Case of Study: Access Control Executive

Secure Design and Programming

1st Jorge Blanco Prieto
Master's Degree in Cibersecurity Research
University of Leon
Leon, Spain
jblanp03@estudiantes.unileon.es

***Abstract*—This paper shows the case of Access Control Executive, a software system categorized as a well-performed mental model. It shows the importance of good design in the architecture phase and how this design tries to obtain a list of several predefined objectives based on establishing temporary barriers for attackers. In addition, it is shown how software designs can be metaphors for real-world examples to leverage their architectural advantages.**

***Index Terms*—ACE, architecture, design, security, service, access-control, metaphors**

I. Overview

Many systems and applications have been compromised as a result of a wide variety of attacks [1]. In the growth of computer security in the mid-1980s, the software system Access Control Executive (ACE) was created. It provided key security services to various museums in Europe. The purpose of this system is running as a background service, control all resources belonging to the system and ruling on whetheher the action was to be permitted, denied or modified as a real time system.

The architecture defined in ACE allows that in case of attack, the system collapses in stages, appearing to be compromising the internal resources of the system but in reality setting up the defenses against the attacker. This is a metaphor for a real world example: Mayan temples. These temples are built on wide walls and rubble-filled large cavities which in the case of strong earthquakes, the kinetic energy of this internal ruins allows agitation to dissipate and minimices potential construction damage.

For the protection of critical system resources, the ACE software predefines security targets for attacks. The following table shows the requirements:

Table 1: Table to show predefined times under attack cases.

Case of Attack	Time
Attack by a well-informed expert malefactor	1 hour
Determined and computer-savvy attacker from the community of authorized users	1 day
Generally aimless or brute-force attack by a casual user	1 week

Design characteristics

By design, ACE software is comprised in a system of several downstream applications. Each application belonging to the system must request permission to execute a routine at its initialization. This architecture allows each application to be independent within the system, even though the permissions are dependent on ACE.

ACE was called capability cube since part of the system specification was based on customer controlling actions based on five request parameters:

- Job category
- Time
- Day of the week
- Physical location
- Application used to make the request

In the development, these parameters are treated as a five-dimensional matrix to make the model scalable, minimize computational cost and disk space.

A different architecture was considered proposing to build applications themselves with a modified version of the library itself, thus avoiding any change in the application source code. This idea was rejected because of the future concern of code maintenance, forcing to follow specialized techniques for all applications developed dowstream of the Acces Control Executive.

II. Conclusions

A. Design Metaphor

In the world of software development, within many of its fields (cybersecurity, artificial intelligence, optimization algorithms, etc.), there are a multitude of metaphors that try to replicate these architectures using real-world examples. Many systems have even been designed from real-world architectures seeking to create their same characteristics. This method is used to abstract the components and their interactions with each other [2]. The ACE design clearly shows how a metaphorical design can be brought into the software world.

B. Authorization

This article shows the case of a system that meets its objectives given a good definition of its architecture phase.

In addition, it shows a useful example of how a mechanism based on a centralized authorization system can work.

REFERENCES

[1] Mark G.Graff, Kenneth R. van Wyk. Secure Coding: Principles and Practices, O'Reily Media, Inc, pp.12

[2] Smolander, K. (2002, October). Four metaphors of architecture in software organizations: finding out the meaning of architecture in practice. In Proceedings International Symposium on Empirical Software Engineering (pp. 211-221). IEEE.