Human Machine Interface development and vulnerabilities in SCADA systems

Introduction

- Why your topic is important (convince us!)
- Where is it used? Applications
- What you will talk about / do
- Overview of the rest of your paper

Background and related work

- Any relevant and specific info, e.g. software / hardware statistics, equipment used
- What other people had to say on this topic
- What other people did on this topic
- Problems and shortcomings of their work
- How your work is different and better

Proposed methodology

- Your approach to the problem
- What you did
- Code / Algorithms
- What did / didn't work
- Results – include graphs, equations, pictures, as appropriate

Conclusions

- What was accomplished / learned
- What you would have done differently
- Future work

References

Appendix

- **Main point:** Go over how the interface between human and machines for SCADA processes developed over time and how the web HMI is ignored for security vulnerabilities since the focus is on networks.
- Use GSU, CANDU, and Georgia Nuclear Power plants as examples

Technology Discussion

A. What SCADA is and how does the HMI come into play
    a. Go over SCADA framework (data historian, control, where digital meets analog, RTUs and PLCs). Emphasis on data.
    b. 2 sentences on how everything was old in microcomputer world and then distributed systems came along. It was console then applications with basic GUI.
    c. Now the world is web.
B. Research in Vulnerabilities of HMI

    a. Pretty much tossed aside, people focus on networks [cite from garbage sources].
    b. Talk about NIST and CERT, SANS courses
    c. Talk about ASME, IEEE, standards and specifications
    d. Security through obscurity. Show CVEs, show ICS-CERT alerts, discuss OT/IT. Discuss the scope of problems and what can be reasonable assured with not just updates, but action.

C. Project Purpose
    a. Look at the current SCADA scale and see vulnerabilities that affect it and the systems with it.
    b. Won't be looking into password/auth hacks. Assume control of external operator laptop with hacked credentials and access.
    c. We should secure in what happens to changes in the data to ensure integrity, not only its transportation and rulesets.

Problem Approach

A. Identify XML, JSON, Excel, Word, possible attacks. Give example of Angular interface in Siemens application. Source siemens slides. (qt and angular qualification needed)
B. Use conpot with an HMI (either built-in or maybe find one). Give example of multiple attacks browser-based with OWASP (must give qualifications like siemens shows windows 10 with wincc, so this is reasonable)
C. If possible, try to do XSS, remove SVG by tree transversal or other attack.
D. Show splunk monitoring, snort monitoring. Compare with monitoring of packaged systems. Maybe Wireshark.

Conclusion

- Focusing on networks, improper organizational reporting and security through obscurity, and dealing with the symptoms is bad.
- Realistically adding more monitoring functions including SNORT and Splunk-like stuff is good.
- Check for external, unknown Javascript, check for symbols that aren't supposed to be in XSS. SCADA-antivirus needs to check not only for signatures but also polymorphic viruses.
- A web attack is faster and easier to deploy but also detect. If caught, might signal a bigger issue.

****Things for tomorrow/today: try the honeypot and read papers on it, do slides first then paper (can say in the process of HMI), and then do paper.**

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 21st 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.

Background and related work

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 [Wagneer, 18]. 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 in order 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 [Sciencedirect] article] 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 [https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1612765&tag=1]. 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 [https://ac.els-cdn.com/S0167404806000514/1-s2.0-S0167404806000514-main.pdf?_tid=52dce760-cac4-4c08-a1b1-a7500b38aa30&acdnat=1524864535_1e42ac4b1d32fc50a118a2be11ddda8c]. 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 [https://ics-cert.us-cert.gov/sites/default/files/recommended_practices/RP_CaseStudy_XSS_20071024_S508C.pdf] using cross-site scripting. 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.

Methods and Implementation

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

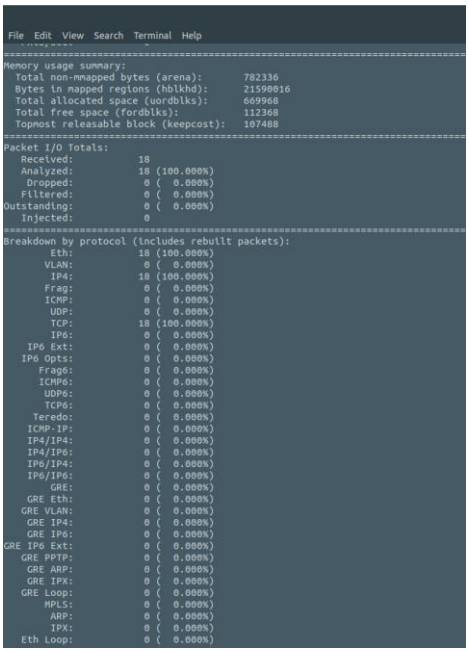
i. Conpot, a SCADA honeypot that emulates a Siemens S7-200
ii. Snort, a tool for SCADA to detect and alert for malicious behavior
iii. Docker, a tools used for virtualized environments
iv. OS is the Ubuntu 17.10 Artful Aardvark
v. Default HMI provided by Conpot, which is an HTML file with python template 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.

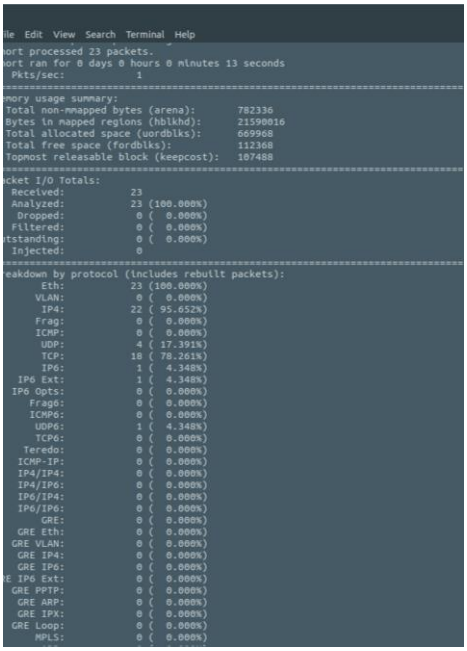
```
<script>

        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.



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.



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.

Conclusion

By looking at the attacks internal to a SCADA's HMI and surveying the research related to SCADA vulnerabilities,