The Importance of Testing Smart Grid IEDs against Security Vulnerabilities

PUBUDU EROSHAN WEERATHUNGA, ANCA CIORACA
General Electric Grid Solutions - Canada
Email: anca.cioraca@ge.com

SUMMARY

As the Smart Grid becomes highly interconnected, the power protection, control, and monitoring functions of the grid are increasingly relying on the communications infrastructure, which has seen rapid growth. At the same time concerns regarding cyber threats have attracted significant attention towards the security of power systems. A properly designed security attack against the power grid can cause catastrophic damages to equipment and create large scale power outages. The smart grid consists of critical IEDs, which are considered high priority targets for malicious security attacks.

For this reason it is very important to design the IEDs from the beginning with cyber security in mind, starting with the selection of hardware and operating systems, so that all facets of security are addressed and the product is robust and can stand attacks.

Fact is that the subject of cyber security is vast and it covers many aspects.
This paper focuses mainly on one of these aspects, namely the aspect of IED firmware system testing from the security point of view. The paper discusses practical aspects of IED security testing, and introduces the reader to types of vulnerability exploitations on the IED communication stack and SCADA applications, practical aspects of security testing, the importance of early vulnerability detection and ways in which the security testing helps towards regulatory standards compliance, such as NERC-CIP.
Finally, based on the results from the simulated attacks, the paper discusses the importance of good security practices in design and coding, so that the potential to introduce vulnerabilities is kept to a minimum.
Designing with security in mind also includes good security practices, both in design and coding, and adequate policies for the software development process. Critical software development milestones must be established, such as design and test documentation review, code review, unit, integration and system testing.

KEYWORDS

IED = Intelligent Electronic Device
OT = Operational Technology
TCP/IP = Transmission Control Protocol/Internet Protocol
IT = Information Technology
CVE = Common Vulnerabilities and Exposures
CPU = Central Processing Unit
OSI = Open System Interconnect
UDP = User Datagram Protocol
DNP = Distributed Network Protocol
IP = Internet Protocol

MAC = Media Access Control
MTU = Maximum Transmission Unit

1 INTRODUCTION

The reliability and security of the power grid is essential. Every device in the grid must participate into making the security of the grid a reality.
The modern IEDs must be themselves designed with cyber security features and functions which, on one side, will make them robust and resistant to cyberattacks and, on the other side, allow them to assist in secure centralized management and remedial action schemes in the grid.
Cyber security is a very complex endeavor at any level, whether at the IED level or the whole OT and for that reason proper and detailed cyber security system testing of IED firmware is of utmost importance, as the system testing phase is the last point in the firmware development before a new release hits the market.
In this paper we are going to present some practical aspects of IED security testing which we consider essential to a robust product.

1.1 GENERAL

Cyber security in substations and IEDs is still in an early age. While this is a subject that has been given a lot of attention and for a long time in the IT world, utility companies and industrial control systems have been slow in acknowledging its importance. This is partially because of the fact that traditionally these environments have been isolated, the communication being done over dedicated lines and proprietary protocols and providing some sort of security by isolation and obscurity. Utility companies for instance were used to believe (and many still are) that physical security is all that is needed. Hence they developed very strong policies for personnel access into the substation. While these are very good measures, they are far from being enough in today's world. There are a few reasons contributing to this situation:

- Communications is not isolated anymore. IEDs may communicate with SCADA and control systems over an IP based network, even the Internet.
- Communications protocols are not proprietary anymore. Many standards have been developed for IED protocol communications for the purpose of interoperability among various vendors.
- There has been a marked increase in malicious attacks and the focus on unauthorized access has broadened from amateur attackers or disgruntled employees to deliberate criminal or terrorist activities aimed at impacting large groups and facilities.

Only recently, due to NERC CIP regulations [1], cyber security in the electric grid starts to see more focus and vendors are asked to add more security protections in IEDs.

This paper focuses on testing the security aspects of the IED. Also, based on the results from the simulated attacks, the paper discusses the importance of good security practices in design and coding as well as adequate policies for the software development process, so that the potential to introduce vulnerabilities is kept to a minimum.

2 IED SECURITY TESTING

Testing the IED from the security perspective involves testing all the security features and mechanisms introduced by design in the IED, as well as testing for possible vulnerabilities, which are security weaknesses inadvertently introduced in software due to poor design and coding.

Most IEDs are already implementing or in the process of implementing security mechanisms for supporting utility companies with NERC CIP compliance. The five security functions NERC CIP addresses and utility companies need to respond to are: Identify, Protect, Detect, Respond and Recover. To assist utility companies, IEDs may implement security features from all these aspects. For instance protection measures within the IED may include mechanisms for user authentication and accounting, user authorization based on need and encryption on communications protocols. Detection measures may include event logging of all security events with classification and timestamps.

All these security mechanisms and features must be thoroughly tested, initially by developers and secondly by a separate testing group, who performs system tests for functionality, to ensure the end result is as described by the requirements in the functional specification. The test cases must be formulated to follow exactly these requirements. But this is not enough.

The second part of security system testing must address vulnerabilities.

Vulnerabilities may become a problem if they are uncovered and exploited by malicious organizations or individuals. Depending on the weakness, various penetration schemes may be implemented by attackers. So finding these weaknesses as early as possible and definitely before the software is released, is highly desirable. Of course, some of these weaknesses may have been already addressed in earlier stages of the software development, during the design or code review, as well as during the unit and integration testing. But the last chance is during the system testing and, since this testing is done by different people than the developers, from a different perspective and with different tools, chances are that additional vulnerabilities will be found at this point.

3 VULNERABILITIES EXPLOITATION

3.1 Vulnerabilities and Ways Hackers Exploit Them

Vulnerabilities are weaknesses within the product that may be penetrated and exploited by an intruder. The penetration may be done either from outside or inside the Electronic Security Perimeter. Vulnerabilities may be present in the device due to intentionally left backdoors in the released software or unintentional weaknesses produced by bad design and coding practices. As an example of intentional vulnerability, a programmer might leave a backdoor open in the firmware for debug purpose, allowing outside intruder access to sensitive hardware, which an attacker may use to escalate user's rights and access sensitive information. Examples of unintentional vulnerabilities can be manufacturing defects, programming errors or design flaws. Vulnerabilities are the weakest places in a device. Attackers can exploit these vulnerabilities to interrupt device communications or take the device down completely. Therefore, the impact of a security breach by exploiting an existing vulnerability is very high. Publicly known information-security vulnerabilities and exposures are listed in the Common Vulnerabilities and Exposures (CVE) system.

Table 1 lists some of the vulnerabilities sources and attack methods specific to smart grid IEDs.

Vulnerabilities	Description
Unchanged default passwords	If default passwords are left unchanged, attackers may access the device using default login information available in user guides and manuals
Bad design and coding practices	There are many situations of varying complexity in this category. Among them we mention missing input validation, race conditions, buffer overflows and memory leaks. The attack type depends on the actual vulnerability. e.g: If the input validation is missing, an attacker may enter out of range values that may not be tolerated by the internal design, resulting in a device crash. Memory leaks may be exploited by consuming all memory and leaving the device unusable. In exploitations of buffer overflows, an attacker may write outside the allocated memory and crash the device.
Test and debug features left in the production build	This may allow an attacker to bypass normal login protection and get a higher level of rights to sabotage device functionality. Displaying debug information will reveal device functionality, allowing attackers to understand device internal behaviour.
Unencrypted/ Unsigned sensitive information	Network traffic is exposed to network sniffing, spoofing, replay attacks and Man-in-the -Middle attacks
No limit conditions in incoming packet rates	Packet storm attacks may cause Denial of Service by raising the network service CPU utilization and causing extreme network congestion.
Weak password policies and no limit in number of authentication failed attempts	Weak passwords may be exploited by password guessing, dictionary attacks and brute-force attacks.
Vulnerabilities from third party software components and operating systems (the TCP/IP stack, security stack, operating system)	Third party software may not have been thoroughly tested and may contain vulnerabilities. Once integrated in your device, those vulnerabilities will be transferred to your device.

Table 1 – Examples of vulnerabilities

Attackers use vulnerabilities such as those described above to exploit the system.
Table 2 divides the attacks based on how they are conducted, and presents possible counter measures, which, if implemented, may help defend them.

Attack	Description	Countermeasures
Passive	Passive monitoring of communications sent over public media.	Virtual private networks (VPN), cryptographically protected networks
Active	Attempts to circumvent or break security features, introduce malicious code (e.g. computer viruses), and subvert data or system integrity.	Strong enclave boundary protection (e.g., firewalls and guards), access control based on authenticated identities (ID) for network management interactions, protected remote access, quality security administration, automated virus detection tools, auditing, and intrusion detection.
Close-in	Attacks in which an unauthorized individual gains close physical proximity to networks, systems, or facilities for the purpose of modifying, gathering, or denying access to information. Gaining such proximity is accomplished through clandestine entry, open access, or both.	Security awareness and training; Auditing and intrusion detection; Security policy and enforcement; Specialized access control to critical servers and network elements; Strong authentication and authorization.
Insider	Performed by a person who either is authorized to be within the physical boundaries of the information system or has direct access to the information security processing system	
Distribution	Malicious modification of hardware or software between the time of its production by a developer and its installation, or when it is in transit from one site to another.	Strong in-process configuration control for addressing vulnerability at the factory Use of controlled distribution or by signed software and access control that is verified at the final user site for addressing in transit vulnerabilities.

3.2 Product Vulnerability Assessment

Products may handle several communications protocols to support communications with other devices and services. Depending on their functionality, these protocols may be located at different layers of the OSI communications stack. Figure 1 shows some of the protocols and their positioning in the network stack. When we test these communications protocols for vulnerability assessments, testing has to be carried out at all layers of the network stack that the protocol touches.

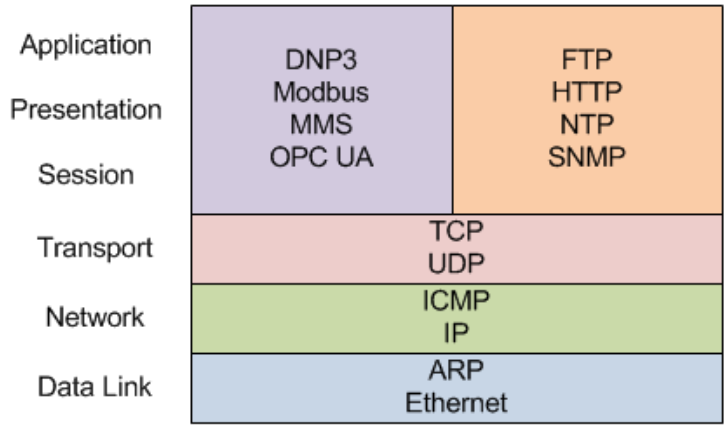


Figure 1: The OSI Stack and Associated Protocols

For our vulnerability assessment we used tools similar with those an attacker might use to exploit product vulnerabilities. As an example, an attacker may use a port scanner to find an open TCP port and then execute a TCP SYN storm attack on that port. In this section, we describe some of the attacking tools that can be used to perform a product vulnerability assessment.

Scans

Scanning tools are one of first tools an attacker uses to gather information about a device in order to exploit it. Scanning tools, such as port scanning, probe a device to find open ports. There are several types of scanning methods that are named after the type of service or protocol they are scanning. Some scanning methods are TCP, UDP, SYN, ACK and FIN scanning. As the names imply TCP scans give a list of TCP ports open in a device, while UDP scans give a list of open UDP ports in a device.

Storms

As the name implies, storms are sending packets in a higher rate to a target device to execute a denial of service attack. In general, the ability of a device to handle packets varies with packet rate. So it is important to execute each storm test case with different rate limits to determine the maximum threshold that the device can handle.

Fuzzers

Fuzz testing is injecting malformed, unexpected, or random data in the device and observing its behaviour under attack. Fuzzer tests generate valid and invalid packets with randomized header values and lengths. In general, fuzzers are randomly choosing fields to fuzz in the protocol. Fuzzing can find a range of critical bugs including input validation failures, memory corruption and infinite loops. Some examples of fuzzing tests are Ethernet Fuzzers, IP Fuzzers and DNP3 Fuzzers.

Grammars

Grammars are another type of fuzz testing. But the difference is that grammars don't choose randomly the fields to fuzz. Grammars iterate over each field and choose fuzz values in a predetermined way, to better serve test coverage. Protocol grammars like DNP3 grammar, manipulate protocol data format at different layers of the protocol stack. Protocol grammars use valid protocol interactions at one layer to transport invalid data to a higher layer. The handler in each layer should validate its input and discard the data if it contains invalid or malformed fields. If the handler does not validate its input, these malformed fields might cause the device to crash or behave abnormally. Grammars are useful for in depth protocol handler analysis, because these handlers may be unprepared for invalid input.

3.3 Attack Simulation Examples

The test setup in Figure 2 was used for the attack simulations described below. The device under test had two redundant Ethernet ports, and we used 100 Mbps Ethernet links. We simulated attacks on the first port and we executed a DNP3 Class poll and "Select Before Operate" control, using DNP3 TCP/IP on the other port. In storm tests, we increment the storm rate (percentage of link utilization) and monitor ability to execute DNP3 TCP/IP poll and SBO controls.

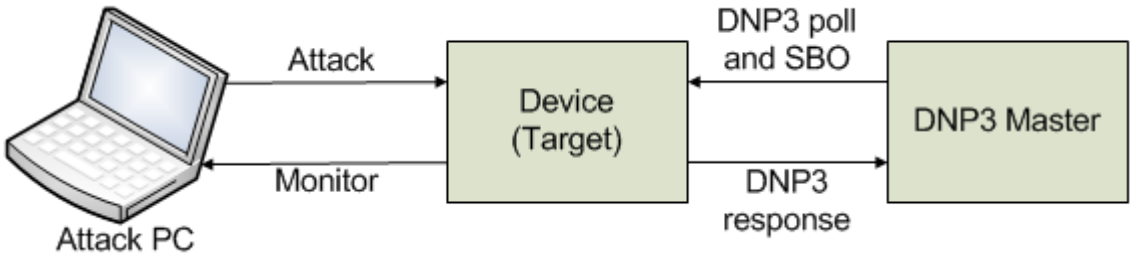


Figure 2: Test setup for attack simulations

The following IP addresses are used in the test setup,

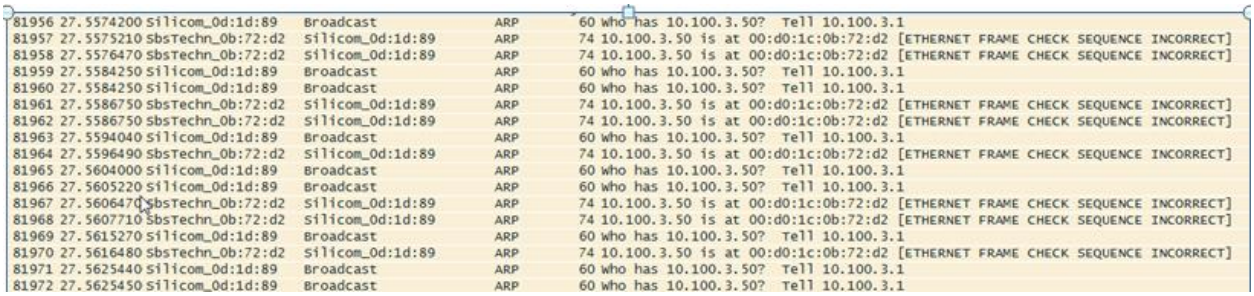
- Attack PC IP address : 10.100.3.100
- Device IP address : 10.100.3.50

3.3.1 ARP Tests

Address Resolution Protocol (ARP) is a data link layer protocol used to map network layer protocol addresses to Ethernet MAC addresses. In other words, this means mapping IP addresses to Ethernet MAC addresses. It uses a memory table, called the ARP cache, to map network layer protocol addresses to data link layer hardware addresses.

3.3.1.1 ARP Request Storm

ARP Request Storm sends identical, valid ARP requests to the device at a chosen packet rate. We changed the packet rate in each test and observed Device behaviour.
As shown in Figure 3, the destination MAC address is set to the global broadcast MAC address (FF:FF:FF:FF:FF:FF) and the destination IP address is set to the device's IP address (10.100.3.50). We send these valid ARP requests in higher packet rate to the target device and examine the device ability to maintain normal DNP3 polls and controls while dealing with the storm.

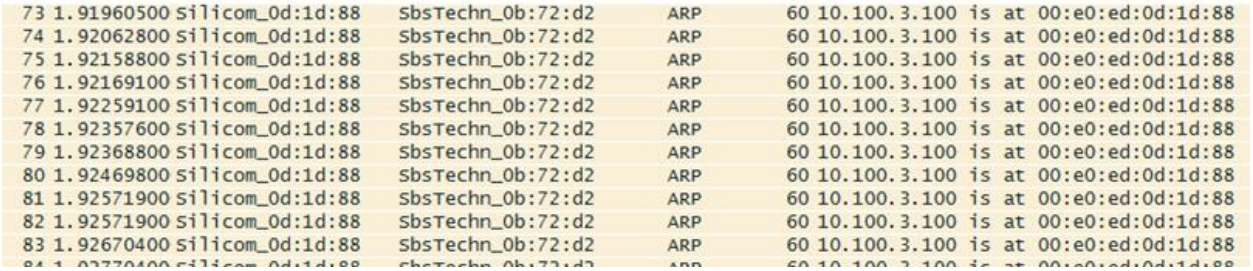


81956	27.5574200	Silicom_0d:1d:89	Broadcast	ARP	60 Who has 10.100.3.50? Tell 10.100.3.1
81957	27.5575210	SbsTechn_0b:72:d2	Silicom_0d:1d:89	ARP	74 10.100.3.50 is at 00:d0:1c:0b:72:d2 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
81958	27.5576470	SbsTechn_0b:72:d2	Silicom_0d:1d:89	ARP	74 10.100.3.50 is at 00:d0:1c:0b:72:d2 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
81959	27.5584250	Silicom_0d:1d:89	Broadcast	ARP	60 Who has 10.100.3.50? Tell 10.100.3.1
81960	27.5584250	Silicom_0d:1d:89	Broadcast	ARP	60 Who has 10.100.3.50? Tell 10.100.3.1
81961	27.5586750	SbsTechn_0b:72:d2	Silicom_0d:1d:89	ARP	74 10.100.3.50 is at 00:d0:1c:0b:72:d2 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
81962	27.5586750	SbsTechn_0b:72:d2	Silicom_0d:1d:89	ARP	74 10.100.3.50 is at 00:d0:1c:0b:72:d2 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
81963	27.5594040	Silicom_0d:1d:89	Broadcast	ARP	60 Who has 10.100.3.50? Tell 10.100.3.1
81964	27.5596490	SbsTechn_0b:72:d2	Silicom_0d:1d:89	ARP	74 10.100.3.50 is at 00:d0:1c:0b:72:d2 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
81965	27.5604000	Silicom_0d:1d:89	Broadcast	ARP	60 Who has 10.100.3.50? Tell 10.100.3.1
81966	27.5605220	Silicom_0d:1d:89	Broadcast	ARP	60 Who has 10.100.3.50? Tell 10.100.3.1
81967	27.5606470	SbsTechn_0b:72:d2	Silicom_0d:1d:89	ARP	74 10.100.3.50 is at 00:d0:1c:0b:72:d2 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
81968	27.5607710	SbsTechn_0b:72:d2	Silicom_0d:1d:89	ARP	74 10.100.3.50 is at 00:d0:1c:0b:72:d2 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
81969	27.5615270	Silicom_0d:1d:89	Broadcast	ARP	60 Who has 10.100.3.50? Tell 10.100.3.1
81970	27.5616480	SbsTechn_0b:72:d2	Silicom_0d:1d:89	ARP	74 10.100.3.50 is at 00:d0:1c:0b:72:d2 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
81971	27.5625440	Silicom_0d:1d:89	Broadcast	ARP	60 Who has 10.100.3.50? Tell 10.100.3.1
81972	27.5625450	Silicom_0d:1d:89	Broadcast	ARP	60 Who has 10.100.3.50? Tell 10.100.3.1

Figure 3: ARP request storm

3.3.1.2 ARP Host Reply Storm

ARP Host Reply Storm sends identical, valid, but unsolicited ARP replies from the attack PC IP address, to the device at a chosen packet rate. We changed the storm rate in each test to analyse device behaviour. As shown in Figure 4, the attack PC (10.100.3.100) sends unsolicited ARP replies to the device in higher packet rate. Attack PC IP address should be in the device's ARP cache already. All ARP replies should process properly and this should not affect the ARP cache. We send these valid ARP replies in higher packet rate to the target device and examine the device ability to maintain normal DNP3 polls and controls while dealing with the storm.



73	1.91960500	Silicom_0d:1d:88	SbsTechn_0b:72:d2	ARP	60 10.100.3.100 is at 00:e0:ed:0d:1d:88
74	1.92062800	Silicom_0d:1d:88	SbsTechn_0b:72:d2	ARP	60 10.100.3.100 is at 00:e0:ed:0d:1d:88
75	1.92158800	Silicom_0d:1d:88	SbsTechn_0b:72:d2	ARP	60 10.100.3.100 is at 00:e0:ed:0d:1d:88
76	1.92169100	Silicom_0d:1d:88	SbsTechn_0b:72:d2	ARP	60 10.100.3.100 is at 00:e0:ed:0d:1d:88
77	1.92259100	Silicom_0d:1d:88	SbsTechn_0b:72:d2	ARP	60 10.100.3.100 is at 00:e0:ed:0d:1d:88
78	1.92357600	Silicom_0d:1d:88	SbsTechn_0b:72:d2	ARP	60 10.100.3.100 is at 00:e0:ed:0d:1d:88
79	1.92368800	Silicom_0d:1d:88	SbsTechn_0b:72:d2	ARP	60 10.100.3.100 is at 00:e0:ed:0d:1d:88
80	1.92469800	Silicom_0d:1d:88	SbsTechn_0b:72:d2	ARP	60 10.100.3.100 is at 00:e0:ed:0d:1d:88
81	1.92571900	Silicom_0d:1d:88	SbsTechn_0b:72:d2	ARP	60 10.100.3.100 is at 00:e0:ed:0d:1d:88
82	1.92571900	Silicom_0d:1d:88	SbsTechn_0b:72:d2	ARP	60 10.100.3.100 is at 00:e0:ed:0d:1d:88
83	1.92670400	Silicom_0d:1d:88	SbsTechn_0b:72:d2	ARP	60 10.100.3.100 is at 00:e0:ed:0d:1d:88

Figure 4: ARP host reply storm

3.3.1.3 ARP Cache Saturation Storm

ARP Cache Saturation Storm sends valid but unsolicited ARP replies from various IP addresses to the device at a specific packet rate in an attempt to saturate the ARP cache. As shown in Figure 5, we send ARP replies with different IP addresses and MAC addresses. We examine the ARP cache of the device, while the device is under attack. Then we increase the storm rate and the time duration of the test to saturate the ARP cache.

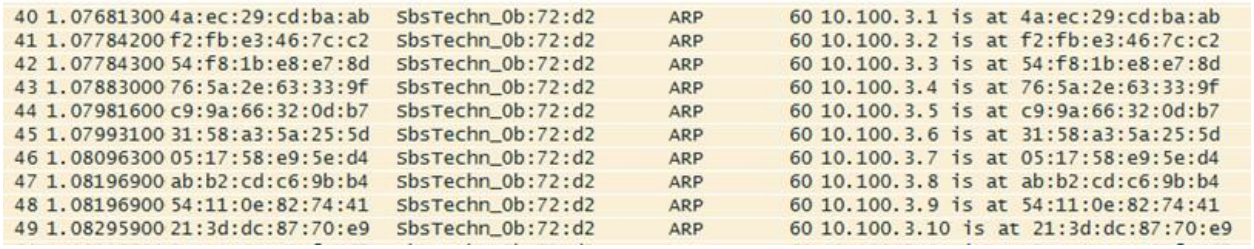


Figure 4: ARP cache saturation

3.3.1.4 ARP Grammar

ARP Grammar sends ARP request packets with valid and invalid header values.
As shown in Figure 5, ARP header fields are manipulated by the grammar, resulting in both valid and invalid values. The following ARP header fields have been manipulated; Hardware Type, Protocol Type, Hardware Size, Protocol Size, Operation, Sender MAC Address, Sender Protocol Address, Target MAC Address, and Target Protocol Address.

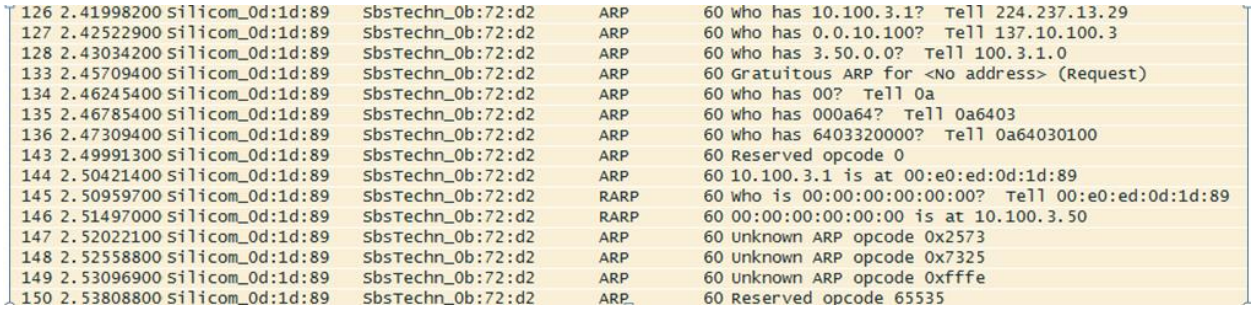


Figure 5: ARP grammar

3.3.2 IP tests

Internet Protocol (IP) is located at the network layer of the communication stack. IP packets are used by many upper layer protocols in their network layer.

3.3.2.1 IP Unicast Storm

IP Unicast Storm sends identical, valid IP packets to the device at a specific packet rate. Each packet contains a null payload. As shown in Figure 6, the source IP address is set to the attack PC IP address and the destination IP address is set to the device IP address.

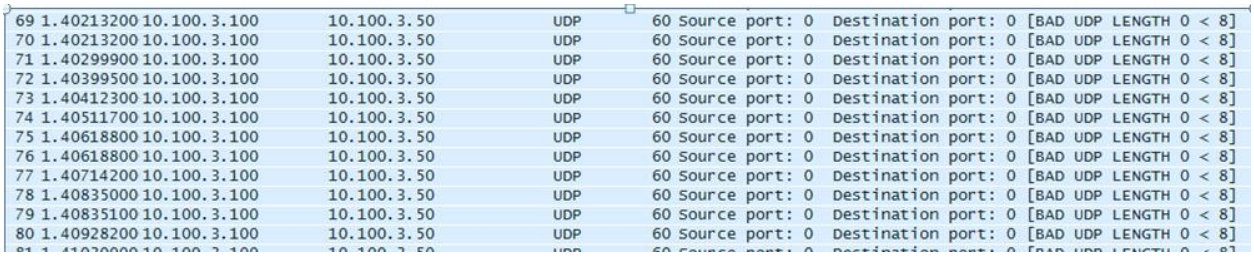
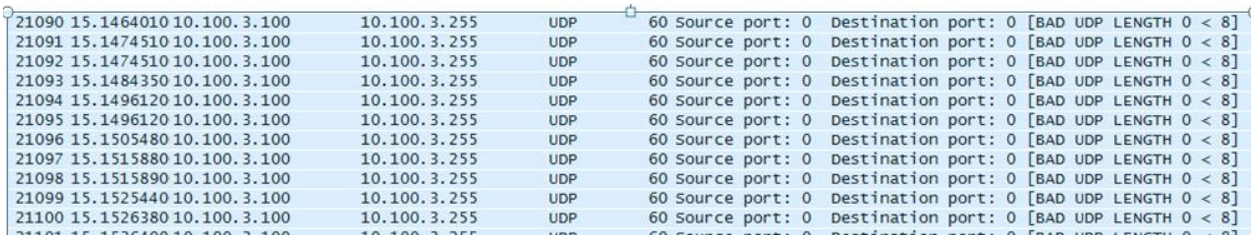


Figure 6: IP unicast storm

3.3.2.2 IP Broadcast Storm

IP Broadcast Storm sends identical, valid IP packets and broadcasts them over the Ethernet link at a specific packet rate. Each packet contains a null payload. As shown in Figure 7, the source IP address is set to the Attack PC IP (10.100.3.100) address and the destination IP address is set to a broadcast IP address (10.100.3.255).

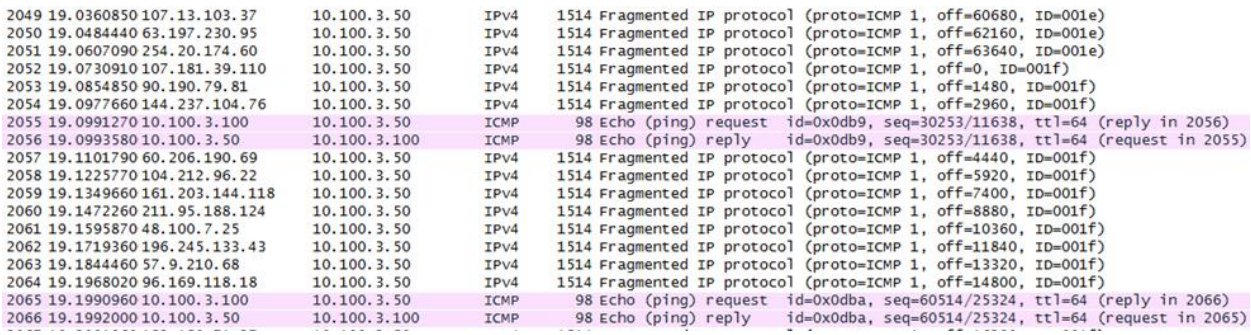


21090	15.1464010	10.100.3.100	10.100.3.255	UDP	60 Source port: 0 Destination port: 0 [BAD UDP LENGTH 0 < 8]
21091	15.1474510	10.100.3.100	10.100.3.255	UDP	60 Source port: 0 Destination port: 0 [BAD UDP LENGTH 0 < 8]
21092	15.1474510	10.100.3.100	10.100.3.255	UDP	60 Source port: 0 Destination port: 0 [BAD UDP LENGTH 0 < 8]
21093	15.1484350	10.100.3.100	10.100.3.255	UDP	60 Source port: 0 Destination port: 0 [BAD UDP LENGTH 0 < 8]
21094	15.1496120	10.100.3.100	10.100.3.255	UDP	60 Source port: 0 Destination port: 0 [BAD UDP LENGTH 0 < 8]
21095	15.1496120	10.100.3.100	10.100.3.255	UDP	60 Source port: 0 Destination port: 0 [BAD UDP LENGTH 0 < 8]
21096	15.1505480	10.100.3.100	10.100.3.255	UDP	60 Source port: 0 Destination port: 0 [BAD UDP LENGTH 0 < 8]
21097	15.1515880	10.100.3.100	10.100.3.255	UDP	60 Source port: 0 Destination port: 0 [BAD UDP LENGTH 0 < 8]
21098	15.1515890	10.100.3.100	10.100.3.255	UDP	60 Source port: 0 Destination port: 0 [BAD UDP LENGTH 0 < 8]
21099	15.1525440	10.100.3.100	10.100.3.255	UDP	60 Source port: 0 Destination port: 0 [BAD UDP LENGTH 0 < 8]
21100	15.1526380	10.100.3.100	10.100.3.255	UDP	60 Source port: 0 Destination port: 0 [BAD UDP LENGTH 0 < 8]

Figure 7: IP broadcast storm

3.3.2.3 IP Fragmented Storm

When an IP packet is larger than the Maximum Transmission Unit (MTU) of the network it is using, the packet has to be broken into smaller blocks of data (fragmentation) for transmission. In the receiver, these packets are reassembled to the original packet prior to further processing. There fragments may arrive out of order, therefore the receiver keeps received fragments in memory until it receives the final fragment. IP Fragmented Storm sends valid, fragmented IP packets and sends them to the device at a specific packet rate. Also in this test, we didn't send the final fragment packet, by forcing receiving device to keep the incomplete packets in memory. Device should discard incomplete fragmented packets from memory to prevent memory and resource exhaustion.



2049	19.0360850	107.13.103.37	10.100.3.50	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=60680, ID=001e)
2050	19.0484440	63.197.230.95	10.100.3.50	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=62160, ID=001e)
2051	19.0607090	254.20.174.60	10.100.3.50	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=63640, ID=001e)
2052	19.0730910	107.181.39.110	10.100.3.50	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=0, ID=001f)
2053	19.0854850	90.190.79.81	10.100.3.50	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=1480, ID=001f)
2054	19.0977660	144.237.104.76	10.100.3.50	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=2960, ID=001f)
2055	19.0991270	10.100.3.100	10.100.3.50	ICMP	98 Echo (ping) request id=0x0db9, seq=30253/11638, ttl=64 (reply in 2056)
2056	19.0993580	10.100.3.50	10.100.3.100	ICMP	98 Echo (ping) reply id=0x0db9, seq=30253/11638, ttl=64 (request in 2055)
2057	19.1101790	60.206.190.69	10.100.3.50	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=4440, ID=001f)
2058	19.1225770	104.212.96.22	10.100.3.50	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=5920, ID=001f)
2059	19.1349660	161.203.144.118	10.100.3.50	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=7400, ID=001f)
2060	19.1472260	211.95.188.124	10.100.3.50	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=8880, ID=001f)
2061	19.1595870	48.100.7.25	10.100.3.50	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=10360, ID=001f)
2062	19.1719360	196.245.133.43	10.100.3.50	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=11840, ID=001f)
2063	19.1844460	57.9.210.68	10.100.3.50	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=13320, ID=001f)
2064	19.1968020	96.169.118.18	10.100.3.50	IPv4	1514 Fragmented IP protocol (proto=ICMP 1, off=14800, ID=001f)
2065	19.1990960	10.100.3.100	10.100.3.50	ICMP	98 Echo (ping) request id=0x0dba, seq=60514/25324, ttl=64 (reply in 2066)
2066	19.1992000	10.100.3.50	10.100.3.100	ICMP	98 Echo (ping) reply id=0x0dba, seq=60514/25324, ttl=64 (request in 2065)

Figure 8: IP fragmented storm

3.3.3 ICMP storm

ICMP Storm generates ICMP packets with different type/code combinations, both valid and invalid, and sends them to the device at a specific packet rate.

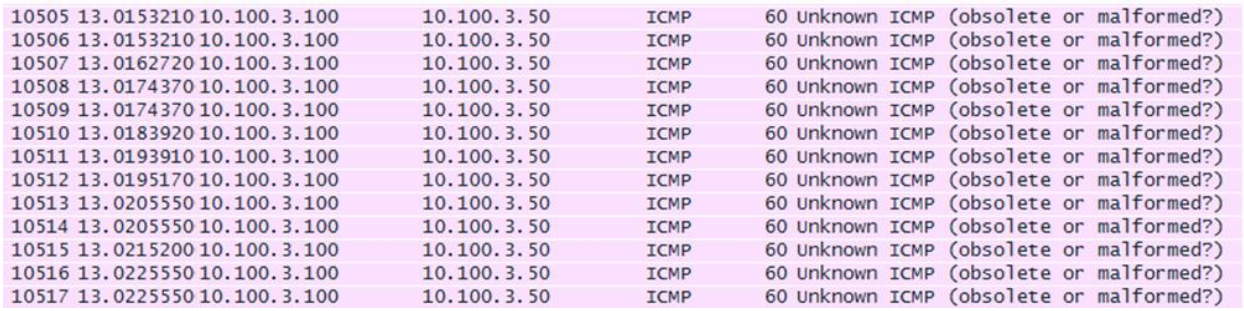


Figure 9: ICMP storm

3.3.4 TCP tests

Transmission Control Protocol (TCP) is located at Transport layer. It provides reliable, ordered, and error-checked delivery of a packet stream. Critical application layers protocols are rely on TCP implementation. These TCP tests are trying to exploit vulnerabilities in TCP.

3.3.4.1 TCP/IP Land attack

LAND attack is a well-known protocol attack that can cause a device to continuously reply to itself. As shown in Figure 10, we generated TCP packets with the same address for the source and destination IP address (10.100.3.50), and the same port for the source and destination TCP port (port 22).

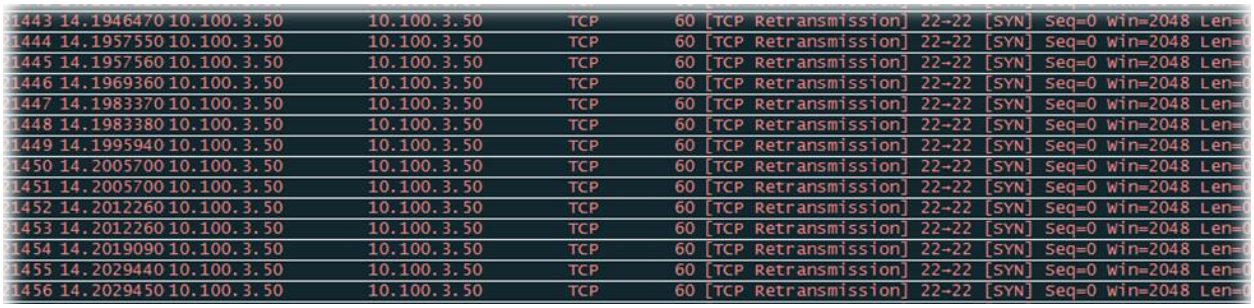


Figure 10: TCP/IP Land attack

3.3.4.2 TCP SYN Storm

TCP SYN Storm generates valid SYN packets and sends them to the device at a varying packet rate to examine the device's ability to maintain normal behaviour. We didn't send any SYN ACK packets. Therefore, TCP connections are not established. In these attack packets, we set the source IP address to an unused IP address and the destination IP address is set to the device IP address.

76	1.42161800	10.100.3.1	10.100.3.50	TCP	60 39277→21 [SYN] Seq=0 Win=5000 Len=0
77	1.42252100	10.100.3.1	10.100.3.50	TCP	60 40218→21 [SYN] Seq=0 Win=5000 Len=0
78	1.42370900	10.100.3.1	10.100.3.50	TCP	60 37883→21 [SYN] Seq=0 Win=5000 Len=0
79	1.42371000	10.100.3.1	10.100.3.50	TCP	60 46546→21 [SYN] Seq=0 Win=5000 Len=0
80	1.42462900	10.100.3.1	10.100.3.50	TCP	60 37915→21 [SYN] Seq=0 Win=5000 Len=0
81	1.42564300	10.100.3.1	10.100.3.50	TCP	60 45619→21 [SYN] Seq=0 Win=5000 Len=0
82	1.42574800	10.100.3.1	10.100.3.50	TCP	60 41782→21 [SYN] Seq=0 Win=5000 Len=0
83	1.42676100	10.100.3.1	10.100.3.50	TCP	60 38396→21 [SYN] Seq=0 Win=5000 Len=0
84	1.42787000	10.100.3.1	10.100.3.50	TCP	60 41282→21 [SYN] Seq=0 Win=5000 Len=0
85	1.42787100	10.100.3.1	10.100.3.50	TCP	60 47028→21 [SYN] Seq=0 Win=5000 Len=0
86	1.42877000	10.100.3.1	10.100.3.50	TCP	60 43592→21 [SYN] Seq=0 Win=5000 Len=0
87	1.42975600	10.100.3.1	10.100.3.50	TCP	60 43067→21 [SYN] Seq=0 Win=5000 Len=0
88	1.42989900	10.100.3.1	10.100.3.50	TCP	60 43224→21 [SYN] Seq=0 Win=5000 Len=0
89	1.43090200	10.100.3.1	10.100.3.50	TCP	60 40423→21 [SYN] Seq=0 Win=5000 Len=0

Figure 11: TCP SYN storm

During the TCP SYN storm, the DNP3 slave stops responding to the DNP3 master requests. When the device is under attack, the device should reject unnecessary TCP SYN request. If the device doesn't discard unnecessary SYN requests, the device resources (CPU) are fully consumed by network services. Unavailability of CPU resources to the DNP3 task causes DNP3 slave to stop responding to master requests.

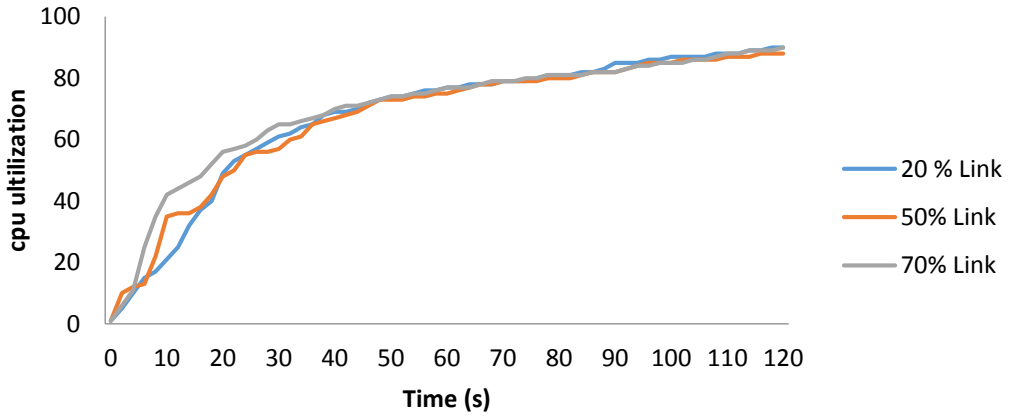


Figure 12: CPU utilization on device during a TCP SYN storm

3.3.5 Open Port attack

Open ports are possible entries into the device. Behind open ports, there are applications and services listening for connections from the outside world. CIP-007-5 System Security Management advises that ports should be open only if they are required and in use. Such required TCP ports may be SSH, DNP3 TCP, Modbus TCP, HTTP and required UDP ports may be NTP, SNMP, DNP3/UDP.

Each open port may be the target of denial of service (DoS) attacks. Some applications and services have protection mechanisms implemented to withstand DoS attacks to a certain extent. But other applications may not have such defence, causing the device to go down when sufficient stress is put on it. But even robust protocol implementations, with proper discard mechanisms for invalid and suspicious requests, may still not be enough. The security community has identified a

list of ports commonly used by malware for attacks and they named this list of ports the “Trojan ports”.

By performing active port scanning and banner grabbing, one can determine what ports are open. This analysis gives instant visibility into the security of the device from the outsider’s perspective. Figure 13 shows a TCP scan that tries to find opened TCP ports. Figure 14 shows a UDP scan that tries to find open UDP ports and increments the destination port in each UDP packet.

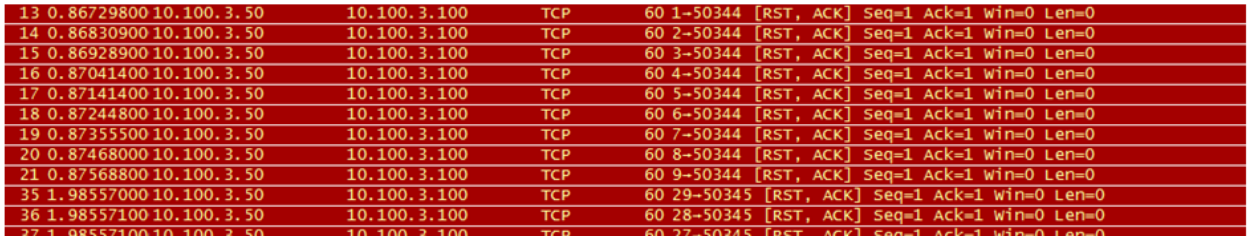


Figure 13: TCP port scan

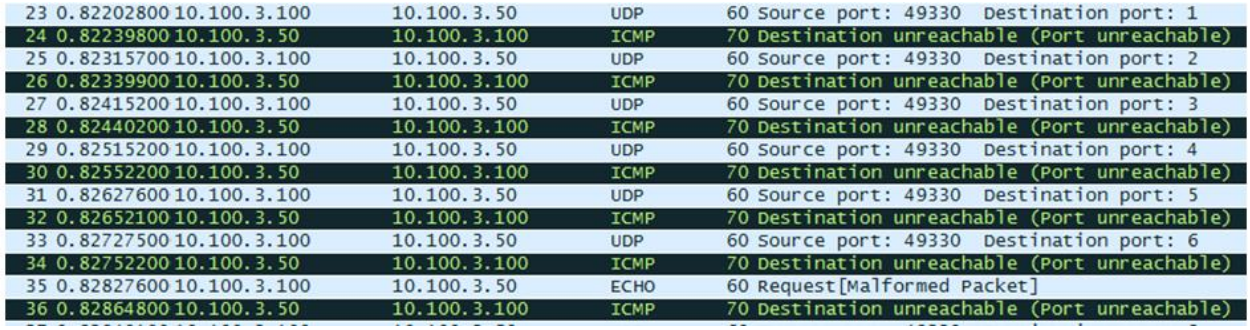


Figure 14: UDP port scan

When an attacker finds an open TCP or UDP port, the attacker may send a storm to that specific port. The port may not be even used for an active service in the device. This attack may cause system instability and may even crash the whole device. This may interrupt all communications on the device, including DNP3 polls and controls. Thus, we can see that an attacker can perform successful denial of service attacks on any protocols, without even targeting the specific port of interest.

3.4 DNP3 Grammar

Distributed Network Protocol (DNP3) is a master/slave protocol. The DNP3 fuzzer can run on master or client. We simulate DNP3 fuzzer on the client. As shown in Figure 15, DNP3 fuzzer acts as the master, which generates malformed DNP3 packets and sends them to the client.

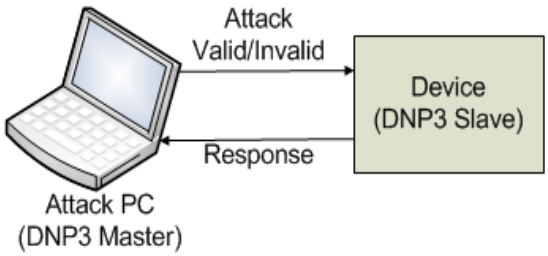


Figure 15: Setup for testing DNP3 grammar

DNP3 protocol stack consists of Datalink Layer, Transport Layer, Application Layer and User Layer. DNP 3 data link layer adds addressing and error detection prior to sending frame over the physical layer. See Figure 16 for a full description of the DNP3 packet. DNP3 was originally a serial protocol, and it was later developed to be transported over TCP and UDP.

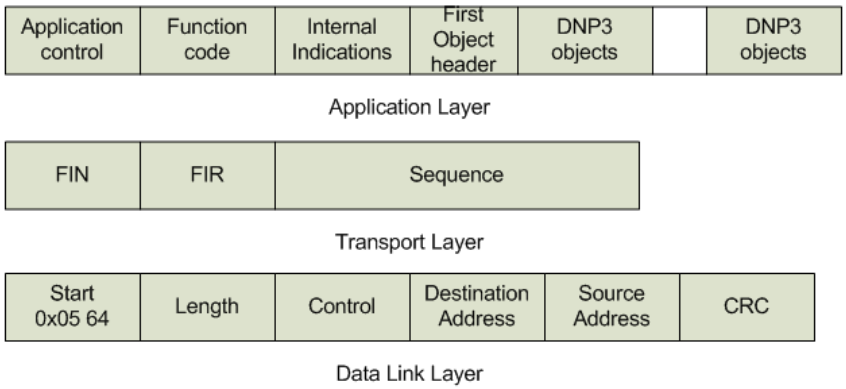


Figure 16: DNP3 packet

Some of the DNP3 requests we used for DNP3 fuzzing are:

1. DNP Read Requests - Masters send DNP3 Read messages to request information from an outstation. For Read messages, the function code is 01 and the payload contains Read Data Objects.
2. DNP select and operate requests - Masters use Select message to instruct an outstation to select data points, as directed by the data objects in the message; an Operate message instructs an outstation to activate those points. Selected points are not activated until the outstation receives a corresponding Operate message from the master.
3. DNP write requests - Masters send DNP3 Write requests to instruct an outstation to store the information contained in the message. For Write messages, the function code is 02 and the payload contains Write Data Objects.

In the above requests, we manipulated the fields in datalink, transport and application layers. Figure 17 shows the methods we use to manipulate DNP3 packets. We used DNP3 over TCP to attack the target. And we used invalid values in those fields to generate malformed packets. Invalid values included values outside of the acceptable range specified by the DNP3 protocol specification and values known to cause problems for receivers (NULL characters). Receiver should reject these malformed packets and only accept packet with valid values.

In the Figure 18 we changed the Datalink start bytes to 0x4141, instead of 0x0564, which is the correct value. In Figure 19 we generated a packet with invalid datalink layer control. In Figure 20, target device received a DNP3 packet from a different source than it was configured to. Figure 21 shows an example of invalid CRC and Figure 22 shows an example of an invalid application layer function code. In Figure 23, we are sending an operate request with bogus number of objects to the target. Target device should correctly validate DNP3 packets according to the protocol specification and reject DNP3 packets that contain values outside of the acceptable range.

We remove values or fields from the protocol structure to create packets with empty fields. Receivers expect data to appear in the order specified by the protocol standard. By sending these empty packets we cause the receiver to be unable to recognize the packet type, process data. And we monitor how the receiver handling these DNP 3 packets with missing fields.

We can create overflow attacks by inserting input with a large number of random bytes in an effort to cause the data to exceed the boundaries of its specified location. Receivers should correctly validate integer calculations and field lengths to prevent overflow data to execute arbitrary code.

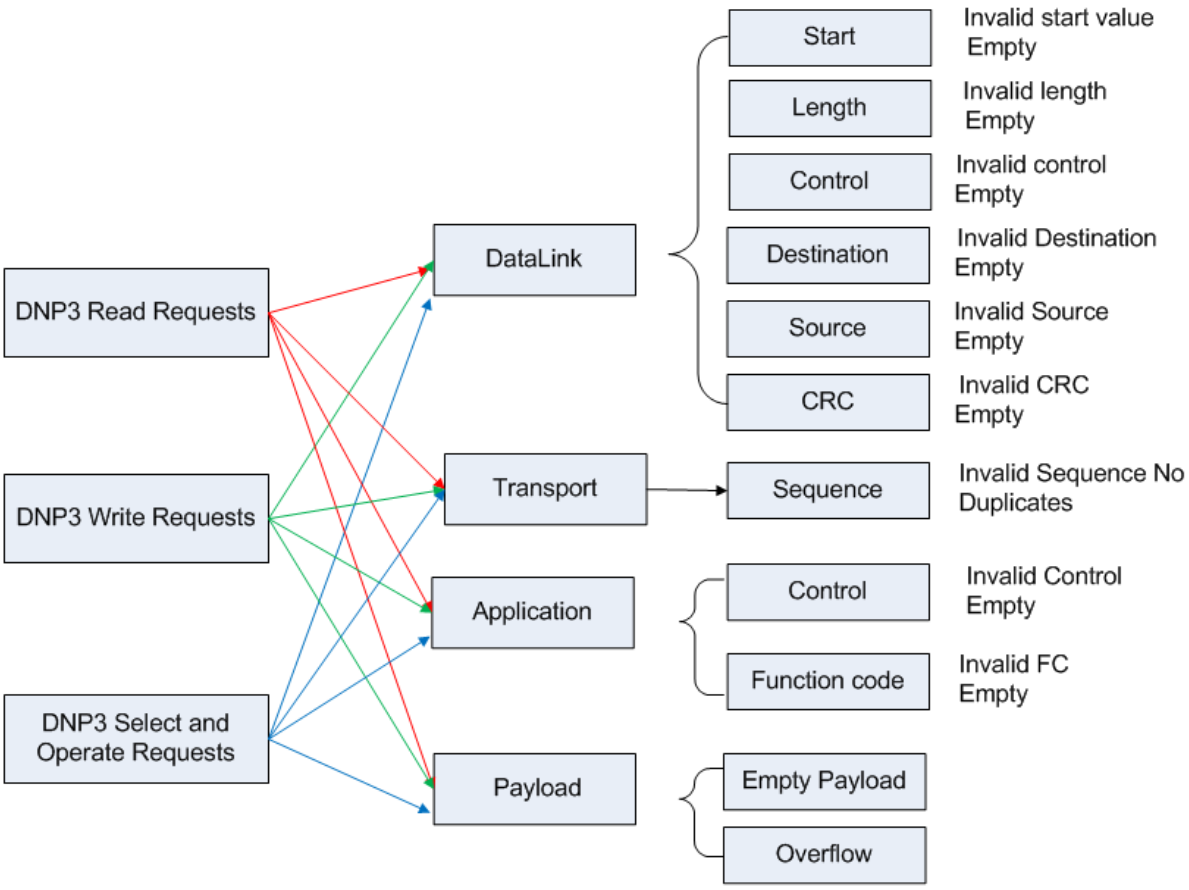


Figure 17 DNP3 Request Fuzzing

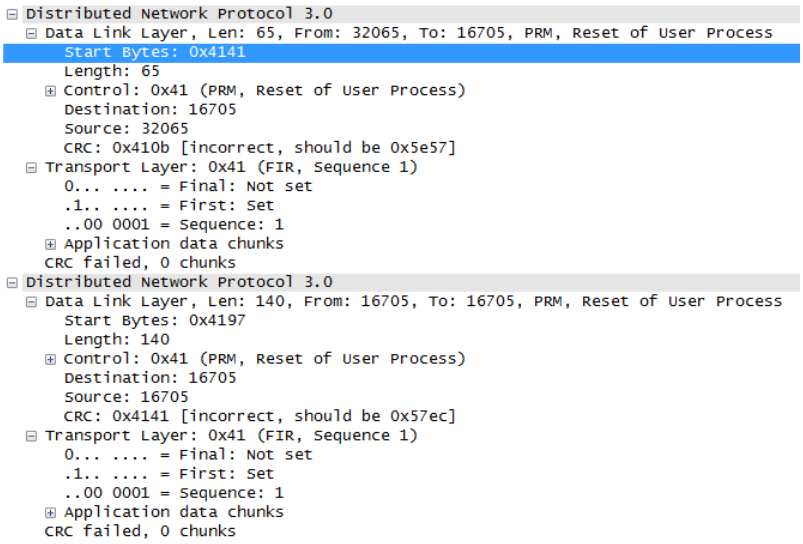


Figure 18: Invalid start value

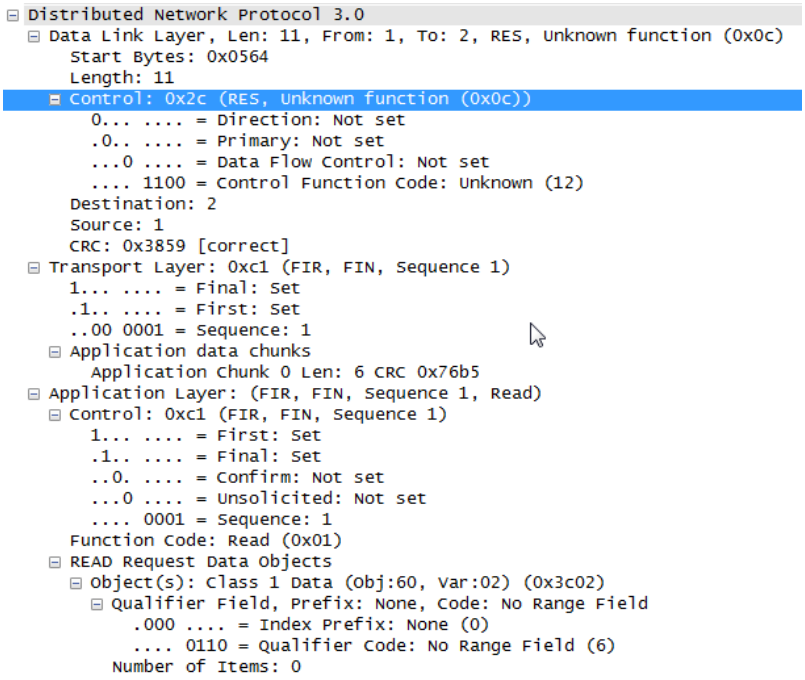


Figure 19: Invalid datalink control

```
Distributed Network Protocol 3.0
  Data Link Layer, Len: 11, From: 65312, To: 2, DIR, PRM, Unconfirmed User Data
      Start Bytes: 0x0564
      Length: 11
    Control: 0xc4 (DIR, PRM, Unconfirmed User Data)
        1... .... = Direction: Set
        .1.. .... = Primary: Set
        ..0. .... = Frame Count Bit: Not set
        ...0 .... = Frame Count Valid: Not set
        .... 0100 = Control Function Code: Unconfirmed User Data (4)
      Destination: 2
      Source: 65312
      CRC: 0x6ed1 [correct]
  Transport Layer: 0xc1 (FIR, FIN, Sequence 1)
  Application Layer: (FIR, FIN, Sequence 1, Read)
```

Figure 20: Invalid source

```
Distributed Network Protocol 3.0
  Data Link Layer, Len: 16, From: 1, To: 2, DIR, PRM, Unconfirmed User Data
      Start Bytes: 0x0564
      Length: 16
    Control: 0xc4 (DIR, PRM, Unconfirmed User Data)
        1... .... = Direction: Set
        .1.. .... = Primary: Set
        ..0. .... = Frame Count Bit: Not set
        ...0 .... = Frame Count Valid: Not set
        .... 0100 = Control Function Code: Unconfirmed User Data (4)
      Destination: 2
      Source: 1
      CRC: 0x0000 [incorrect, should be 0x55ce]
  Transport Layer: 0xc1 (FIR, FIN, Sequence 1)
      1... .... = Final: Set
      .1.. .... = First: Set
      ..00 0001 = Sequence: 1
    Application data chunks
      Application Chunk 0 Len: 11 Bad CRC got 0x010d expected 0xd89e
    CRC failed, 0 chunks
```

Figure 21: Invalid CRC

```
Distributed Network Protocol 3.0
  Data Link Layer, Len: 10, From: 1, To: 2, DIR, PRM, Unconfirmed User Data
      Start Bytes: 0x0564
      Length: 10
    Control: 0xc4 (DIR, PRM, Unconfirmed User Data)
      Destination: 2
      Source: 1
      CRC: 0x9164 [correct]
  Transport Layer: 0xf8 (FIR, FIN, Sequence 56)
      1... .... = Final: Set
      .1.. .... = First: Set
      ..11 1000 = Sequence: 56
    Application data chunks
  Application Layer: (FIR, Sequence 7, Unknown function (0xbf))
    Control: 0x87 (FIR, Sequence 7)
        1... .... = First: Set
        .0.. .... = Final: Not set
        ..0. .... = Confirm: Not set
        ...0 .... = Unsolicited: Not set
        .... 0111 = Sequence: 7
      Function Code: Unknown function (0xbf) (0xbf)
```

Figure 22: Unknown function code

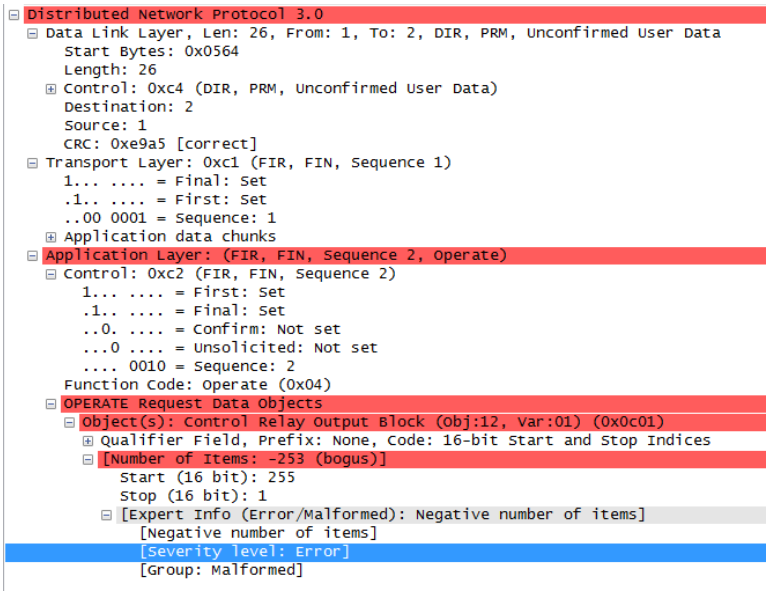


Figure 23: Invalid number of items

4 SECURITY TESTING AND REGULATORY STANDARDS

Cyber security regulatory standards impose compliance to the requirements they define.

Examples of regulatory standards are NERC CIP [1] and IEEE 1686-2013 Standard for Intelligent Electronic Devices Cyber Security Capabilities [2].

If the vendor of an IED claims compliance with a security regulatory standard, that vendor should ensure that all the requirements that apply to the IED have been system tested and they work. Test cases must be designed for each of these requirements and run as part of the security system testing.

For instance NERC CIP 5, section 004, requirement 4.1, which refers to the access management program, requires a "*process to authorize based on need, as determined by the Responsible Entity*". This requirement applies to IEDs. An IED may implement a role based access control scheme (RBAC) that would ensure authorization is given to authenticated users only as defined by the role associated with that user. For example an operator would be allowed to execute commands, but not to change the configuration of the IED. If this feature is implemented and it is verified as working during system testing, then requirement 4.1 from NERC CIP 5 section 004 is met.

Another example from NERC CIP 5 is section 007, requirement 4.1 regarding Security Event Monitoring, which requires to "*log events at the BES Cyber System level (per BES Cyber System capability) or at the Cyber Asset level (per Cyber Asset capability) for identification of, and after-the-fact investigations of Cyber Security Incidents*".

This requirement applies to IEDs. An IED may meet this requirement if it implements all the security events specified in section 007, requirement 4 and, during system testing, it is proved that the events are all properly issued and logged.

One interesting example from NERC CIP 5 refers to open ports. Section 007, requirement 1.1, specifies that devices should, where technically feasible: "*enable only logical network accessible ports that have been determined to be needed by the Responsible Entity*". This requirement shows that NERC CIP recognises that open ports raise a device's vulnerability.

We can see then how important the system security testing step is, as it takes care of verifying all aspects of security on the IED. Security features, vulnerabilities and compliance with regulatory standards are all supposed to be covered during the IED system testing.
But for this to happen, a proper, well thought testing specification must be produced in advance of testing, the specification must be reviewed by the security architect and the design leaders and and, after that, its test cases run. All these steps are part of the process called software development process, which we will talk about in the next section.

5 THE SOFTWARE DEVELOPMENT PROCESS

The software development process is a set of correlated activities involved in creating and maintaining a software product. IEDs contain embedded software, which must be developed following a strict software development process, in order for quality to be guaranteed.
There are many approaches to software project management, known as software development life cycle models. Among the most known models is the traditional waterfall and the agile software development. However, independent of the model used, the following stages will always be part of the software development:

1. Problem analysis
2. Drawing the requirements
3. Design
4. Implementation
5. Testing
6. Deployment
7. Maintenance

The first two steps do not bring in any additional security vector, unless the feature under design is itself a security feature. Researchers analyze the problem, study the market and write the requirements in a functional specification. The document is reviewed and the design process is started.

From the design point on however the security vector must be taken into account, even if the feature under design is not directly related to cyber security.
For instance multiple processes accessing a common resource must ensure the use of semaphores or else, in certain circumstance, the system may be locked permanently, which would manifest like a denial of service attack, or the information read may not be accurate.
Another typical example is related to memory management. An improper memory management design could manifest as reading or writing out of boundaries, which may lead to system crashes, but may also be exploited to leak important information outside the system.
And as important, the event handling mechanism must be carefully designed. Events have usually context information associated with them, which contains pointers to data that the event needs to access. If this access is not designed properly, chances are that the data may be released without knowing that there are still events accessing it and this may lead to unexpected behavior or system crashes.
The design phase must be ended with a through design review and the documentation stored on a secure source control server.

The implementation phase is also prone to introducing security issues/vulnerabilities.
Developers should be trained in secure coding practices to ensure best quality coding. Among the secure coding practices I will mention the need to initialize all the variables and to validate inputs. Uninitialized variable and non-validated inputs are some of the most common coding vulnerabilities.
OWASP [3] has compiled a reference guide for secure coding practices, which applies for any coding language and software.
Once a portion of the code is written and compiled, peer code review is essential as is unit testing done by the developer who wrote that part/module of code and integration testing with the other related modules, done by all involved developers.
Next phase is the secure validation testing. Looking at the image in Fig 24 we can get a better understanding of the important role this phase has. There is a high probability that vulnerabilities have been introduced during design and implementation. Some may have been discovered and rectified during design and code review or during unit and integration testing. But the last chance before deployment is at this point, during the validation testing. Beyond this point, the product is released and the vulnerability management is harder and more painful, as it may involve unhappy customers and lost revenue.

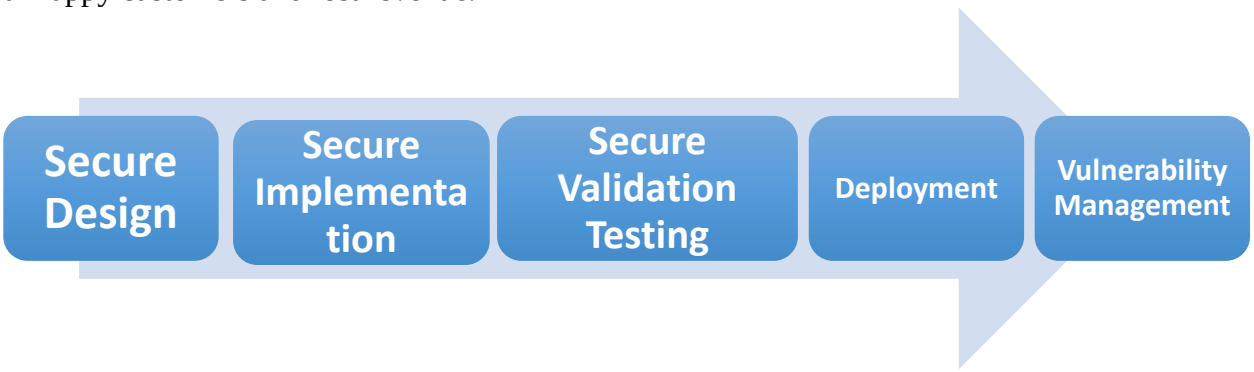


Figure 24: Software Development Process

6 CONCLUSSION

This paper demonstrated the importance of thoroughly testing the robustness of IEDs from the security point of view and it explored the various types of vulnerability tests to be considered while showing some of the testing methods available.

The paper emphasized the importance of testing not only the security features deliberately designed and implemented in the product, but also testing for software vulnerabilities potentially introduced during the software development process. It discussed aspects such as vulnerability exploitations, the importance of early vulnerability detection, ways in which the security testing helps towards compliance with regulatory standards and the importance of designing and coding with security in mind.

7 BIBLIOGRAPHY

[1] NERC CIP version 5, http://www.nerc.com/pa/Stand/Pages/CIPStandards.aspx
[2] IEEE 1686-2013: http://standards.ieee.org/findstds/standard/1686-2013.html

[3] OWASP The Open Web Application Security project
https://www.owasp.org/images/0/08/OWASP_SCP_Quick_Reference_Guide_v2.pdf

8 BIOGRAPHIES

Pubudu Eroshan Weerathunga is a firmware Engineer for Substation Automation Systems at GE Grid Solution. Eroshan has implemented several SCADA communication protocols and his work is mainly focused on cyber security of Substation Automation products. He received his B.Sc. in Electronics and Telecommunication Engineering from University of Moratuwa, Sri Lanka in 2009 and his M.E.Sc in Electrical and Computer Engineering from Western University, Canada in 2012.

Anca Cioraca is an Information Technology Professional with over twenty years hands on experience in system and software architecture, specialized in communications, networking and cybersecurity.
Anca has a Master of Engineering degree in Electronics and Telecommunications from Bucharest Polytechnic University, Romania. In 1991 Anca moved to Canada and for the following twenty years she focused on software architecture and cyber security for network devices, such as routers, firewalls and security servers, while working for Motorola, Enterasys, Siemens and WatchGuard. In 2012 Anca joined GE Grid Solutions. Currently Anca leads the cyber security architecture for next generation GE Grid Solutions Grid Automation products.
Anca is a member of IEEE Communications Society and the IEC TC57 working group WG15, where she actively contributes to the definition of security requirements for the TC 57 series of protocols.