SQA CONSULTANT
Inventing the future....

SECURITY TESTING FOR THIRD PARTY COMPLIANCE

CLIENT

Our client deals with number of Flexible Alternating Current Transmission System (FACTS) devices. These devices are installed on the transmission lines and help with power flow control. When installed on transmission lines; major purpose of these devices is to:

- Protection of the associated FACTS devices from fault currents
- Maintaining normal power flow in the line when contributions from the FACTS device are not required.

These Devices are installed in the field, on the transmission lines or below them on the ground. These filed devices then communicate wirelessly with a coordinator in the substation, which in turn relays commands and information to a protocol concentrator. The protocol concentrator communicates with a desktop application for remote control of the field devices. The communication scheme consists of both wired and wireless protocols, some of which are proprietary while others are standard. The solution consists of desktop applications, web applications, Linux systems and Smart IoT devices running secure firmware.

PROBLEM STATEMENT AND CHALLENGES

The client contacted us for the cybersecurity evaluation of their end-to-end system before submitting their solution to a third-party evaluation. This was a critical for their success since the solution was to be deployed on Power Grids worldwide and Grids are considered highly sensitive critical infrastructure in most countries and governments/distribution companies were extra cautious about getting it installed at their stations without proper evaluation.

We were required to fully evaluate, test, and harden the system before they went for the expensive third-party audits. The goal of this assignment was passing those audits on the first attempt.

During multiple iterations of security testing, we faced several key challenges:

- In order to test the security of the proprietary protocols, we had to figure out how to capture, decipher, modify and re-transmit the traffic on the wired network.
- An even bigger challenge was to capture, decipher, modify, and re-transmit the traffic over the wireless network.
- One key aspect of security was the randomness of security keys, so we had to figure out a way to test if the keys were truly random or not.
- Another challenge was to test all the services against millions of possible packet structures in order to test how well corrupt/invalid packets were handled.
- Finally, we had to make sure that we tested for wide range of scenarios not thought of by the designers and hence not covered in the requirements or incorporated in the design.

SLOUTION STRATEGY

We started this activity by going through the design and architecture documentation in order to fully understand how the system was designed and how each component worked. We focused especially on the security design and architecture documentation, going through all the security mechanisms in place as well as studying all the communication protocols in depth at the packet level.

Here`s how we handled the various challenges:

- For the wired protocols, we used a two-pronged approach. First we designed python scripts using Scapy tool to copy the packet structure and transmit custom packets to the targets. Secondly, we used actual devices and modified the code inside such that they periodically sent rogue packets to the target. These two techniques allowed us to bypass the step requiring capturing and deciphering the packets as we could use the existing code and hardware to transmit modified/corrupted packets of our choosing.
- For the wireless protocols, we had two challenges, we needed a way to listen in on the wireless transmissions and we needed to simulate rogue devices sending out malicious traffic. For the first challenge, we put probes on the SPI pins coming out of the radio modules on the device. These signals were then sent to a Logic Analyzer (Salae) and once we had figured out the correct frequency and timing and the data was captured,

we exported it into excel and designed a Macro to decipher it in granular details down to a single bit.

- Next part was simulating a rogue device, this was accomplished by using one of the actual devices and adding a script in the code that would send rogue packets to the target devices periodically. Some of these packets had valid hashes while others had invalid hash values. This allowed us to test if the system was correctly validating integrity of the packets as well as test for systems resilience against packets that were incorrectly formatted or had invalid data.
- The system utilized several pairs of security keys, some of which were generated on the devices during session establishment. We had to make sure that these were truly random. This was not an easy task as we had to manually establish/re-establish the sessions to force key generation and then we copied those keys to an excel sheet for comparison. This technique allowed us to catch a high severity issue where one of the keys appeared to have a repeating pattern. Further investigations of the code allowed us to uncover a serious flaw where part of a counter was being copied to the key variable.
- The solution employed various protocols and we needed to make sure that all the nodes handled rogue packets safely. To achieve this goal we used the fuzz testing techniques. While it is possible to do a blind fuzz test where we use freely available open source scripts bombard the system with corrupted data. It is always more efficient and productive to do a grey box or white box fuzz testing technique. This required an in-depth study of the protocol specifications, structure, allowed ranges, data types, security functions and handshake mechanism.
- Next, we used a tool called fuzzing frameworks like Peach Fuzz or Sulley/BooFuzz, that allowed us to replicate the packet in python. These frameworks provide some handy functions that help automate the task of creating fuzzing test cases. The goal of this exercise is to generate a script that would make TCP connection to the target device behaving like the actual node and then start sending packets that an actual device would send followed by hundreds of thousands of corrupt or mutated packets that were only slightly different from the original. It is always preferred to keep 99% of the packet as close to the original as possible and mutate each byte separately, because it makes it more likely that the packet will not be blocked at the firewall.
- One of the biggest challenges in security testing is to think outside the requirements and find issues in the buisness logic or issues that are outside the scope of the requirements/design of the products. This required a lot of exploratory testing where our team used their experience and expertise to identify several serious flaws. For example, we did not have any requirements for the maintenance tools as the client did not think that security was important if a tool was not going to the customers. However, when we studied these tools, we realized that one of these tools that was used for troubleshooting by the maintenance engineers could connect to the devices without authentication, and when we looked at the code, we realized that these tools were effectively using backdoors to connect and issue all kinds of commands to the devices. This was a serious issue that would have been a deal breaker for many auditors and customers. The client realized the severity of the issue and put in security mechanisms in all their maintenance tools.

HIGHLIGHTS

As a result of our cybersecurity evaluation of the client's end-to-end system, which contains IoT devices, coordinators, data concentrators, web applications and desktop applications, following are some of the highest severity issues that were identified and fixed in the end-to-end system:

- Identified a critical weakness in security authentication mechanism related to the generation of security keys
- During fuzz testing, identified several packets in multiple devices that could be remotely sent to crash the system
- Identified hidden user accounts and broke passwords
- Bypassed authentication in a desktop application
- Retrieved secure data and files from the web application bypassing the login.

CONTACT US

IExplore ways to use our expertise in growing your
business while establishing a valuable partnership with
us.
Contact our consultants at:
Phone: **+1.412.533.1700 (Ext: 585)**
E-mail: **info@sqaconsultant.com**
Website: **www.sqaconsultant.com**