Towards an Automated Attack Tree Generator for the IoT

by

Carlos Caldera

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2017

Author . . . .
Department of Electrical Engineering and Computer Science
September, 5 2017

Certified by . . . .
Howard Shrobe
Principal Research Scientist
Thesis Supervisor

Accepted by . . . .
Christopher J. Terman
Chairman, Masters of Engineering Thesis Committee

Towards an Automated Attack Tree Generator for the IoT

by

Carlos Caldera

Submitted to the Department of Electrical Engineering and Computer Science
on September, 5 2017, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The growing frequency and scale of cyber security attacks is daunting. Notable areas of concern are the Internet of Things (IoT) and Operational Technology (OT) systems; the IoT is becoming intimately integrated into our lives, and the physical repercussions of attacks on OT systems can be devastating. Risk analysis tools can prove to be very helpful towards defining counter measures that can either prevent or dampen the effect of these seemingly inevitable cyber security attacks. One such tool, attack trees, provide a formal way of describing the varying attacks that could be mounted against a system. Though they are limited because their development is time intensive, work has been done on automating this process with attack tree generators. In this thesis, we provide suggested design modifications to be made on existing attack tree generators to work on IoT and OT systems.

Thesis Supervisor: Howard Shrobe
Title: Principal Research Scientist

Acknowledgments

I would like to sincerely thank my supervisor Dr. Howard Shrobe for the wonderful opportunity to work with him towards this project, as well as for his invaluable guidance and support throughout. I would also like to extend my deepest gratitude to my colleauge Gregory Falco; without his insight and encouragement this would not have been possible.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

1.1 Problem Description

Cyber security attacks[1] are becoming the rule rather than the exception. In fact, the daily number of such attacks on US companies more than tripled from 50 to 160 between 2010 and 2015 [26], and every second, 12 people become a victim of cybercrime [56]. To make matters worse, the severity of these attacks has also been on the rise: it is estimated that global annual cybercrime costs will grow from \$3 trillion in 2015 to \$6 trillion by 2021 [38]. Clearly, our defenses have to be sharpened to counter this slew of attacks.

All cyber security defense methodologies fall under one of three distinct phases: preventing a cyber attack before it can happen (prevention); detecting that a cyber attack occurred, or is occurring (detection); or responding to a cyber attack after it occurred, or while it's occurring (response) [30]. A popular technique to use towards the prevention phase is risk analysis – identifying and assessing all factors that may jeapordize something's success. This technique helps define preventative measures to reduce the probability of these factors occurring, and identifies useful countermeasures to deal with those factors in the case that they do develop [43].

One tool that is very useful towards risk analysis is an attack tree; it provides a

[1]We generalize the term "attack" to include any and all types of incidents, breaches, crimes, or other related activities that are carried out by means of computers or the Internet.

formal, systematic way of describing the varying attacks that could be dealt towards a system [50]. Attack trees can be used to itemize all of a system's attack vectors — the means by which an adversary can compromise a system — to give anyone, not just security experts, a glance at a system's weak points. One of their major drawbacks, however, is that developing attack trees manually is a time consuming process, since someone with deep knowledge of the system in question has to spend a potentially large number of hours thinking through every possible way an attacker can compromise their system.

Because of the substantial benefit they can provide for the cyber security community, though, there has been work done on mitigating this limitation by automating the development process via attack tree generators. An example of such a system was built by Dr. Howard Shrobe at the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL) and worked on the lab computing environment at the time (2001) [53]; this is what we will focus on in this thesis.

1.2 Motivation

The aforementioned growth in cyber security attacks becomes even more ominous when one considers the so-called "Internet of Things" (IoT) and how prevalent these technologies have become: IoT devices numbered 3.9 billion in 2013 and are expected to grow to nearly 21 billion by the year 2020 [12]. This rather astonishing number of projected IoT devices will only add to the number of cyber security attacks that occur, and may also contribute to their severity.

Another critical domain that's been getting more attention recently are Operational Technology (OT) systems, the control systems that are typically used in industries such as electrical, water, oil, gas, and nuclear. Attacks to these types of systems could have destructive physical repurcussions — think, for example, what could happen if a nuclear power plant malfunctioned due to an attack — that could be difficult, if not impossible to fully recover from. What's more, the number of cyber attacks against OT systems in 2016 increased over 110% relative to the numbers in

2015, with nearly 90% occuring in the United States [34]; this should be cause for alarm.

1.3 Contribution and Outline

Because we believe that protecting IoT and OT systems from cyber attacks is of paramount importance, and because of the complexity of these technologies, we see these as strong use cases for an automated attack tree generator. To that end, in this thesis, we offer modifications to be made to Dr. Shrobe's attack tree generator to work with IoT and OT systems. Moreoever, because IT systems have changed since the original development of the attack tree generator (2001), modifications to reflect these new systems should also be made; these changes will be discussed at length. Our methodology involves conducting a gap analysis by identifying the main system components of the mentioned spaces that aren't addressed by the current generator.

The rest of this paper will be organized as follows. Chapter 2 covers the relevant background: namely, it will provide context for IoT and OT systems, and attack trees. Chapter 3 details the original attack tree generator, and provides the suggested design modifications for IoT and OT systems, as well as updates for IT systems. Chapter 4 describes a sample IoT system to test our design modifications against. Chapter 5 proposes future work to be done towards the attack tree generator, and Chapter 6 concludes.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Background

2.1 Internet of Things

2.1.1 Intro

The Internet of Things (IoT) is a device networking paradigm that can generally be defined as “a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies” [51]. In an Internet of Things, the devices are able to perceive their context, communicate with each other, access Internet services, and interact with people — providing truly ubiquitous computing [33]. These IoT devices, many of which have donned the prefix “smart” (e.g., smart phones, smart TVs, smart cars), have undoubtedly made their presence known in our world; the number of IoT devices in 2016 reached 6.4 billion [12].

This should not come as a surprise considering the benefits they can provide in domains like transportation and logistics (e.g., assisted driving for increased safety), healthcare (e.g., smart sensors on patients for improved diagnoses), smart environments (e.g., smart thermostats for better home comfort), and personal and social (e.g., personal assistants like Amazon’s Alexa or Apple’s Siri) [3]. Moreover, the economic benefits IoT devices bring companies is manifold — the IoT market is expected to rise from $130 billion in 2015 to $884 billion by 2022 [28] — so it’s no surprise that the IoT is here to stay (and grow).

The IoT is not a single novel technology, but rather the result of synergy between several complementary technical developments that provide capabilities which taken together help bridge the gap between the virtual and physical world [33]. These capabilities include: communication infrastructure and protocols, the ability to be uniquely identifiable, the ability to collect information about their surroundings, and embedded information processing. A brief overview of the various enabling technologies in the IoT is given in Table A.1

2.1.2 History

The term “Internet of Things” was coined in 1999 by Kevin Ashton of Procter & Gamble [2], and was popularized by the work of the MIT’s Auto-ID Center, which in that same year started to design and propagate a cross-company Radio Frequency Identification (RFID) infrastructure [33]. By 2003, IoT was mentioned in main-stream publications like The Guardian, Scientific American, and the Boston Globe [44]. Not too long after that, the term “Internet of Things” spread rapidly; starting in 2004, it started to blow up as a Google search trend; in 2005 it could already be found in some book titles, and in 2008 the first scientific conference was held in this research area [33, 24].

According to the Cisco Internet Business Solutions Group (IBSG), though, the Internet of Things wasn’t born per se until sometime between 2008 and 2009 – at the point in time when more “things” were connected to the Internet than people [44]. Moreoever, in that same year (2008), the US National Intelligence Council included IoT in the list of six “Disruptive Civil Technologies” with potential impacts on US national power out to 2025 [24]. Since then, IoT was added to the annual Gartner Hype Cycle — a graphic that tracks technology life-cycles from “technology trigger” to “plateau of productivity” — in 2011, and has hit the Hype Cycle’s “Peak of Inflated Expectations” in 2014 [44]. Currently, we are still seeing the effects of the huge hype behind IoT, as our smart devices keep getting smarter.

2.1.3 The IoT Model

Because of the large scope that's covered by the IoT, work was needed to be done towards providing a standardized model for understanding and compartmentalizing IoT sytems. In an IoT system, data is generated by various types of devices, processed in many ways, transmitted to all sorts of locations, and acted upon by a myriad of applications. So, there was need for a unifying model that can capture all this complexity. In 2014 Cisco proposed the "IoT Reference Model" [8], which can be seen in Figure B-1.

The reference model starts at its first layer with the physical endpoints, the "things" in the IoT, that send and recieve information. This list includes all the "smart" devices that range in size and complexity from smart phones and smart cars. Other devices like sensors, actuators, and microcontrollers and microprocessors are also part of this layer.

The second layer, the connectivity layer, includes the crucial communication technologies for the IoT that allow for the wireless (or, less impressively, wired) communication amongst devices and between different networks across short-to-long distances. The two most extensively used of these technologies are the familiar Bluetooth — this includes its conservative counterpart, Bluetooth Low Energy (BLE) — and ZigBee communication protocols [36]. BLE is a low-power, short-range networking protocol that's suitable for small data chunk transfers, and has a communication range of up to 150m; it is widely used in smart phones and other mobile devices [9]. ZigBee also offers low-power and has a short range (up to 100m), but also offers high security, robustness and high scalability; it is gaining much traction in machine-to-machine (M2M) applications [9].

The edge, or fog, computing layer addresses the need to convert the large volumes of data traffic into a suitable format for storage or processing at higher levels of the stack. So, this layer focuses on high-volume data analysis and transformation, since a tenet of the IoT Reference Model is that intelligent systems initiate information processing as early and as close to the edge of the network as possible [8]. Examples of processes that occur at this layer are: data filtering, packet content inspection, thresholding, and event generation.

The remaining layers can, for our purposes, be consolidated into a single application layer. This is where there will be intersections between the IoT system and the supporting applications, services, and protocols. In devices connecting to the cloud, for example, one would expect to see REST APIs and/or WebSockets, and device management and interfaces can be handled through services, such as Telnet and SSH. Other common application protocols used in IoT are Message Queueing Telemetry Transport (MQTT), a pub/sub messaging protocol designed for M2M communications used in measuring and monitoring applications, and Constrained Application Protocol (CoAP), a web transfer protocol designed as a compressed replica of HTTP.

2.1.4 Security

Context

The hype IoT has garnered over the past years has caused production for IoT devices to have surged in order to catch up with the huge customer demand. Because of this production rush, and since there is little to no manufacturer regulation, IoT device security has greatly suffered because manufacturers will produce insecure devices to save on costs [28]. This proliferation of (vulnerable) IoT devices has caused it to be a target for adversaries to invest their time and resources into developing exploits for IoT devices; a single exploit could yield a horde of IoT devices, a botnet, under their control [28]. These botnets can be used, among other things, to effectively mount a Distributed Denial-of-Service (DDoS) attack against most any target: Mirai, an infamous botnet strain, took over enough IoT devices to generate 620 Gbps of traffic, taking down KrebsOnSecurity.com [29]; not long after, a botnet of an estimated 100,000 devices DDoS'd Dyn, a company that controls much of the Internet's DNS infrastructure, with a reported 1.2 Tbps of traffic [58]

The amount of data generated by IoT devices is massive: it's estimated that 200 exabytes were produced in 2014 and that number will reach 1.6 zetabytes in 2020 [48].

This should be very worrisome since it means that an alarming volume of sensitive data could be (and likely is) vulnerable to attacks like theft or manipulation. It isn't just the IoT devices in technological applications that are vulnerable – things like medical devices and even (smart) cars are at risk.

In fact, earlier this year the FDA confirmed that St. Jude Medical's implantable cardiac devices, like pacemakers and defibrillators, are vulnerable to hacker access, which could allow them to deplete the battery or administer incorrect pacing or shocks [31]. As for the vulnerabilities facing cars, a group of white hat hackers found a zero-day vulnerability in a Jeep Cherokee that, once exploited, allowed them to indirectly send commands over the Internet to its dashboard functions, steering, brakes, and transmission [23].

Vulnerabilities

In 2014, the Open Web Application Security Project (OWASP), an online community "dedicated to enabling organizations to conceive, develop, acquire, operate, and maintain applications that can be trusted" [45] published a list of it's top 10 IoT vulnerabilities [46]. This list can be seen in Figure B-2. As evidenced by the OWASP list, there are security vulnerabilities present throughout the different layers of the IoT stack. For brevity, we will not discuss all the listed vulnerabilities and instead only concern ourselves with those vulnerabilities associated with specific IoT technologies (i.e., the IoT devices and their communication protocols).

One of the largest factors contributing to the security vulnerabilities faced by IoT devices (even today) is actually rather surpising: many of these devices still have their default credentials in place. In fact, three of the largest botnets that have recently been released into the wild — Mirai, Hajime, and BrickerBot — all leveraged this blunder to achieve their large sizes [47, 20, 27]. To make matters worse, it's actually extremely easy to generate a list of default passwords for many of today's IoT devices: a quick Google search would suffice.

In addition, IoT devices will inevitably have software bugs that someone will find and eventually exploit. In devices like computers or even smart phones, a software bug can easily be fixed with a developer patch that can be released to all affected devices. Most IoT devices, however, “will never receive a single update from the manufacturer as time goes on, so patches aren’t an option to help address emerging threats” [28]. Whatever the reason is for this manufacturer behavior, this is obviously a strong security concern since undisclosed vulnerabilities may likely remain exploitable for a device’s lifetime.

As potentially unsurprising as it sounds, the common communication protocols used by IoT devices are also speckled with vulnerabilities. BLE, for example, has been shown to be susceptible to packet sniffing, man-in-the-middle attacks, and battery-draining denial-of-service attacks [41]. As for ZigBee, there exists KillerBee, a Python tool developed by Joshua Wright, that conveniently — depending on whether or not or you’re a hacker — allows one to “eavesdrop on ZigBee networks, replay traffic, attack cryptosystems and much more” [28].

Similarly, the common application protocols seen in IoT systems are also vulnerable in one way or another. MQTT, for example, leaves itself open to DoS attacks against open ports and buffer overflows attacks, and also sends message usernames and passwords in plaintext [19]. CoAP, on the other hand, is typically implemented atop Datagram Transport Layer Security (DTLS), which provides general security at the cost of high overhead. Without DTLS, however (which may happen under tight energy constraints), CoAP is subject to various man-in-the-middle attacks [49].

2.2 Operational Technology Systems

2.2.1 Intro

Operational Technology (OT) Systems are systems in which computers or other processing devices are used to monitor or alter the physical state of a system. This is a term that has been established to provide a distinction between traditional IT systems and Industrial Control Systems (ICS), computer systems that are specialized in maintaining the various control systems that are often found in the industrial sectors and critical infrastructure (assets deemed essential for the functioning of a society and economy). The types of control systems that fall under the ICS umbrella include Supervisory Control and Data Acquisitions (SCADA) systems, Distributed Control Systems (DCS), and Programmable Logic Controllers (PLC) [55].

SCADA systems are highly distributed (usually over several thousand square kilometers) that are used to control geographically dispersed assets in which centralized data acquisition is as important as control [55]. They are used in distributed systems, such as water distribution, oil and natural gas pipelines, and electrical power grids. As shown in Figure B-3, these system are comprised of various subsystems like PLCS, Remote Telemetry Units (RTUs), Human Machine Interfaces (HMIs), data historians, control servers, and various network components (e.g., routers, modems, access points).

SCADA systems are often networked together alongside it's sister system — DCS, similarly architectured systems that are used in more localized and process-based systems — due to their complementary roles in controlling and maintaining industrial systems, as in the case with electric power control centers (SCADA) and generation facilities (DCS), for example. Consequently, not only are these control systems vital to the operation of the world's critical infrastructures, they are also often well interconnected and tend to be mutually dependent, such that a failure or incident in one infrastructure could directly (or indirectly) affect another infrastructure.

PLCs are small industrial computers originally designed to perform the logic functions executed by electrical hardware (relays, switches, and mechanical timer/counters

[55]. They are used to continuously monitor the state of input devices and then control the state of the associated output devices based on a custom program it has been given. These controllers are widely used in SCADA and DCS systems as part of their control components to provide local management of electromechanical processes [32].

A brief overview of the various components found in ICS is given in Table A.2

2.2.2 IIoT

OT systems strongly overlap with an emerging sub-domain of the IoT known as the "Industrial Internet of Things" (IIoT), which can be defined as "the embedded systems, controls, sensors, and monitors on industrial equipment and the software systems powering them" [6]. It is essentially the use of some technologies found in the IoT — including machine learning, big data, sensors, M2M computing — in industrial systems. Because of its large overlap with OT systems, IIoT and OT systems can be thought of as one and the same for our purposes.

This union of IoT and OT systems providessubstantial benefits for the industrial realm. The first of these is the notion of a smart enterprise control: the IIoT will allow for closer integration of the various enterprise silos, such as production, supply chain management, and even customer relationship management systems. This optimized manufacturing enterprise enables a greater degree of business control, and will yield high efficiency and profitability gains across the board. Examples of smart enterprise control include reducing the size of product recalls, detection of defective products earlier in the manufacturing process and modification of product design to eliminate root causes, and modification of production planning based on weather forecasts. [11]

Another crucial benefit of IIoT is that of asset performance management. The deployment of the various sensors and actuators found in IoT allows for data to be easily gathered from the field and converted into actionable information in real time. This in turn means that applications such as energy management and predictive maintenance become much more easily implemented in these types of systems. These applications will help enterprises potentially save abundantly by avoiding pitfalls like maintaining equipment that doesn't require maintenance, or neglecting equipment that fails and causes unanticipated production downtime. [11]

This "industrial internet," as it is often called, is still at an early stage, but its adoption is accelerating. In fact, two notable government initiatives, Germany's Industrie 4.0 (2011) and China's "Made in China 2025" (2014), are aiming to bring together the private and public sectors, and academia to form a vision and action plan for the widespread application of these technologies to their respective industrial sectors in the years to come. Another indicator of the IIoT momentum is the impressive growth of the Industrial Internet Consortium (IIC). The IIC was founded in March 2014 by AT&T, Cisco, General Electric, IBM, and Intel with the mission of "deliver[ing] a trustworthy IIoT in which the world's systems and devices are securely connected and controlled to deliver transformational outcomes;" it has now grown to a membership size of 258 companies. [40]

2.2.3 Difference from IT Systems

Unlike typical IT systems, processes carried out by OT systems tend to have direct impact on the physical world, which means that most, if not all, attacks on these systems can have devastating physical effects. The Aurora Generator Test, for example, conducted by the Idaho National Laboratory in 2007 — a controlled cyber attack on a component of an electric generator — resulted in the destruction of power generation equipment [35]. Stuxnet, a now infamous computer worm that was allegedly jointly built by the American and Israeli government, carried out a mutli-staged attack that caused substanstial damage to Iran's nuclear program[21]. More recently, in 2015, a group of hackers hijacked three power distribution company systems in Ukraine and cut the power to more than 230,000 people, and sabotaged operator workstations to make it difficult to restore the electricity [59]. A year later, Ukraine was again targeted – this time, however, by a fully automated malware dubbed "Crash Override" that struck an electric transmission station, blacking out a portion of the Ukrainian capital that was equivalent to a fifth of its total power capacity [22].

OT systems are hard real-time systems — the operations on these types of systems have low latency and high communication integrity requirements because many of the processes involve feedback loops [61]. This is in stark contrast to IT systems, which can more easily tolerate latency and errors. Without these real-time constraints in place, buggy behavior in various OT system components could propagate, resulting in potentially destructive behavior.

Another difference lies in the type of operating systems used: OT systems typically use Real-time Operating Systems (RTOS) — Wind River Systems' VxWorks, for example, was the most popular embedded operating system[1] in 2005 and claimed 300 million devices in 2006 [61] — whereas IT systems use general purpose operating systems, such as Unix. RTOS contribute to the low latency requirements of OT systems, as well as the prevalence of buffer overflow vulnerabilities observed therein because many field level devices are embedded systems that run years without rebooting, accumulating fragmentation [61].

The communication protocols used in OT systems are also different compared to those in IT systems. In IT systems, general purpose communication protocols are used, such as TCP and UDP; these protocols were developed in the face of expected packet losses and variable latencies. Those network assumptions, however, do not (or rather, cannot) hold true in OT systems. As such, special and proprietary network protocols have been developed for use in OT environments.

One such example is MODBUS (and related variants), which was created in the late 1970's by the now Schneider Electric company to allow communications to its line of PLCs. The protocol's simplicity, which follows a simple request-reply scheme, and efficiency have caused it to become widely adopted throughout OT systems as a defacto industrial standard [4]. Another OT system communication protocol is Distributed Network Protocol v3.0 (DNP3), and it is used between master control stations and outstations for the electric utility industry and water companies; it's implemented by several manufacturers — primarily only in the US — due to its small memory consumption [61].

These mentioned differences between OT and IT systems, as well as others, are

[1]The terms "embedded operating systems" and "RTOS" are oftentimes used synonymously.

summarized in Table A.3.

2.2.4 Security

Context

Historically, most of the OT systems used today were deployed decades ago, before cyber security was thought to be a prevalent issue. At that time the design of the systems were predominantly focused on performance and communication integrity (via error correction and detection capabilities), and security for them meant not much more than physically securing them, as well as ensuring their complete separation from the Internet — this is called air gapping the system. The security associated with this network air gap was eventually overcome, though, in successful attacks like Stuxnet. OT systems have also changed since then, and are now integrating the OT network with corporate networks and the Internet.

This change has brought with it several factors that contribute to the growth of security vulnerabilities for OT systems, which include: (1) vulnerabilities that arise from the new network interconnectivity, (2) lack of security in the access links used for remote diagnostics and maintenance, (3) vulnerabilities found in the standardized technologies that are now being used (e.g., Microsoft Windows), and (4) the growing availability of technical information regarding the design of these control systems that can prove dangerous in the hands of adversaries [42]. This growth in OT system vulnerabilities is well captured in Dell's annual Security Threat Report: in it, Dell reported that the number of their observed attacks on OT systems more than doubled from 163,000 in 2013 to 675,000 in 2014, and, unsurprisingly, 25% of those attacks were because of buffer overflow vulnerabilities [17].

Vulnerabilities

The factors contributing to the new security vulnerabilities faced by OT systems are a result of their connectedness with corporate networks and the Internet. However, OT systems by themselves are still fraught with vulnerabilities that stem predominantly from their real-time constraints.

Notwithstanding the pervasiveness of buffer overflow vulnerabilities that has already been mentioned, there are other crucial software vulnerabilities that need to considered. The major embedded operating system VxWorks, for example, is essentially a monolithic kernel with applications implemented as kernel tasks; this means that there is little privilege separation since all tasks run with the highest privileges [61]. Another blatant security hole is the lack of sophisticated encryption algorithms that can be used for authentication (to prevent man-in-the-middle attacks), or data encryption. Because of this, the prominent communication protocols, MODBUS and DNP3, are easily exploited since their messages contain function codes that specify an action to be taken by the recipient — all of this is done without any authentication [4, 61]. This drawback is largely due to the fact that many field and control devices have reduced computing capabilities and/or limited energy availability to perform such a task [5].

Another consequence of the characteristics shared by OT systems is that the security practices followed are fairly limited. Hard/software security patching and upgrading — a very routine activity in IT systems to help in coping with new vulnerabilities as they are discovered — usually requires that part of the OT system be set temporarily offline [5]. This action, though, can often take weeks or months of planning, which leaves the system vulnerable in the meantime. Moreover, any OT system downtime can have unfavorable real-world consequences, such as an electric grid going off line.

The real-time constraints of most OT systems also makes asynchronous and/or sporadic actions difficult to be carried out. Firewalls, intrusion detection systems (IDS), and other complex filters commonly used in IT systems can introduce unpredictable or unacceptable delays in OT systems [5]. Finally, this constraint means that OT systems are highly vulnerable to DoS attacks.

2.3 Attack Trees

2.3.1 Intro

In analyzing the security of systems, there is a great benefit in being able to systematically itemize the different ways in which a system can be attacked (i.e., all a system's attack vectors). One such tool, named "attack trees", was introduced by Bruce Schneier in 1999 [50], and has proved to be an intuitive aid in risk analysis. The concept for these attack trees was derived from a reliability analysis technique introduced by Bell Phone Laboratories in 1962; it was called "fault tree analysis", and was similarly used to itemize all the different ways a fault intolerant system (in this case, the launch of an intercontinental ballistic missile) could fail [18].

Attack trees represent potential attacks on a system using a tree diagram. The root of the tree represents the ultimate goal of the attack, typically an event that could significantly harm the system in question. The different ways that an attacker can cause this compromise are represented as lower level nodes of the tree; these are sub goals that have to be met in order the achieve the root goal. The dependencies between different nodes on the same level of the tree are modeled as either "AND" compositions (all must be achieved for the upper level goal to succeed), or "OR" compositions (any can be achieved for the the upper level goal to succeed). The overall attack is completed (or, rather, can be completed) if there is some path, an attack vector, from the root node to a leaf node that can be satisfied. As these trees each only represent a single compromise, a complex system can have a forest of attack trees that are relevant to its operations [37].

An example attack tree for a simple attack is shown in Figure B-4. In it, the ultimate goal for an attacker is "Open Safe," and the root's children nodes are all "OR" nodes, so satisfying any one of those nodes will satisfy the root node. As can be seen, different sub goals can have varying levels of accompanying complexity; the "Learn Combo" goal, for example has three levels of goals underneath it; the other goals on the same level as it — "Pick Lock", "Cut Open Safe", and "Install Improperly" — have none.

2.3.2 Benefits

The benefits that attack trees provide, other than attack vector itemization, are strong. They allow the refinement of attacks to be at a layer of abstraction chosen by the developer [37]. This ability to work at different levels of abstraction means that researchers can work with as much or as little detail of a system as neededeither when developing the attack tree, or when analyzing it. A more technical analyst can develop an attack tree using whatever level of abstraction he or she needs, and then present a higher-level abstraction (i.e., easier to understand) of the attack tree to someone in a managerial role.

Attack trees also allow common attacks to be referenced as reusable modules that apply to multiple network scenarios [4]; this re-usability adds to their scalability. With this feature, common attack trees (or sub trees) can be used as "plug-and-play" modules in more complex system attack scenarios.

Lastly, attack trees can be augmented with numerical values (at each node) to allow for a quantitative analysis of potential attack scenarios [50, 4]. For example, assigning exploitation probabilities or values representing the resource consumption to an attacker (e.g., monetary cost, time spent) to each node can help someone determine the overall feasibilities or cost of different attacks vectors. This can prove invaluable when thinking about a system's security: if certain attack vectors are highly improbable, or would prove to be too costly for an attacker to execute, then resources might not have to be deplyed to prevent the exploitation of that vulnerability. Alternatively, if an attack seems to be very probable, a system administrator would know what security aspects of the system he or she should focus on enhancing.

2.3.3 Limitations

Attack trees are certainly useful for risk analysis, but they are not without their set of challenges. One of the most obvious limitations to attack trees is how time-intensive it is to develop one. In fact, the original development method proposed stated that it should be an iterative process that potentially spans several months

[50]. Of course, this is a very unattractive method, especially when cyber systems can (and most likely will) change, maybe even many times, over the course of several months.

Another hurdle stems from the enormous complexity and connectedness of cyber systems. As opposed to physical systems, which only have a finite number of failure points (e.g., there are only so many ways to break into a safe), cyber systems have an essentially endless number of attack vectors due to their scale and integration with other potentially insecure systems. The breadth that has to be covered by a set of attack trees for most cyber systems can potentially grow exponentially as new additions are made to the system.

The process of developing an attack tree requires extensive knowledge of the system at hand, but it's nearly impossible for even a system designer or developer to be able to consider every possible attack vector. What's more, in cyber systems there can be several unknown unknowns – zero-day vulnerabilities that not even system designers or developers know about. Considering the fact that zero-day vulnerabilities are on the rise — the number of such vulnerabilities discovered rose from 23 in 2013 to 54 in 2015 [12] — it is difficult to ever consider an attack tree to be fully complete.

Another limitation is a direct consequence of the two previously mentioned challenges: because cyber systems are so complex and face many unknowns, it is extremely difficult to accurately determine any worthwhile risk metrics that can be added to the attack tree. For our purposes, we define risk as *the expression of the likelihood that an adversary can exploit a specific vulnerability of a particular target to cause a given set of consequences* [4]. Without any such justifiable metrics[2] it becomes infeasible to run any sort of quantifiable analyses on the attack vectors proposed by an attack tree to gain a better understanding of the system vulnerabilities.

[2]A standardized source of risk metrics does exist, the Common Vulnerability Scoring System (CVSS) developed by the Forum of Incident Response and Security Teams (FIRST), but these "CVSS scores" are not determined objectively, so one may be skeptical in placing full trust in it.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Attack Tree Generator

3.1 Current System

The attack tree generator we will be focusing on was developed by Dr. Howard Shrobe at MIT CSAIL [53]. It was developed for usage in self-diagnosising adaptive systems: systems that are capable of detecting their own malfunctions, diagnosing the failure, repairing themselves after the failure. It was also seen as a useful tool for long-term system monitoring, specifically as a means to detect attacks that would avoid detection from systems that use straightforward methods like attack signatures or anomaly profiling. Of course, the attack tree generator can also be used offline to identify the security vulnerabilities in a computational environment.

The core components of the attack tree generator are (1) a structural model of the potential computing environments, and (2) a set of rules describing the potential attack methods and techniques on the system. The structural model describes the network structure and topology, the system machine components and their subsystems, data access rights and locations, and the dependency relationships between these system components

The rules are are coded in a Lisp-based rule system and are written as goals with missing antecedents, such as the one shown in Figure B-5. They are used to reason about how an adversary might affect a desirable system property, such as: reliable performance, privacy of communications, integrity of communications, integrity of stored data, or privacy of stored data. Using a backward chaining inference system, as well as a description of the computing environment[1] as supporting data, the rules are expanded to either more rules, or the leaves of the reasoning chain. For this generator, the leaves contain specific information about the system vulnerabilities and exploitation methods (e.g., a web server with an OS that is vulnerable to buffer overflow attack).

Together, the structural model and rule set covers the types of computing resources present in the environment; what components those resources are composed of; how the components depend on one another; and the attack methods and techniques that could be used on those components. Given an adversarial goal against a computing environment, the generator consults the rule set and computing environment description, and expands the relvant rules to reason through all pertinent attack vectors, eventually yielding an attack tree. Figure B-6 shows a sample attack tree generated for the MIT CSAIL AI lab computing environment at the time (2001). In this case, the adversary's goal was to affect the reliable performance of a process on a "typical dwarf" machine (a generic machine in the AI lab).

A summary of the structural model that was described in the original system can be found in Figure B-7. As can be seen, we distilled the structural model into three categories: *Network structure and topology*, *System machine components and resources*, and *Access rights*. The first of these categories contains basic networking components and abstractions like firewalls, routers, switches, and subnets. The second is comprised of computer resources — code, data, and processes — and the operating system, which has a scheduler and network stack. The last encompasses agents (i.e., users or processes), their capabilities, and machine access pools (i.e., collective access rights).

Similarly, Figure B-8 shows a summary of the types of rules described in the original system's rule set. We categorized the different types of rules found into one of the three prongs of the traditional security goals of the CIA model – Confidential-

[1]Note: the computing environment description is an explicit summary of the components found in the system of interest, whereas the structural model is more of a template detailing what descriptions the generator supports.

ity, Integrity, and Availability[2]. For succinctness, we only include high level attack techniques and methods, not specific vulnerabilities. The techniques and methods we identified towards breaching confidentiality are against communications, uncovering secrets, and discovering system code and data. To compromise integrity, there are techniques that target code and data, and access rights. Lastly, the techniques we identified against availability target performance and network communications.

3.2 Proposed Modifications

3.2.1 New IT Modifications

IT Structural Model Changes

The current structural model contained basic IT system components as mentioned in Section 3.1. However, to capture the components found in modern IT systems, the structural model has to be modified to include components not originally considered. The first of these new components is actually an entirely new category that should included in the structural model: *Communication protocols*.

The machines found in current IT systems use a myriad of communication protocols for various purposes. Since these communication protocols can be exploited in different ways, it would be prudent to include them in the structural model. With respect to the OSI model of networks [62], we mainly concern ourselves with application layer communication protocols. The types of application-layer protocols we consider are transfer protocols, and logon protocols. We focus on these two types of protocols because we believe them to be the prominent forms of interactions on IT systems.

Common transfer protocols we think should be included are: HTTP, HTTPS, FTP, FTPS, SFTP, and SCP. We consider the insecure and secure versions of these protocols to be distinct because the security protocol they run on top of may have

[2]Although there are different models of security goals that might be more applicable, such as the Parkerian Hexad that builds upon the CIA model [1], we chose the CIA model because it has garnered the most consensus.

their own vulnerabilities that need to be taken into account. Our proposed logon protocols include: SSH; Telnet[3]; and Remote Desktop Protocol (RDP), a common protocol used by Windows computers.

Since communications in IT systems have moved to both wired and wireless mediums, we must also include some physical layer protocols to capture these changes. To that end, we include the most common wired and wireless physical layer protocols used in IT systems, Ethernet and Wi-Fi, respectively. To capture the notion of wireless communications, we add wireless access points (WAPs) to the *Network structure and topology* category since these access points can be used carry out different attacks (e.g., capturing network traffic, spoofing access points, etc) [57].

The rest of our suggested modifications fall under the *System machine components and resources* category. Something the original model does not consider are the various types of peripherals that may be connected to a machine; the exact peripherals themselves are not entirely interesting, but rather the presence of the external ports are. Specifically, peripheral ports like USB and Thunderbolt have been subjected to attacks in the past [39, 52], so we think it's valuable to include them.

Next, we need to consider a computer resource that is often overlooked: firmware, the low-level software code that provides the necessary capabilities and functionalities for the underlying hardware work. Firmware has recently gained much attention in the security field as it can provide a strong foothold for adversaries, and a number of firmware vulnerabilities have been found[60, 63].

These proposed changes to the structural model for IT systems are summarized in Figure B-9.

IT Rule Set Changes

As IT systems have grown, so too have the techniques and methods adversaries use to compromise their confidentiality, integrity, and availability goals. In order to reflect these new possibile attacks, the original rule set has to be expanded. With

[3]The current system does include notions of SSH and Telnet, but not as their own communication protocols.

that in mind, we identified the new attack techniques and methods that should be included in a modified rule set.

Originally, exfiltration and reverse engineering were not considered. To us, exfiltration is distinct from just reading files on a target system in that exfiltrating implies the actual movement of the file out of the target system (instead of just the information), which would allow an adversary to, for example, decrypt secured files at their leisure. Similarly, reverse engineering, can be performed on binary files to determine the underlying code and data and reveal potential vulnerabilities.

Another novel technique against confidentiality is a side-channel attack against cryptographic keys, whose secrecy is paramaount since they're used in the encryption or authentication of data. Side-channel attacks are those that retrieve side-channel information — information that is neither the plaintext or ciphertext of an encryption process — based on the physical implementation of a cryptosystem. This side-channel information can potentially be used to break the system. Examples of relevant information include timing information, power consumption, electromagnetic leaks, and even sound. [54]

The techniques adversaries can use to compromise the integrity goals of a system have evolved since the the development of the attack tree generator. To compromise the communications of a system (as well as that of the data), adversaries can employ a variety of man-in-the-middle attacks, attacks in which adversaries alter the communication between two parties; the result of this is usually a person or program successfully masquerading as a legitimate person or program. Man-in-the-middle attacks include spoofing and session hijacking attacks [10]. These attacks can also be used to compromise the confidentiality of network traffic, but we keep it as an attack against integrity because of the impersonation qualities.

Another devastating attack adversaries use to subvert a system's integrity is to focus their efforts on compromising some of the elements found throughout the system supply chain. This type of attack can be especially fruitful because of the growing use of commercial off-the-shelf technology, which potentially have vulnerabilies unexplored by the developers and users, that are being used as a result of systems' growing size and complexity. A system supply chain can be found at all levels of abstraction: open-source software, for example, can be thought of as part of a system's supply chain, as can hardware components developed by upstream manufacturer.

Lastly, adversarial attacks against a system's availability include DoS attacks against system performance, and encryption attacks against the code and data. DoS attacks come in many flavors, but they're all aimed at making a machine or network resource unavailable to its intended users either temporarily or indefinitely; more often than not, they're especially hard to prevent. On the other hand, encryption attacks[4] utilize crypto systems to encrypt files on a target system, making them inaccessible.

These proposed changes to the rule set for IT systems are summarized in Figure B-10.

3.2.2 IoT and OT Modifications

Structural Model Changes

The main structural model modifications that need to be made to account for IoT systems fall under the *Communication protocols* and *System machine components and resources* categories. The new communication protocols are the commonly used Bluetooth and ZigBee physical layer protocols, as well as the MQTT and CoAP application protocols. Security details regarding these communication protocols are described in Section 2.1.4

New system components that need to be considered are embedded operating systems, and the various types of physical devices found in IoT systems. Details regarding embedded operating systems are described in Sections 2.2.3 and 2.2.4. The physical devices that need to be included are actuators, sensors, and microcontrollers and microprocessors; as endpoints, these devices can be used as entry points into an IoT system.

OT systems bring about new elements to the *Communication protocols* and *Net-*

[4]We use the term "encryption attack" here instead of the common "cryptoransomware" to distinguish between the goal of the latter to ransom the secret keys and goal of the former to just make the data inaccessible.

work structure and topology categories. The new communication protocols that need to be considered are the common OT application protocols: MODBUS and DNP3. Security details regarding these application protocols are described in Sections 2.2.3 and 2.2.4.

As described in Sections 2.2.3 and 2.2.4, the real-time constraints of these systems means that their are strict latency requirements on the network; this is an important aspect of OT systems that needs to be captured.

These proposed changes to the structural model for IoT and OT systems are summarized in Figure B-11.

Rule Set Changes

The integrity of the code and data in IoT systems is subjected to security vulnerabilities not already mentioned. The various sensors found therein can be fed false data to disrupt the overall functionality of the system. Similarly, the physical availability of the IoT devices leads to physical tampering of the onboard code and data. The performance (availability) of these devices can also easily be affected by any amount of physical damage an adversary can cause them.

The class of attacks against OT systems as mentioned in Section 2.2.4 are already taken into account by the proposed modifications that have already been mentioned. With that said, there aren't new modifications to suggest.

These proposed changes to the rule set for IoT and OT systems are summarized in Figure B-12.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

Example IoT System

The sample IoT system we analyzed was an urban surveillance system, which can be found in Figure B-13; because of its size and complexity, it makes for a relevant use case. An urban surveillance system collects and stores data from the many video feeds it manages, processes and renders that data, and transmits it to external applications that act on it accordingly. In this case, the external applications trigger alarms and/or events on the interpretted data, and display the relevant video feeds.

The highlighted section comprises the data collection, and data processing and video rendering functionalities. Though many components are shown in Figure B-13, we only consider a handful of them to be relevant. The IP surveillance camera acts as our data source; for our purposes, this type of camera is regarded as an IoT device. The video surveillance manager (VSM) directs a group of these cameras to provide the appropriate video coverage. The video data produced travels across the local access network (LAN) and collects at the storage server. From there, the display server (DP) renders the data and transmits it to a third party for displaying. At the same time, the video processing server (MD) processes the data into something suitable for use by the alarm and event application(s). The main operator console server (CM) allows for supervision and control of the entire operation.

Against a system like this, there are a number of attacks an adversary could consider. A list of example attacks is given in Figure B-14; these attacks are those that one would see in modern IT and IoT systems. Fortunately, these types of attacks are covered by the attack tree generator design changes offered in Sections 3.2.1 and 3.2.2, as well as the original structural model and rule set.

As mentioned in Section 3.1, a description of the computing environment is also necessary for the attack tree generator to be utilized. To that end, we provide an example, Lisp-based description of the urban surveillance system in Figure B-15. It covers the structural model definitions for the relevant components of the surveillance system mentioned earlier.

Chapter 5

Future Work

An obvious next step is to implement the suggested modifications in the mentioned spaces — IT, IoT, and IIoT — and verify their validity against sample system environments. We don't claim that the suggested modifications are fully comprehensive, but believe them to be sufficient; we understand that completeness is unobtainable and realize that we will inevitably need a refinement process.

Other modifications to the attack tree generator we'd like to consider are towards the recognition of Advanced Persistent Threats (APTs). The current system only concerns itself with all the steps leading up to and including an exploit. APTs, however, include post-exploitation activity, such as persisting on a device for an arbitrarily long time, and evading detection or avoiding other defense mechanisms. Analyses on the life cycle — the steps an adversary takes to compromise a system — of these APTs have been done by Lockheed Martin [25] and MITRE [13], which could guide our efforts.

Along those lines, we would like to put some work into developing a framework which can systematically guide someone in the development of attack trees. To the best of our knowledge no such framework exists, and the only methdology mentioned thus far involves spending the time thinking through all adversarial possibilities and then having someone else look over it, and repeating that cycle [50]. If one can emulate the thinking process and methodology an adversary uses, then he or she can more easily think through all the possible attack cases; we believe that a framework built around this could drastically simplify the process of developing attack trees for many users. Resources like MITRE's Common Vulnerabilities and Exposures (CVE) [15]; Common Weaknesses Enumerations (CWE) [16]; and Common Attack Pattern Enumeration and Classification (CAPEC) [14] would prove invaluable towards this endeavor as they highlight the different adversarial patterns seen in the wild, and at different levels of abstractions.

Appendix A

Tables

Communication	Objects have the ability to network with Internet resources or even with each other, to make use of data and services and update their state. Relevant examples include Wi-Fi, Bluetooth and Zigbee.
Addressability	Objects can be located and addressed via discovery, look-up or name services, and hence remotely interrogated or configured.
Identification	Objects are uniquely identifiable. Relevant examples include RFID, NFC (Near Field Communication), and optically readable barcodes.
Sensing	Objects collect information about their surroundings with sensors, record it, forward it or react directly to it.
Actuation	Objects contain actuators to manipulate their environment.
Emedded information processing	Smart objects feature a processor or microcontroller, plus storage capacity, that can be used, for example, to process and interpret sensor information.
Localization	Smart things are aware of their physical location, or can be located. Relevant examples include GPS and mobile phone networks.
User interfaces	Smart objects can communicate with people in an appropriate manner.

Table A.1: IoT enabling technologies [33]

Control Server	Hosts supervisory control software that communicates with lower-level control devices.
Master Terminal Unit (Unit)	Device that acts as the master in an ICS with respect to RTUs and PLCs.
Remote Terminal Unit (Unit)	Special purpose data acquisition and control unit designed to support remote stations.
Intelligent Electronic Devices (IED)	"Smart" sensor/actuator that acquires data, communicates to other devices, and perform local processing and control.
Human-Machine Interface (HMI)	Software and hardware that allows human operators to monitor the state of a process under control.
Data Historian	Centralized database for logging all process information within an ICS.
Input/Output (IO) Server	Control component responsible for collecting, buffering and providing access to process information from control sub-components.

Table A.2: ICS components [55]

	OT	**IT**
Average node complexity	Low (simple devices, sensors, actuators)	High (large servers/file systems/databases)
Number of users	Limited	Very high
Multi-vendor environment	Frequent	Moderate
System lifetime (years)	Some tens	Some
Outage of system availability	Rarely/never tolerable	Often tolerable
Tolerability of time delays	Low/none (real-time)	Medium/high
Acceptable processing times	Milliseconds – minutes	Minutes – days
Tolerability of failures	Low/none	Medium/high
Communication protocol stacks	Special purpose/proprietary/real-time	General-purpose (i.e., TCP/IP, UDP)
Operating systems	Real-time, embedded, special-purpose	General-purpose (i.e., Windows, Unix)

Table A.3: System characteristics for OT and IT systems [5]

Appendix B

Figures

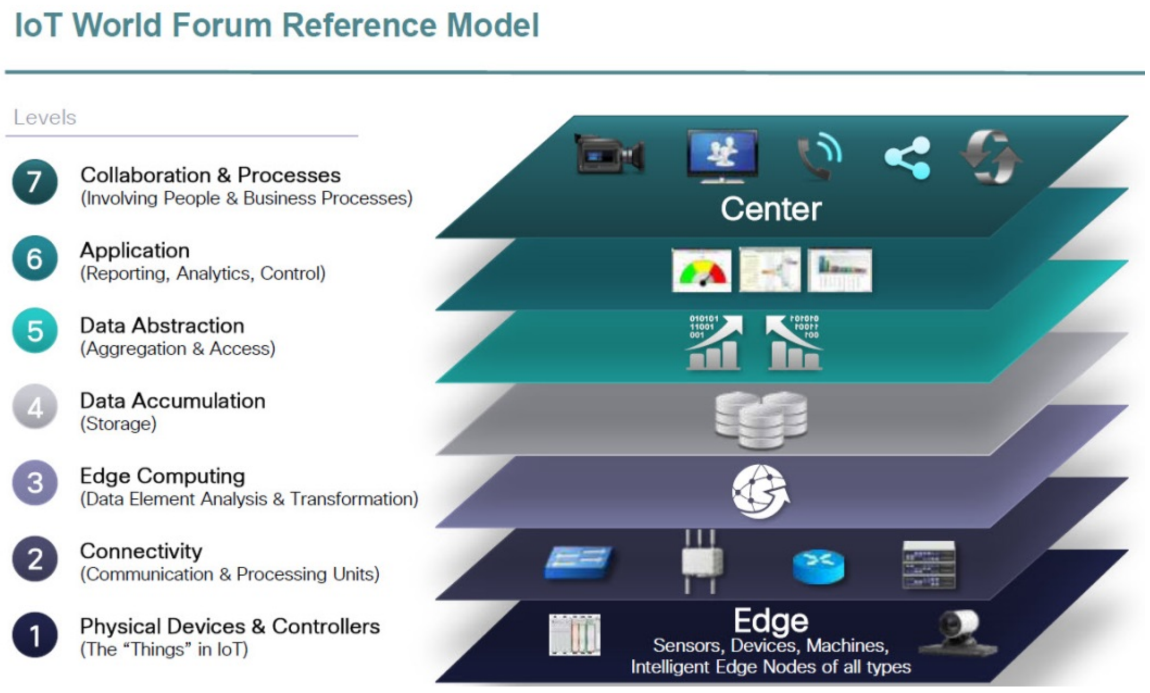

IoT World Forum Reference Model
Levels
7 Collaboration & Processes
(Involving People & Business Processes)
6 Application
(Reporting, Analytics, Control)
5 Data Abstraction
(Aggregation & Access)
4 Data Accumulation
(Storage)
3 Edge Computing
(Data Element Analysis & Transformation)
2 Connectivity
(Communication & Processing Units)
1 Physical Devices & Controllers
(The "Things" in IoT)
Center
Edge
Sensors, Devices, Machines,
Intelligent Edge Nodes of all types

Figure B-1: IoT reference model [8]

I Insecure web interface

II Insufficient authentication/authorization

III Insecure network services

IV Lack of transport encryption/integrity verification

V Privacy concerns

VI Insecure cloud interface

VII Insecure mobile interface

VIII Insufficient security configurability

IX Insecure software/firmware

X Poor physical security

Figure B-2: The Top 10 IoT Vulnerabilities for 2014 [46]

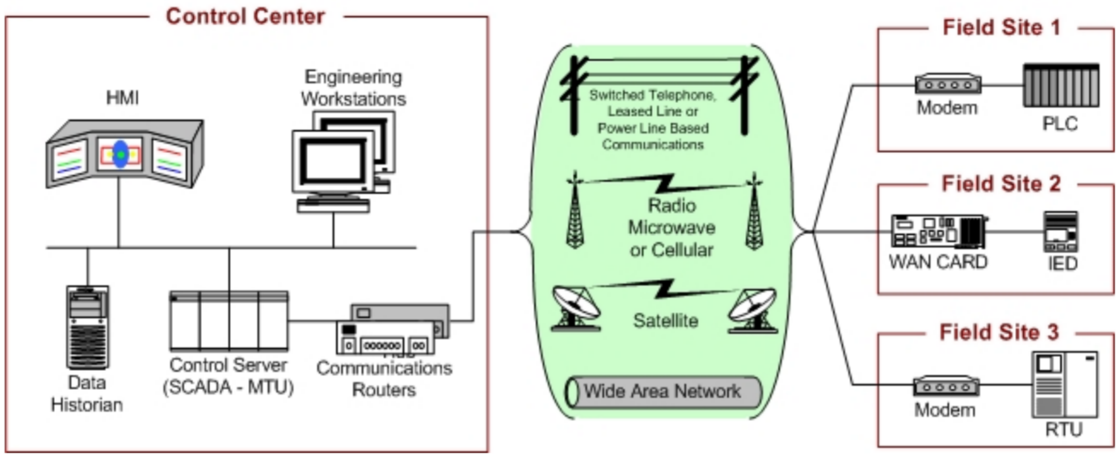

Control Center
HMI
Engineering Workstations
Data Historian
Control Server (SCADA - MTU)
WAN Communications Routers
Switched Telephone, Leased Line or Power Line Based Communications
Radio Microwave or Cellular
Satellite
Wide Area Network
Field Site 1
Modem
PLC
Field Site 2
WAN CARD
IED
Field Site 3
Modem
RTU

Figure B-3: General SCADA system layout [55]

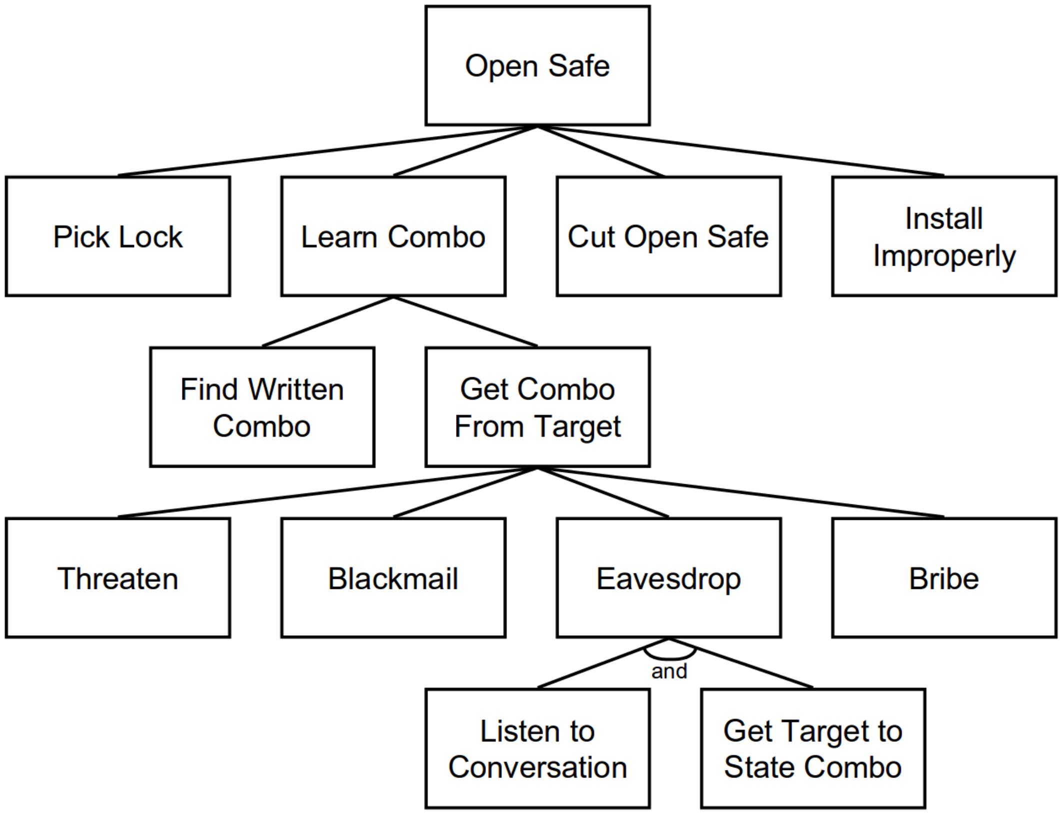
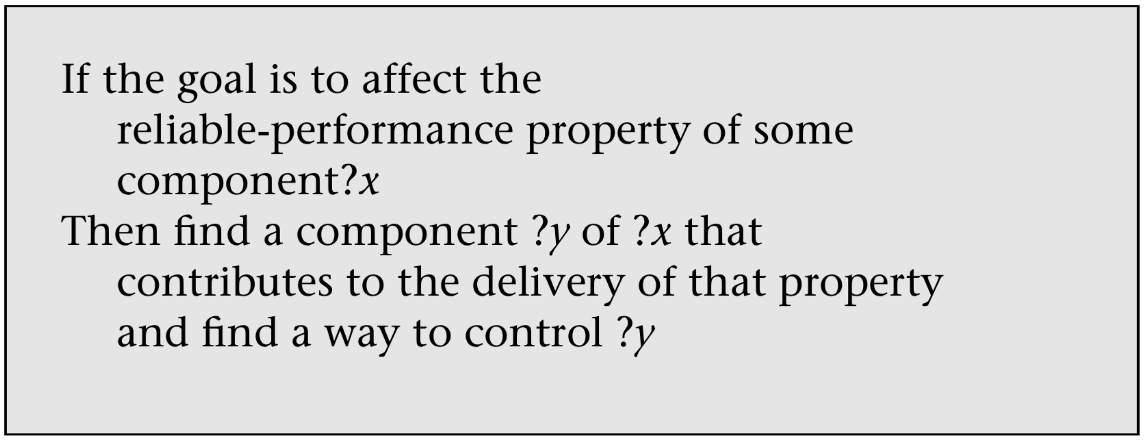


Figure B-4: Example attack tree [50]



If the goal is to affect the
    reliable-performance property of some
    component?*x*
Then find a component ?*y* of ?*x* that
    contributes to the delivery of that property
    and find a way to control ?*y*

Figure B-5: Example vulnerability rule for control relationships [53]

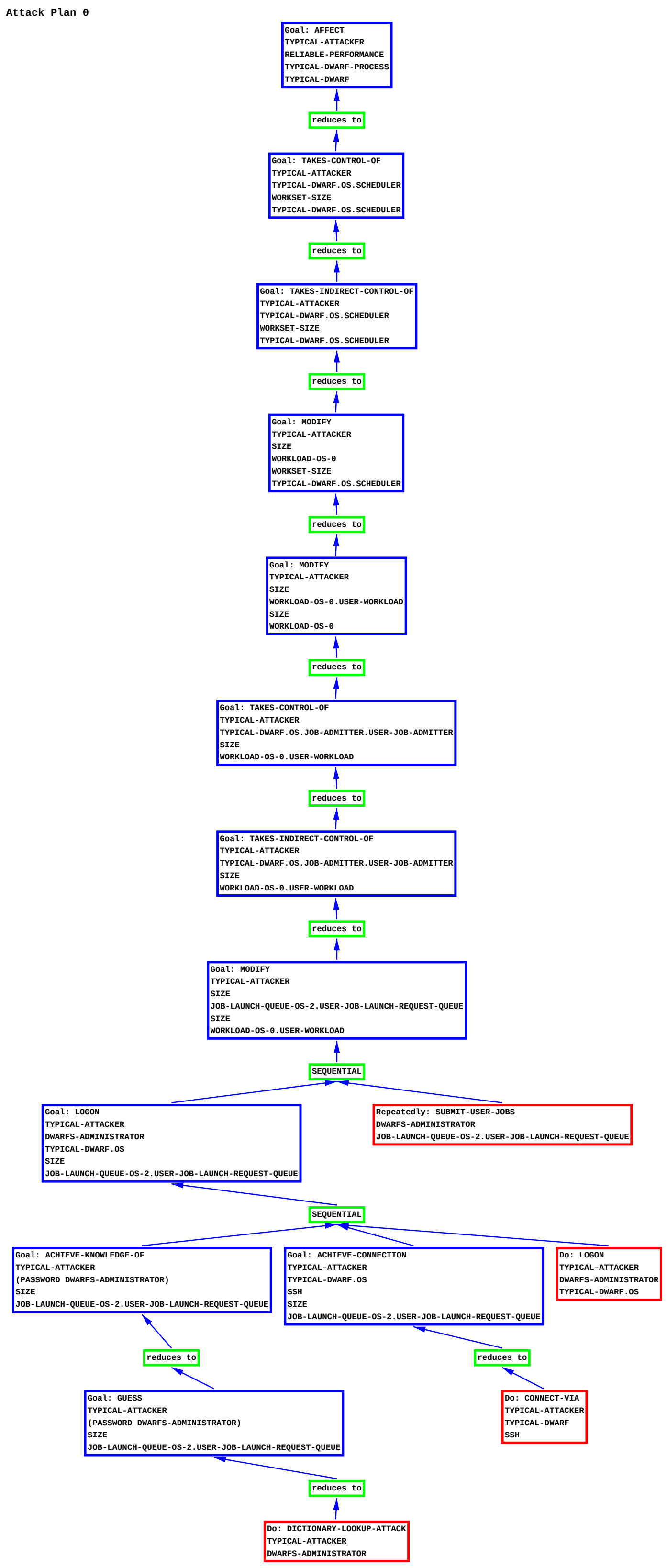

Attack Plan 0
Goal: AFFECT
TYPICAL-ATTACKER
RELIABLE-PERFORMANCE
TYPICAL-DWARF-PROCESS
TYPICAL-DWARF
reduces to
Goal: TAKES-CONTROL-OF
TYPICAL-ATTACKER
TYPICAL-DWARF.OS.SCHEDULER
WORKSET-SIZE
TYPICAL-DWARF.OS.SCHEDULER
reduces to
Goal: TAKES-INDIRECT-CONTROL-OF
TYPICAL-ATTACKER
TYPICAL-DWARF.OS.SCHEDULER
WORKSET-SIZE
TYPICAL-DWARF.OS.SCHEDULER
reduces to
Goal: MODIFY
TYPICAL-ATTACKER
SIZE
WORKLOAD-OS-0
WORKSET-SIZE
TYPICAL-DWARF.OS.SCHEDULER
reduces to
Goal: MODIFY
TYPICAL-ATTACKER
SIZE
WORKLOAD-OS-0.USER-WORKLOAD
SIZE
WORKLOAD-OS-0
reduces to
Goal: TAKES-CONTROL-OF
TYPICAL-ATTACKER
TYPICAL-DWARF.OS.JOB-ADMITTER.USER-JOB-ADMITTER
SIZE
WORKLOAD-OS-0.USER-WORKLOAD
reduces to
Goal: TAKES-INDIRECT-CONTROL-OF
TYPICAL-ATTACKER
TYPICAL-DWARF.OS.JOB-ADMITTER.USER-JOB-ADMITTER
SIZE
WORKLOAD-OS-0.USER-WORKLOAD
reduces to
Goal: MODIFY
TYPICAL-ATTACKER
SIZE
JOB-LAUNCH-QUEUE-OS-2.USER-JOB-LAUNCH-REQUEST-QUEUE
SIZE
WORKLOAD-OS-0.USER-WORKLOAD
SEQUENTIAL
Goal: LOGON
TYPICAL-ATTACKER
DWARFS-ADMINISTRATOR
TYPICAL-DWARF.OS
SIZE
JOB-LAUNCH-QUEUE-OS-2.USER-JOB-LAUNCH-REQUEST-QUEUE
Repeatedly: SUBMIT-USER-JOBS
DWARFS-ADMINISTRATOR
JOB-LAUNCH-QUEUE-OS-2.USER-JOB-LAUNCH-REQUEST-QUEUE
SEQUENTIAL
Goal: ACHIEVE-KNOWLEDGE-OF
TYPICAL-ATTACKER
(PASSWORD DWARFS-ADMINISTRATOR)
SIZE
JOB-LAUNCH-QUEUE-OS-2.USER-JOB-LAUNCH-REQUEST-QUEUE
Goal: ACHIEVE-CONNECTION
TYPICAL-ATTACKER
TYPICAL-DWARF.OS
SSH
SIZE
JOB-LAUNCH-QUEUE-OS-2.USER-JOB-LAUNCH-REQUEST-QUEUE
Do: LOGON
TYPICAL-ATTACKER
DWARFS-ADMINISTRATOR
TYPICAL-DWARF.OS
reduces to
reduces to
Goal: GUESS
TYPICAL-ATTACKER
(PASSWORD DWARFS-ADMINISTRATOR)
SIZE
JOB-LAUNCH-QUEUE-OS-2.USER-JOB-LAUNCH-REQUEST-QUEUE
Do: CONNECT-VIA
TYPICAL-ATTACKER
TYPICAL-DWARF
SSH
reduces to
Do: DICTIONARY-LOOKUP-ATTACK
TYPICAL-ATTACKER
DWARFS-ADMINISTRATOR

Figure B-6: Sample attack plan for the MIT CSAIL AI lab system topology

- Network structure & topology
  - Filters/firewall policies
  - Routers
  - Switches
  - Subnets
- System machine components & resources
  - Computer resource
    - Code
    - Data
      - Data residences & format transformations
    - Processes
      - System
      - Server
  - OS
    - Scheduler
    - Network stack
- Access rights
  - Agents
  - Access Pools
  - Capabilities

Figure B-7: Original structural model

- **Confidentiality**
  - Communication
    - Monitor network traffic
  - Secrets
    - Guess password
    - Dictionary lookup
    - Social engineering
  - Code/data
    - Read code/data
- **Integrity**
  - Code/data
    - Modify code/data
  - Access rights
    - Hack access rights
- **Availability**
  - Performance[1]
    - Gain direct/indirect control of processes
  - Network
    - Control network stack

Figure B-8: Original rule set

[1]Here we abuse the definition of availability to include that of reliability (i.e., reliable performance).

- Communication protocols
  - Ethernet
  - Wi-Fi
  - Transfer protocols
  - Logon protocols
- Network structure & topology
  - Wireless access points
- System machine components & resources
  - Peripherals
    - USB
    - Thunderbolt
  - Computer resources
    - Firmware

Figure B-9: Changes for IT structural model

- **Confidentiality**
  - Code/data
    - Exfiltrate code/data
    - Reverse engineering
  - Secrets
    - Side-channel attack
- **Integrity**
  - Communications
    - Man-in-the-middle attack
  - System subcomponents
    - Infiltrate supply chain
- **Availability**
  - Performance
    - DoS
  - Code/data
    - Encryption

Figure B-10: Changes for IT rule set

- Communication protocols
  - Bluetooth
  - ZigBee
  - CoAP
  - MQTT
  - MODBUS
  - DNP3
- System machine components & resources
  - OS
    * Embedded OS
  - Actuators
  - Sensors
  - Microcontrollers and microprocessors
- Network structure & topology
  - Latency requirements

Figure B-11: Changes for IoT and OT structural model

- **Integrity**
  - Code/data
    * False data injection
    * Physical tampering
- **Availability**
  - Performance
    * Physical damage

Figure B-12: Changes for IoT and OT rule set

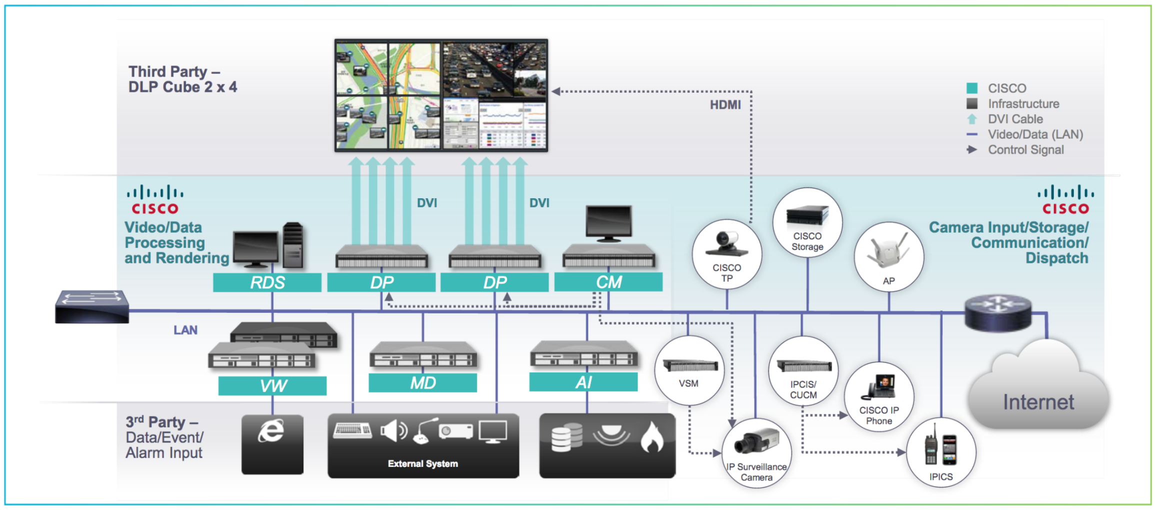


Figure B-13: Urban surveillance IoT network [7]

- Exfiltrate data
  - From camera (e.g., physical tampering)
  - From storage server
- Limit system functionality
  - Disrupt video display
    * Corrupt data
    * Inject false data
    * Physically damage cameras
    * Corrupt or prevent DP rendering
  - Disrupt event & alarm generation
    * Corrupt data
    * Inject false data
    * Corrupt or prevent MD processing
  - Interfere with generated events & alarms
    * Hide events & alarms
    * Intercept events & alarms
- Limit system access
  - DoS attack
  - Gain control of cameras (i.e., botnet)
  - Gain control of VSM
  - Gain control of CM

Figure B-14: Example attacks against urban surveillance IoT network

Figure B-15: Code for urban surveillance system

```
;;; Example CCTV Network

(in-package :aplan)

;; We're assuming that entire network is internal (LAN)
(defsite cctv-network "192.0.0.0" "255.0.0.0")

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Communication pool
;;
;; Encompasses Routers and Switches
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(make-object 'authorization-pool :name 'communication-pool)

;; Capabilities for communication pool
(defcapability communication-super-user communication-pool)

(defcapability communication-user-write communication-pool
  :greater (communication-super-user))

(defcapability communication-user-read communication-pool
  :greater (communication-user-write))

;; Define users in communication pool
(defuser router-administrator
  :capabilities (communication-super-user)
  :authorization-pools (communication-pool))
```

```
(defuser switch-administrator
  :capabilities (communication-super-user)
  :authorization-pools (communication-pool))

;; Define machines in communication pool
(defrouter cctv-router ("192.0.0.0") :authorization-pool
    communication-pool :superuser router-administrator)

(defswitch cctv-switch switch "192.1.1.1" :authorization-pool
    communication-pool :superuser switch-administrator)

;; Define resources in communication pool
(defresource router-password-file password-file
    :capability-requirements ((write communication-super-user)
    (read communicationr-user-read))
    :machines (cctv-router))

(defresource router-configuration-file configuration-file
    :capability-requirements ((write communication-super-user)
    (read communication-user-read))
    :machines (cctv-router))

(defresource switch-configuration-file configuration-file
    :capability-requirements ((write communication-super-user)
    (read communication-user-read))
    :machines (cctv-switch))

;; Define router access policies
(tell-policy cctv-router telnet :negative-location-mask "255.0.0.0"
    :negative-location-address "192.0.0.0")
(tell-policy cctv-router ssh :positive-location-mask "0.0.0.0"
    :positive-location-address "0.0.0.0")
```

```
;; Define switch access policies
(tell-policy cctv-switch telnet :negative-location-mask "255.255.0.0"
    :negative-location-address "192.1.0.0")
(tell-policy cctv-switch ssh :positive-location-mask "0.0.0.0"
    :positive-location-address "0.0.0.0")

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Server pool
;;
;; Encompasses Storage Server, Display Server,
;; and Video Processing Server
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(make-object 'authorization-pool :name 'server-pool)

;; Capabilities for server pool
(defcapability server-super-user server-pool)

(defcapability server-user-write server-pool
  :greater (server-super-user))

(defcapability server-user-read server-pool
  :greater (server-user-write))

;; Define users for server pool
(defuser server-administrator
  :capabilities (server-super-user)
  :authorization-pools (server-pool))

(defuser server-user
  :capabilities (server-user-write)
  :authorization-pools (server-pool))

;; Define machines/computers for server pool
(defcomputer storage linux-computer "192.1.1.2"
  :authorization-pool server-pool
  :superuser server-administrator)

(defcomputer display linux-computer "192.1.1.3"
  :authorization-pool server-pool
  :superuser server-administrator)

(defcomputer video-processing linux-computer "192.1.1.4"
  :authorization-pool server-pool
  :superuser server-administrator)

;; Define resources in server pool
(defresource video-database database
    :capability-requirements ((write server-super-user)
        (read server-user-read))
    :machines (storage))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Manager pool
;;
;; Encompasses Main Operator Console Server, and Video Surveillance
;;  Manager
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(make-object 'authorization-pool :name 'manager-pool)

;; Capabilities for manager pool
(defcapability manager-super-user manager-pool)
```

```
(defcapability manager-user-write manager-pool
  :greater (manager-super-user))

(defcapability manager-user-read manager-pool
  :greater (manager-user-write))

;; Define users for manager pool
(defuser manager-administrator
  :capabilities (manager-super-user)
  :authorization-pools (manager-pool))

(defuser manager-user
  :capabilities (manager-user-write)
  :authorization-pools (manager-pool))

;; Define machines/computers for manager pool
(defcomputer main-operator-console windows-7-computer
    "192.1.1.5"
  :authorization-pool manager-pool
  :superuser manager-administrator)

(defcomputer typical-video-surveillance-manager windows-7-computer
    "192.1.1.6"
  :authorization-pool manager-pool
  :superuser manager-administrator)

;; Define resources in server pool

;; Master password file contains login info for all servers/devices
;; on the network
(defresource master-password-file password-file
    :capability-requirements ((write manager-super-user)
        (read manager-user-read))
    :machines (main-operator-console))

;; Local password file contains login info for the (camera) devices
;; that fall under the video-surveillance-manager
(defresource local-password-file password-file
    :capability-requirements ((write manager-user-write)
        (read manager-user-read))
    :machines (typical-video-surveillance-manager))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Device pool
;;
;; Encompasses Video Surveillance Manager, and IoT Camera
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(make-object 'authorization-pool :name 'device-pool)

;; Capabilities for device pool
(defcapability device-super-user device-pool)

(defcapability device-user-write device-pool
  :greater (device-super-user))

(defcapability device-user-read device-pool
  :greater (device-user-write))

;; Define users for device pool
(defuser device-administrator
  :capabilities (device-super-user)
  :authorization-pools (device-pool))
```

```
(defuser device-user
  :user-type typical-user
  :capabilities (device-user-write)
  :authorization-pools (device-pool)
  :machines (typical-camera))

;; Define machines/computers for device pool
(defcomputer typical-camera embedded-linux-computer "192.1.1.7"
  :authorization-pool device-pool
  :superuser device-administrator)

;; Define resources for device pool
(defresource devices-scheduler-policy scheduler-policy-file
  :capability-requirements ((write device-super-user)
    (read device-user-read))
  :machines (typical-camera))

(defresource devices-password-file password-file
  :capability-requirements ((write device-super-user)
    (read device-user-read))
  :machines (typical-camera))

(defresource device-video-file graphic-video-file
  :capability-requirements ((write device-user-write)
    (read device-user-read))
  :machines (typical-camera))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Processes and attackers
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;; Instantiate server processes
(instantiate-a-process 'storage-server-process '(storage))
(instantiate-a-process 'display-server-process '(display))
(instantiate-a-process 'video-processing-server-process
   '(video-processing))

;; Instantiate manager processes
(instantiate-a-process 'operator-console-server-process
   '(main-operator-console))
(instantiate-a-process 'video-surveillance-manager-process
   '(typical-video-surveillance-manager))

;; Instantiate device processes
(instantiate-a-process 'typical-user-process
   '(typical-camera) :role-name 'typical-camera-process)

;; Instantiate attacker
(create-attacker 'typical-attacker :negative-mask-address "192.1.0.0"
   :negative-mask-mask "255.255.0.0")
```

THIS PAGE INTENTIONALLY LEFT BLANK

Bibliography

[1] Jason Andress. *The basics of information security: understanding the fundamentals of InfoSec in theory and practice*. Syngress, 2014.

[2] Kevin Ashton. That 'Internet of Things' Thing. Available at `http://www.rfidjournal.com/articles/view?4986` (2009/06/22).

[3] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[4] Eric J Byres, Matthew Franz, and Darrin Miller. The use of attack trees in assessing vulnerabilities in scada systems. In *Proceedings of the international infrastructure survivability workshop*, 2004.

[5] Manuel Cheminod, Luca Durante, and Adriano Valenzano. Review of security issues in industrial networks. *IEEE Transactions on Industrial Informatics*, 9(1):277–293, 2013.

[6] Li Ping Chu. Global trends: Big data analytics for the industrial internet of things. Technical report, O'Reilly Media, Inc, 2016.

[7] Cisco. Unified Management for City Infrastructure. Available at `https://www.cisco.com/c/dam/en\_us/solutions/industries/docs/scc/scc-city-operations-center-aag.pdf` (2014).

[8] Cisco. The internet of things reference model. Technical report, Cisco, 2014.

[9] RS Components. 11 Internet of Things (IoT) Protocols You Need to Know About. Available at `https://www.rs-online.com/designspark/eleven-internet-of-things-iot-protocols-you-need-to-know-about` (2015/04/10).

[10] Mauro Conti, Nicola Dragoni, and Viktor Lesyk. A survey of man in the middle attacks. *IEEE Communications Surveys & Tutorials*, 18(3):2027–2051, 2016.

[11] John Conway. The industrial internet of things: An evolution to a smart manufacturing enterprise. Technical report, Schneider Electric, 2015.

[12] Symantec Corporation. 2016 internet security threat report. Technical report, Symantec Corporation, 2016.

[13] The Mitre Corporation. Adversarial Tactics, Techniques & Common Knowledge. Available at `https://attack.mitre.org` (2016).

[14] The Mitre Corporation. Common Attack Pattern Enumeration and Classification. Available at `https://capec.mitre.org` (2007).

[15] The Mitre Corporation. Common Vulnerabilities and Exposures. Available at `https://cve.mitre.org` (1999).

[16] The Mitre Corporation. Common Weaknesses Enumerations. Available at `https://cwe.mitre.org` (2006).

[17] Inc. Dell. Dell security annual threat report 2015. Technical report, Dell, Inc., 2015.

[18] Thomas W DeLong. A fault tree manual. Technical report, ARMY MATERIEL COMMAND TEXARKANA TX INTERN TRAINING CENTER, 1970.

[19] Patricia Du. IoT Message Protocols: The Next Security Challenge for Service Providers? Available at `https://f5.com/about-us/blog/articles/iot-message-protocols-the-next-security-challenge-for-service-providers-25064` (2017/02/24).

[20] Adam Clark Estes. This Hacker Is My New Hero. Available at `http://gizmodo.com/this-hacker-is-my-new-hero-1794630960` (2017/04/25).

[21] Nicolas Falliere, Liam O Murchu, and Eric Chien. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response*, 5(6), 2011.

[22] Andy Greenberg. 'Crash Override': The Malware That Took Down a Power Grid. Available at `https://www.wired.com/story/crash-override-malware/` (2017/06/12).

[23] Andy Greenberg. Hackers Remotely Kill a Jeep on the Highway – With Me in It. Available at `https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/` (2015/07/21).

[24] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

[25] Eric M Hutchins, Michael J Cloppert, and Rohan M Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1(1):80, 2011.

[26] Ponemon Institute. 2015 cost of cyber crime study: United states. Technical report, Ponemon Institute, 2015.

[27] Swati Khandelwal. Hajime 'Vigilante Botnet' Growing Rapidly; Hijacks 300,000 IoT Devices Worldwide. Available at `http://thehackernews.com/2017/04/vigilante-hacker-iot-botnet\_26.html` (2017/04/26).

[28] Adam Kliarsky. Detecting attacks against the 'internet of things'. (in press), 2017.

[29] Brian Krebs. KrebsOnSecurity Hit With Record DDoS. Available at `https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/` (2016/09/21).

[30] James LaPiedra. The information security process: Prevention, detection and response. *Style (DeKalb, IL)*, 2002.

[31] Selena Larson. FDA confirms that St. Jude's cardiac devices can be hacked. Available at `http://money.cnn.com/2017/01/09/technology/fda-st-jude-cardiac-hack/index.html` (2017/01/09).

[32] PLC Manual. Basic Guide to PLCs. Available at `http://www.plcmanual.com/` (2008).

[33] Friedemann Mattern and Christian Floerkemeier. From the internet of computers to the internet of things. *From active data management to event-based systems and more*, pages 242–259, 2010.

[34] Dave McMillen. Attacks Targeting Industrial Control Systems (ICS) Up 110 Percent. Available at `https://securityintelligence.com/attacks-targeting-industrial-control-systems-ics-up-110-percent/` (2016/12/27).

[35] Jeanne Meserve. Mouse click could plunge city into darkness, experts say. Available at `http://www.cnn.com/2007/US/09/27/power.at.risk/index.html` (2007/09/27).

[36] Rene Millman. Bluetooth and ZigBee to dominate wireless IoT connectivity. Available at `https://internetofbusiness.com/iot-driving-wireless-connectivity/` (2016/04/29).

[37] Andrew P Moore, Robert J Ellison, and Richard C Linger. Attack modeling for information security and survivability. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2001.

[38] Steve Morgan. Hackerpocalypse: A cybercrime revelation. Technical report, Cybersecurity Ventures, 2016.

[39] Karsten Nohl and Jakob Lell. Badusb-on accessories that turn evil. *Black Hat USA*, 2014.

[40] D O'Halloran and E Kvochko. Industrial internet of things: Unleashing the potential of connected products and services. In *World Economic Forum. Davos-Klosters, Switzerland*, 2015.

[41] Harry O'Sullivan. Security vulnerabilities of bluetooth low energy technology (ble). *Tufts University*, 2015.

[42] Metin Ozturk and Philip Aubin. Scada security: challenges and solutions. *Schneider Electric*, 2011.

[43] Thomas R Peltier. *Information security risk analysis*. CRC press, 2005.

[44] Postscapes. Internet of Things (IoT) History. Available at `https://www.postscapes.com/internet-of-things-history/` (2016/02/01).

[45] The Open Web Application Security Project. About The Open Web Application Security Project. Available at `https://www.owasp.org/index.php/About_The_Open_Web_Application_Security_Project` (2017).

[46] The Open Web Application Security Project. Top IoT Vulnerabilities. Available at `https://www.owasp.org/index.php/Top_IoT_Vulnerabilities` (2014).

[47] Steven Ragan. Here are the 61 passwords that powered the Mirai IoT botnet. Available at `http://www.csoonline.com/article/3126924/security/here-are-the-61-passwords-that-powered-the-mirai-iot-botnet.html` (2016/10/03).

[48] Dave Raggett. Tackling data security and privacy challenges for the internet of things. Technical report, World Wide Web Consortium, 2016.

[49] Reem Abdul Rahman and Babar Shah. Security analysis of iot protocols: A focus in coap. In *Big Data and Smart City (ICBDSC), 2016 3rd MEC International Conference on*, pages 1–7. IEEE, 2016.

[50] Bruce Schneier. Attack trees. *Dr. Dobb's journal*, 24(12):21–29, 1999.

[51] ITU Telecommunication Standardization Sector. Overview of the internet of things. Technical report, International Telecommunication Union, 2012.

[52] Russ Sevinsky. Funderbolt: Adventures in thunderbolt dma attacks. *Black Hat USA*, 2013.

[53] Howard Shrobe. Computational vulnerability analysis for information survivability. *AI Magazine*, 23(4):81, 2002.

[54] François-Xavier Standaert. Introduction to side-channel attacks. In *Secure Integrated Circuits and Systems*, pages 27–42. Springer, 2010.

[55] Keith Stouffer, Joe Falco, and Karen Scarfone. Guide to industrial control systems (ics) security. *NIST special publication*, 800(82):16–16, 2011.

[56] Microsoft Digital Crimes Unit. Microsoft digital crimes unit fact sheet. Technical report, Microsoft, 2016.

[57] Sean Wilkins. Wireless Security Considerations: Common Security Threats to Wireless Networks. Available at `https://www.pluralsight.com/blog/it-ops/wireless-lan-security-threats` (2011/11/02).

[58] Nicky Wolf. DDoS attack that disrupted internet was largest of its kind in history, experts say. Available at `https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet` (2016/10/26).

[59] Kim Zetter. Inside the Cunning, Unprecedented Hack of Ukraine's Power Grid. Available at `https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/` (2016/03/03).

[60] Zhenliu Zhou, Jiapeng Fan, Nan Zhang, and Rongsheng Xu. Advance and development of computer firmware security research. In *Proceedings of the 2009 International Symposium on Information Processing (ISIP;09) Huangshan, PR China*, pages 21–23, 2009.

[61] Bonnie Zhu, Anthony Joseph, and Shankar Sastry. A taxonomy of cyber attacks on scada systems. In *Internet of things (iThings/CPSCom), 2011 international conference on and 4th international conference on cyber, physical and social computing*, pages 380–388. IEEE, 2011.

[62] Hubert Zimmermann. Osi reference model–the iso model of architecture for open systems interconnection. *IEEE Transactions on communications*, 28(4):425–432, 1980.

[63] Steve Zurier. 5 Tips for Protecting Firmware From Attacks. Available at `http://www.darkreading.com/iot/5-tips-for-protecting-firmware-from-attacks/d/d-id/1325604` (2016/05/20).