision #8
Created 2022-01-23 19:00:39 UTC by Kyle Heestand
Updated 2022-11-27 19:47:52 UTC by Kyle Heestand