

# Human Machine Interface development and vulnerabilities in SCADA systems

Wasfi Momen

CSC 6222 – Introduction to Infosecurity - Georgia State University  
wmomen1@gsu.edu

## I. INTRODUCTION

Today's society rely on industrial control systems to provide services like gas, water, and electricity with differing loads of demand and supply. Supervisory Control and Data Acquisition (SCADA) is a common framework used to describe control systems feedback in industrial systems. In the 21<sup>st</sup> century, SCADA systems adopt increasingly more computers to control industrial processes, and therefore are targets for nation-states, individuals, and organizations.

SCADA systems do not only exist in the industrial plant, but also play a role in building automation systems. Most universities or organizations that manage a collection of buildings within the campus will utilize a networked SCADA system to maintain control over temperature, water, electricity, and mechanical systems such as elevators.

However, all SCADA systems at some point need to be human operable. Whether it is a mechanical switch, a console command, or a web interface, the feedback system needs to have human intervention and control right down to the individual mechanical components. Today's corporations mainly use the Human Machine Interface (HMI), to represent this component of SCADA and are typically web interfaces that can be accessed through major web browsers.

In this paper, we discuss the development of the HMI, summarize the research that attempts to secure vulnerabilities, and present further protections for the HMI with an implementation created in a SCADA honeypot to model threats.

## II. BACKGROUND

### A. History of SCADA development

To understand the research and implementations to protect SCADA vulnerabilities, a quick overview of the development of SCADA is required. Industrial control systems developed rapidly as the demand for services like natural gas, water purification, and electricity generation escalated during the 1960s along with

technological innovations in both engineering and computers.

SCADA has developed through generations of improvement with new technology. These generations are referred to as the monolithic, distributed, networked, and Internet of Things (IoT) stages [3]. Monolithic and distributed describe systems that used a mixture of mechanical and command line interfaces. The distributed, networked, and IoT SCADA systems rely on a mixture of present day protocols like HTTP and TCP as well as protocols specified by IEC 60870 like Modbus or DNP3.

Most modern day SCADA systems include Programmable Logic Controllers (PLCs) and Remote Terminal Units (RTUs). These devices control the machines running on the industrial plant based on current and past data readings. Other computers placed either on or off-site delegate instructions to PLCs and RTUs to meet the supply and demand of the plant. SCADA systems also include data historian and logging servers that serve to track production of the plant and calls to services.

### B. Research and Response

However, the networks for SCADA systems are very vulnerable to attack since most of the computer to device protocols like BACnet does not include any sort of cryptographic security [3] and could not be upgraded to include cryptographic protocols since the data buses are very small. With the invention of the World Wide Web in the 1990s, protocols with more security features could be used to wrap around the vulnerable ones, so most research focused on using cryptography to secure the networks controlling SCADA systems [2]. Controlling the TCP or UDP connections within the network became a priority, and eventually protocols were established like ISA 95/99 and ASME standards and government guides were produced from organizations like NIST, FIPS, and ISO [3]. Current

research focuses on models or introduces a new security framework to protect SCADA systems.

### C. Problem of the HMI

By looking at the SCADA system from a layered view of computers, current and past research has succeeded in modeling, implementing, and improving the protection of networks within SCADA systems. However, the applications that run within these networks are now targeted towards attack. Different systems have different tech stacks, but all need a common interface in which to issue commands either on-site or remotely: the HMI.

In the last two decades, the popularity of web browsers redefines the HMI into a browser-compatible space. All interactions now go through a web application that is easier to modify and update. Companies that create SCADA packages now offer the hardware PLCs and RTUs along with web and mobile accessible pages. While these applications increase usability, web technologies increase the attack space further for SCADA systems more so than that of network protocols, which are slower to change and develop.

Out of the standards and guides researched, only ICS-CERT had any mention of potential vulnerabilities in web applications using cross-site scripting [1]. Since there is no research into security vulnerabilities in HMI, there are no codes in any standard or guide and therefore no models in existing tools to detect web application vulnerabilities. If there are any attacks on HMIs in SCADA systems, they are normally reported to ICS-CERT and have dedicated CVEs, such as the case with Siemens WinCC version 7 (CVE-2014-8551).

### D. Attacking the HMI

In this research, we will implement attacks on the HMI of a mock SCADA setup. The attack will not include compromising authentication or access control and assume the attacker controls an off-site computer with access. The point of the attack is to see if a vulnerable HMI triggers the intrusion detection tools and logs to ensure that a loss of integrity within the system can be tracked.

## III. METHODS AND IMPLEMENTATION

To test the HMI attack, the following software was used:

- i. Conpot, a SCADA honeypot that emulates a Siemens S7-200
- ii. Snort, a tool for SCADA to detect and alert for malicious behavior

- iii. Docker, a tools used for virtualized environments
- iv. OS is the Ubuntu 17.10 Artful Aardvark
- v. Default HMI provided by Conpot, which is an HTML file with python template strings

To compromise the web application, we will execute a simple cross-site scripting (XSS) attack where a script tag is inserted to repeatedly call a function forever for as long as the page is open. This is useful since the attacker might want to watch a page and maybe add a POST function to send a JSON payload to gather more information on the target.

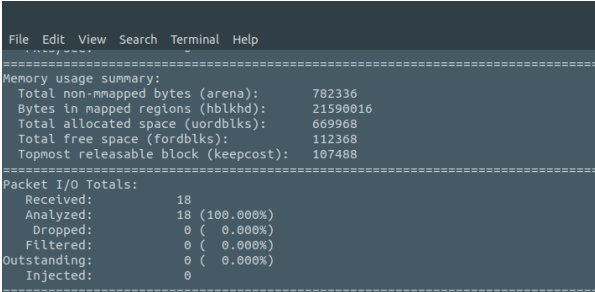
```
<script>
function httpGet() {
    console.log("Getting hacks");
    var xml = new XMLHttpRequest();
    xml.open("GET", "{urlForAttacker}", false);
    xml.send(null);
    return xml.responseText();
}

function run () {
    httpGet();
    setTimeout(function () {runF() }, 500);
}

runF();
</script>
```

Docker containers were set up both for Conpot and for Snort and connected via a network bridge. In this setup, Snort can monitor the network of all protocols, including the SNMP and HTTP protocols.

At the default setup without any vulnerabilities employed SNORT reported zero filtered packets that deserved a warning.



```
File Edit View Search Terminal Help
=====
Memory usage summary:
Total non-mapped bytes (arena):      782336
Bytes in mapped regions (hblkhd):    21590016
Total allocated space (uordblks):     669968
Total free space (fordblks):         112368
Topmost releasable block (keepcost): 107488
=====
Packet I/O Totals:
Received:      18
Analyzed:      18 (100.000%)
Dropped:       0 ( 0.000%)
Filtered:      0 ( 0.000%)
Outstanding:   0 ( 0.000%)
Injected:      0
=====
```

\*. Snort reports no errors in the default setup

With the script in place, there were more TCP packets recorded since the HTTP request was continuously being sent. However, there were still zero packets reported.

```
File Edit View Search Terminal Help
Snort processed 23 packets.
Snort ran for 0 days 0 hours 0 minutes 13 seconds
Pkts/sec: 1
=====
Memory usage summary:
Total non-mapped bytes (arena): 782336
Bytes in mapped regions (hblkhd): 21590016
Total allocated space (uordblks): 669968
Total free space (fordblks): 112368
Topmost releasable block (keepcost): 107488
=====
Packet I/O Totals:
Received: 23
Analyzed: 23 (100.000%)
Dropped: 0 ( 0.000%)
Filtered: 0 ( 0.000%)
Outstanding: 0 ( 0.000%)
Injected: 0
=====
```

\*. Snort reports no errors in the compromised setup

Therefore, with a trivial XSS attack, Snort failed to recognize the attack. However, the data historian should be able to track the number of requests and see the higher number of TCP packets and model an alert, as described by other research on SCADA systems. Unfortunately, Conpot does not come with a data historian/logger by default.

While Snort can be set up to monitor packets of different rulesets, it does lack the ability to interface with application-specific data. It can go into details about specific http packets such as the length of the content payload or headers, however each one of the rules in Snort would have to reflect the exact headers and length that ran in the HMI web application. If the company creating the web application provided a tested Snort rule configuration, then it would provide more piece of mind knowing the data being sent and received is within normal parameters.

Other problems in this setup was building the HMI itself. Running Conpot in docker is a good idea, but originally an Angular web application was considered. However, Conpot wouldn't have played well with the Angular setup since the data coming from the emulated SNMP server goes through python template strings in order to render. Therefore, using Javascript to render

the data would be futile. A perfect use case would be to use the HMI designated for the emulated PLC, which is WinCC version 8 by Siemens. However, obtaining a copy of this software is only for potential buyers of the Siemens package.

## IV. CONCLUSION

By looking at the attacks internal to a SCADA's HMI and surveying the research related to SCADA vulnerabilities, the clear direction of further work into protecting SCADA relies in the ability to detect web application vulnerabilities. Tools such as Snort do help in detecting network intrusions in SCADA systems, but do very little to protect against web vulnerabilities.

Further guides and standards should consider security of the HMI since it is the most malleable aspect of SCADA systems. Prior research and implementations take good care that the networks are not compromised and give proper alerting if they exceed boundaries. By protecting against HMI vulnerabilities, managing SCADA systems will be easier not only on the plant manager, but also on cybersecurity teams that need to respond to attack threats that exploit multiple vulnerabilities.

## REFERENCES

- [1] "Recommended Practice Case Study: Cross-Site Scripting" Homeland Security, February 2007 URL: [https://ics-cert.us-cert.gov/sites/default/files/recommended\\_practices/RP\\_CaseStudy\\_XSS\\_20071024\\_S508C.pdf](https://ics-cert.us-cert.gov/sites/default/files/recommended_practices/RP_CaseStudy_XSS_20071024_S508C.pdf)
- [2] E. Byres, B. Chauvin, J. Karsch, D. Hoffman and N. Kube, "The special needs of SCADA/PCN firewalls: architectures and test results," 2005 IEEE Conference on Emerging Technologies and Factory Automation, Catania, 2005, pp. 8 pp.-884.
- [3] I. Vinay, L. Sean, W. Ronald "Security issues in SCADA networks", University of Virginia, Charlottesville, VA, March 2006
- [4] W. Henrik, "SCADA Intrusion Detection System Test Framework", University of Agder, Norway, May 2017