DOI: http://dx.doi.org/10.14236/ewic/ICS2015.13

Insecure by Design: Using Human Interface Devices to exploit SCADA systems

Grigoris Tzokatziou
School of Computer Science and Informatics
De Montfort University, Leicester, UK
p11248630@myemail.dmu.ac.uk

Leandros Maglaras
School of Computer Science and Informatics
De Montfort University, Leicester, UK
leandros.maglaras@dmu.ac.uk

Helge Janicke
School of Computer Science and Informatics
De Montfort University, Leicester, UK
heljanic@dmu.ac.uk

Modern Supervisory Control and Data Acquisition (SCADA) systems which are used by the electric utility industry to monitor and control electric power generation, transmission and distribution, are recognized today as critical components of the electric power delivery infrastructure. SCADA systems are large, complex and incorporate increasingly large numbers of widely distributed components. Cyber-attacks usually target valuable infrastructures assets, taking advantage of architectural/technical vulnerabilities or even weaknesses in the defense systems. Even though novel intrusion detection systems are being implemented and used for defending cyber-attacks certain vulnerabilities of SCADA systems can still be exploited. In this article we present an attack scenario based on a Human Interface Device (HID) device which is used as a means of communication/exploitation tool to compromise SCADA systems. The attack, which is a normal series of commands that are sent from the HID to the PLC cannot be detected through current intrusion detection mechanisms.

Keywords: SCADA, cyber security, HID

1. INTRODUCTION

One of the biggest issues that SCADA systems face is that they were designed to work solely in their environment segregated from inter-connected IT networks or ad-hoc systems. The primary reason for this is that there was no need for remote access at the time of their introduction. However, nowadays organizations want to establish local convenience or remote access, which will enable them to take decisions on production changes and apply them quickly from a centralized location rather than have to travel to different locations in order to make changes to their ICS systems. This interconnection of Industrial Control System (ICS) networks with organizational ICT network infrastructures, and even with the exterior has brought a new wave of security problems and attacks. In fact, the number of externally initiated attacks on ICS systems has increased much more rapidly than internal ones Igure et al. (2006).

Moreover, SCADA communication protocols, which are responsible for the interaction between field devices, such as PLC (Programmable Logic Controller) or RTU (Remote Terminal Unit) components and the stations that control and monitor them, pose security concerns Robinson et al. (2015). One such example is the Modbus protocol, originally developed by Modicon.

Modbus messages are exchanged between entities by using TCP, which imposes more complexity with regard to managing the reliable delivery of packets in a control environment with strong real time constraints. In addition, it provides attackers with new avenues to target industrial systems Carcano et al. (2008). Modbus is one of the most popular protocols for SCADA applications, but it suffers from security problems such as the lack of encryption or any other protection measures which thus exposes it to different vulnerabilities.

Serial communication has not been considered as an important or viable attack vector, but the researchers say breaching a power system through serial communication devices can be, under some circumstances, than attacking through the IP network since it does not require bypassing layers of firewalls Ashford (2013). Potential attackers use

common vulnerabilities in order to put controlling servers into infinite loops. This case is not the same as not having access to the field network, but it could mean that the operators are not aware of the actual conditions on the ground. The worst of the vulnerabilities exposed so far enables a potential buffer-overflow attack, whereby code stored for one purpose overflows its container, and can end up being executed in different time instances than programmed to or in a different way. This allows for malicious code to be injected into control servers, giving access to attackers to the control system.

If there is not adequate security in place, then the impact of an attack or a disruption in the process of these critical infrastructures could prove hard to deal with, such impacts include :

- Physical Impacts - Loss of life, property and data, also potential damage to the environment i.e. oil spillage.
- Economic Impacts - Loss of income, revenue from attacks which cause the normal process of industrial systems to be halted.
- Social Impacts - If an attack compromises transportation networks or systems which will have a social impact i.e. water distribution systems the public will loose confidence in the Government.

Modern intrusion detection systems (IDSs) focus mainly on analyzing the traffic that flows in the network. By capturing behaviour or traffic patterns in the network, misbehavior is detected and dedicated security events are reported. IDSs can be classified into centralized intrusion detection systems (CIDSs) and distributed intrusion detection systems (DIDSs), according to the way in which their components are distributed. Due to the rapid increase of sophisticated cyber threats with exponentially destructive effects, IDSs are systematically evolving Leandros et al. (2014); Gil et al. (2013); Tiago et al. (2014). Among other approaches, neural networks, support vector machines, K-nearest neighbor (KNN) and the Hidden Markov model can be used for intrusion detection, while existing signature-based network IDS, such as Snort or Suricata can be effective in SCADA environments. However, most of the approaches that have been introduced recently cannot deal with attacks that come straight from serial communication devices.

In this article we investigate the vulnerabilities of a SCADA system and perform an attack directed to an ABB PM564 PLC, using a HID . The Teensy device used is an Arduino based one that allows the user to utilize onboard memory storage on a microcontroller and to emulate a keyboard/mouse. By using this HID device (see Figure 1) we can bypass any autorun protections on the system since it is shown as a keyboard that is connected to the workstation. By sniffing the packets that are exchanged between the HMI and the PLC we manage to extract the information of a STOP command, replicate it and store it in a web host. As the PLC has been set to run, we insert the Teensy HID device into the engineer's machine, or a machine connected to the same subnet. Once the Teensy USB has been plugged into the system, it waits for a specific amount of time in order to download the code and execute it. The attack, although primitive, cannot be detected by any current IDS as it involves the execution of a legitimate 'STOP' order from an authorized device.

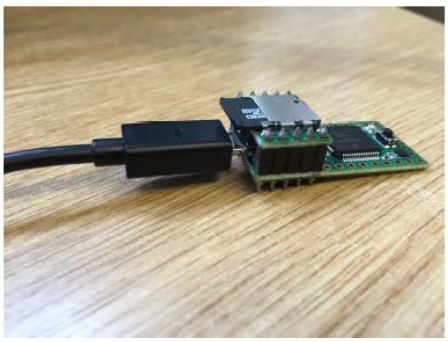

Figure 1: *Teensy HID*

2. HID RELATED ATTACKS

There are a lot of threats to our National Critical Infrastructure systems (SCADA) which have a major effect not only on the public, but also the government and the economy of a country or nation. Most of the attacks have used sophisticated mechanisms to gain entry and exploit well-known vulnerabilities and ones that have yet to be discovered.

In reference to SCADA there have not been any attempts to attack their systems with a HID device, such as the Teensy 3.1 which falls under the HID category as this is how it is recognised by the system although it connects via USB. Further research showed that the primary use of the Teensy board was for personal projects which can all be found under the PJRC13 store. The teensy board itself has been used as a penetration testing tool kit.

3. EXPERIMENTAL SETUP

Our earlier research showed that the commands sent from an engineer's machine to a PLC go through the TCP/IP protocol. We connected the machine and the PLC together (see Figure 2) via a switch so that we could confine any action to

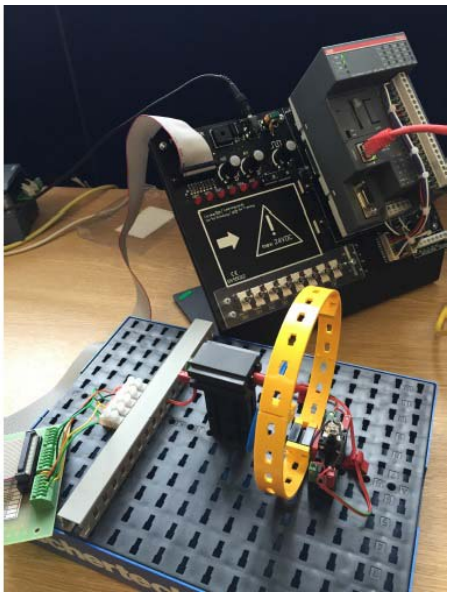

Figure 2: *Architecture of the PLC*

a safe environment without disrupting any other interconnected devices on the network.

By using the Codesys software to start and stop the PLC, while sniffing the connection between these two devices, we noticed that the commands sent between these devices were not encrypted, but rather, were in plain text (HEX). This characteristic is a vulnerability of the system that can be exploited. Since no authentication/encryption is used we can replicate this information without the need of the ABB suite of tools. The packets that are exchanged have a lot of raw data that do not perform any specific action on the PLC. One of the most important findings is the 3-way handshake being performed between the PLC and the computer. To attempt any sort of command execution we need to establish a connection using this 3-way handshake mechanism.

The packets also revealed that when an AA// was included in the raw data it meant that the following code was an attempt at communication. The above syntax was a key, as without this syntax we would have to go through every single piece of data and use a trial and error approach in order to interpret data to actual commands. In order to craft packets, we used a Linux tool which has been made available for Windows called Scapy, which is able to create a packet with a set of parameters specified by the user. Specifically, it is a packet manipulation tool developed by Phillipe Biondi with the ability to forge, decode packets from a different range of protocols and to send them or reply to a request.

In order to be certain that this data did not change per every single connection attempt, we captured the data many times and compared these values. We concluded that the data exchanged in order to perform specific actions on the PLC are exactly the same every time. This finding lead us to the view that ABB PLCs with the specific firmware version use the same set of data to communicate with a workstation. This is very beneficial for our research since it means the attack can work for the same model of PLC without the need to alter the code. Using the same strategy we managed to sniff the 'STOP' command that is sent from a workstation to the PLC. The series of commands was crafted into a packet and was correctly sent from our device to the PLC, since no authentication or encryption was demanded from the PLC. The script was converted to a simple executable program and the file is hosted on local internet host; ready to download.

4. RESULTS AND ANALYSIS

The attack script starts by accessing the Scapy library and importing the time which is important because without this the PLC and engineer's machine could not talk to each other in different times, i.e. they have to be synced. We create a connection socket specifying the IP and port of the PLC. We then define four variables that include the RAW Hex data, send the request, and wait (sleep) 0.1s before sending the second request. After the second request is sent we dispatch the third, which is the ACK and the fourth request is the STOP command. Finally the last raw data that we sent is to close the socket (See Figure 3). We have to mention here that the full information cannot be disclosed in this article for security reasons.

Based on the research and experimental work that we conducted we found that a ready malicious executable file can cause a PLC to STOP running. The executable file can be downloaded from the internet and executed from the workstation that controls the PLC. It can be copied to the startup location of the workstation so that the payload will run with every restart.

5. PROPOSED DEFENSE MECHANISMS

The Teensy HID device appears on the system under the Universal Serial Bus. Traditionally, Windows does not require any privileges for the installation of this device as these drivers are already part of the O/S and by default are automatically installed. A way to stop any input from a certain HID device is to blacklist it by vendor and product ID, but this is not very reliable as the vendor can change the identifiers which then can by-pass the blacklist enabled within Windows Crenshaw (2011).

```
def pkt_send():
    mysocket = socket.socket()
    mysocket.connect
        (('192.168.0.10',1201))
    myStream = StreamSocket(mysocket)
    req = Raw(load="\xbb\xbb\x00\x00
        ......")
    req1 = Raw(load="\xbb\xbb\x00\x00
        .....")
    req2 = Raw(load="\xbb\xbb\x00\x00
        .....")
    req3 = Raw(load="\xbb\xbb\x00\x00
        .....")
    myStream.send(req)
    time.sleep(0.1)
    myStream.send(req1)
    time.sleep(0.1)
    myStream.send(req2)
    time.sleep(1)
    myStream.send(req3)
    time.sleep(0.1)
    mysocket.close()
pkt_send()
```

Figure 3: *Attack code*

Another option would be to create a policy within Windows to allow only one keyboard and mouse to be present at any one time. Another available option is to allow the administrator to specify a list of device set-up GUIDs (global unique identifiers) for device drivers that windows is allowed to install. Cryptographic solutions are incomplete without effective key management which remains an open problem in SCADA networks.

6. CONCLUSIONS

This article has investigated the vulnerabilities of a SCADA system and performed an attack directed at an ABB PM564 PLC, using a HID (Human Interface Device). This PLC uses the Codesys programming software as its SCADA programming interface. The HID device is inserted into the workstation and is recognized as a keyboard. Once the Teensy USB has been plugged into the system it will wait for a specific amount of time (set in the code) in order to download the code and execute it. The attack, although primitive, cannot be detected by any current IDS, since it involves the execution of a legitimate 'STOP' order from an authorized device.

The malicious packet which alters the behaviour of the PLC can be executed in random time periods and in different PLCs, thus making the situation harder to be controlled. The article has also investigated possible counter measures and defense mechanisms against this kind of cyber attack. As future work, more sophisticated attacks are going to be performed with real time defense systems tested against them in order to assess their detection capabilities.

REFERENCES

Igure, V. M., Sean A. L., and Ronald D. W. (2006) Security issues in SCADA networks. *Comput. Secur*, 25 (7), 498–506.

Robinson, M., Kevin, J., and Helge, J. (2015) Cyber warfare: Issues and challenges. *Comput. Secur.*, 49, 70–94.

Carcano, A. et al. (2008) SCADA malware: A proof of concept. In: *Third International Workshop on Critical Information Infrastructure Security*, Berlin, Germany: Springer.

Ashford, W. (2013, Oct. 18) *US Researchers find 25 security vulnerabilities in SCADA systems*. ComputerWeekly.com. Available from http://www.computerweekly.com/news/2240207488/USresearchers-find-25-security-vulnerabilities-in-SCADA-systems

Leandros, A. M., Jianmin, J., and Tiago, J. C. (2014) Integrated OCSVM Mechanism for intrusion detection in SCADA systems. *Electron. Lett.*, 50 (25), 1935–1936.

Tiago, J. C. et al. (2014) A distributed IDS for industrial control systems. *Int. J. Cyber Warfare and Terrorism (IJCWT)*, 4 (2), 1–22.

Gil, P. M. et al. (2013) RepCIDN: A reputation-based collaborative intrusion detection network to lessen the impact of malicious alarms. *J. Netw. Syst. 730 Manage.*, 21 (1), 128–167.

Crenshaw, A. (2011). *Plug and prey: Malicious USB devices*. Available from http://www.irongeek.com/downloads/Malicious%20USB%20Devices.pdf

Khurana, H. et al. (2010) Smart-grid security issues. *IEEE Security & Privacy*, 8 (1), 81–85.