Understanding the Security of Traffic Signal Infrastructure

Zhenyu Ning, Fengwei Zhang(✉), and Stephen Remias

COMPASS Lab, Wayne State University, Detroit, USA
{zhenyu.ning,fengwei,sremias}@wayne.edu

Abstract. With the proliferation of using smart and connected devices in the transportation domain, these systems inevitably face security threats from the real world. In this work, we analyze the security of the existing traffic signal systems and summarize the security implications exposed in our analysis. Our research shows that the deployed traffic signal systems can be easily manipulated with physical/remote access and are vulnerable to an array of real-world attacks such as a diversionary tactic. By setting up a standard traffic signal system locally in our lab and partnering with a municipality, we demonstrate that not only can traffic intersections be manipulated to show deadly traffic patterns such as all-direction green lights, but traffic control systems are also susceptible to ransomware and disruption attacks. Through testing and studying these attacks, we provide our security recommendations and mitigations to these threats.

1 Introduction

As cities and municipalities across the world look to employ smart and connected infrastructure technologies [36], we must remain vigilant of possible exploits against these systems. Traditionally, infrastructure systems have remained isolated from remote control, but as the ability for advanced monitoring and fine-tuned performance has become available, these systems are quickly becoming connected. Amid this emergence of connected systems, traffic signal systems have introduced large regional networks and operation centers to help alleviate vehicle traffic congestion. Traffic signal systems maintain the safety and coordination of all vehicles and pedestrians traversing public roads. Historically, these systems have proven themselves worthy. Governed by the Institute of Transportation Engineers (ITE) [22], the group has looked to excel in safety, education, and standardization of vehicle traffic intersections. The work of ITE has led to the general population trusting these systems and has delivered the expectation that a trip by vehicle or foot will be a safe journey.

Early implementations of traffic signal systems were based upon electro-mechanical controls. In the electro-mechanical systems of yesterday, the devices used nothing more than rotating gears and wheels that would spin and align contact leads to pass electricity to light bulbs contained in a traffic signal system [3].

R. Perdisci et al. (Eds.): DIMVA 2019, LNCS 11543, pp. 154–174, 2019.
https://doi.org/10.1007/978-3-030-22038-9_8

Simple enough, these devices worked but lacked any technology to provide real-time reconfiguration to allow for changes to accommodate ever-changing vehicle traffic flows.

Fast-forward some years, modern traffic signal systems have ushered in numerous technologies due to advancements in computing and the modern need for more efficient systems. With emerging smart cities [36], the new version of traffic signal systems have ushered in numerous advancements compared to the systems of yesterday. Featuring improvements such as Linux based operating systems and network architectures spanning hundreds of miles, intelligent transportation control systems have achieved a degree of efficient control over vehicle traffic that has long been sought after.

With new advancements that have been developed and deployed, it is critical that traffic signal systems be proven for safety and security above all. Previous security research of the traffic signal systems [13,14,20,26,27] mainly focus on the security of the traffic controllers and the wireless network deployed in the traffic signal system, and show that existing traffic systems are vulnerable. However, the security of the other parts (e.g., the Malfunction Management Unit and Cabinet Monitor Unit) in the traffic signal system are left out.

In this paper, we share our pathway and execution for finding and exploiting flaws found in traffic signal systems (e.g., specified by NEMA TS-2 [28] and ITS [23] Cabinet standards). Our work does not focus on a specific component, but instead analyze the security of the whole traffic signal system. Our analysis results show that an array of attacks can be easily launched by adversaries against these systems such as bypassing access controls, disabling monitoring alerts/units, manipulating traffic patterns, or causing denial of services. Moreover, we show that attackers can perform an all-direction greens attack against vehicle traffic signal systems. To the best of our knowledge, it is the first time that such a severe attack has been demonstrated. By setting up a standard traffic signal system locally in our laboratory and leveraging a traffic signal system laboratory in a municipality, we test and verify the effectiveness of all the presented attacks on typical traffic signal systems following the TS-2 and ITS standards. Furthermore, we provide our security recommendations and suggestions for the vulnerabilities and attacks we confirmed. Note that all the findings discovered in this paper was reported to the related municipality, and the municipality patched the vulnerable traffic cabinets and further notified the neighbour counties.

The main contributions of this work are:

- We present a comprehensive vulnerability analysis of vehicle traffic signal systems on both NEMA TS-2 and ITS Cabinet standards. Our analysis exposes a series of security implications in deployed traffic signal systems. With these implications, the attackers can easily gain physical/remote access to the system and manipulate the signal control devices.
- By setting up a standard traffic signal system locally in our lab and partnering with a municipality, we verify and demonstrate the proposed security implications and design different attacking scenarios including stealthy

traffic signal manipulation/control, ransomware deployment, and all-direct green lights.
– We provide our security recommendations and mitigations for the flaws that we confirmed. We would like to draw the attention of the Intelligent Transportation Systems community and governments to increase the awareness of critical cybersecurity concerns regarding the operation of vehicle traffic signal systems.

2 Background

2.1 Advanced Transportation Controller (ATC)

At the heart of an intersection is the Advanced Transportation Controller (ATC). The ATC makes logical decisions based on its inputs and configuration settings to implement traffic patterns. This configuration, based upon what is called the signal timing plan [25], holds parameters such as what duration to run which traffic patterns along with the minimum and maximum times to run the pattern. Then based upon an internal clock setting, the ATC decides which traffic pattern to run and if the pattern needs to be modified based upon its vehicle detection sensors. A large capability of the ATC is its ability to communicate over Ethernet using the NTCIP [4] protocol. The ability to communicate allows for the nearly continuous synchronization of the internal clock from a central server and the ability for transportation engineers to push new configurations on-demand.

According to the ATC standard [7] released by American Association of State Highway and Transportation Officials (AASHTO) [5], Institute of Transportation Engineers (ITE) [22], and National Electrical Manufacturers Association (NEMA) [29], the ATC is built upon a Linux kernel with BusyBox integration, supporting a capable networking stack and access to most typical Linux shell operations such as FTP and SSH. On top of the kernel, the actual control logic is left to the individual software running in the ATC, and the municipalities may use different software according to their specific requirements and existing infrastructure. While all software available offers similar functionality of controlling traffic signals, they defer in additional features that they offer. For instance, some software offers additional featuring for advanced monitoring and integration for connected vehicle communication.

The Linux kernel is selected in the ATCs to open up the traffic controller environment to open source development. This can be seen as beneficial as many manufacturers look to implement technologies such as traffic control programs, connected vehicle communication integration, and programs to monitor real-time traffic congestion. To further push the development of open source traffic control technologies, AASHTO, ITE, and NEMA released the Application Programming Interface Reference Implementation (APIRI) [6] to regulate the I/O control between the Linux kernel and applications running on top of the system.

2.2 Roadside Cabinets

The ATC is normally placed in a roadside cabinet. There are mainly two standards for the cabinet, i.e., the TS-2 standard [28] designed by NEMA and the Intelligent Transportation System (ITS) standard [23] developed by ITE.

The TS-2 Cabinet Standard [28] is a traffic signal cabinet standard that was initially commissioned by NEMA in 1998. The core feature of the modern TS-2 cabinet is its use of a single IBM SDLC serial bus for inter-device communications within the cabinet.

The ITS Cabinet standard [23] is designed to supersede the NEMA TS-2 standard. The standard's main contribution is the introduction of two SDLC serial buses for traffic signal control and monitoring. By effectively using two serial buses, the cabinet maintains separation between the control plane of the traffic signal's relays and the supervisory bus shared between the traffic controller unit and fail-safe unit. Since the control planes (failure handling, signal control, environmental sensing) is separated into different buses, the congestion and latency on the bus are reduced.

2.3 Malfunction Management Unit and Cabinet Monitor Unit

As specified in the NEMA TS-2 Cabinet specification, the Malfunction Management Unit (MMU) is designed to accomplish the detection of, and response to, improper and conflicting signals. Serving as a fail-safe unit and watchdog, the MMU monitors various parameters of the cabinet including the current state displayed on the signal light bulbs of an intersection. If an MMU detects that any monitoring parameter is out-of-range or in disagreement with the expectation, the MMU will override the control of the ATC, and the intersection is placed into a known-safe state called "conflict flash". Conflict flash is a state in that all intersection operations are halted and individual traffic signal will be instructed to strobe their red lights. This effectively leaves the intersection in an inoperable state and thus leaves all vehicle traffic to navigate at their own discretion. In order to return the operation of an intersection to a normal state, the MMU must be manually reset by a technician on-site. Upon reset, the MMU will hand signal control operation back to the ATC and the MMU itself will resume monitoring for faults.

The functionality of the Cabinet Monitor Unit (CMU) in the ITS cabinet is similar to the MMU in the TS-2 cabinet. The ITS Cabinet specification states that the minimum functionality of CMU is as least that provided by the NEMA TS-2 MMU. Additionally, the CMU offers enhanced monitoring and logging capabilities for items such as electrical voltages seen on cabinet peripherals, operating temperatures, and access controls.

To provide an example of an intersection state in which the MMU/CMU would trigger a conflict flash, consider a 4-way intersection consisting of two perpendicular roads. Imagine that the ATC was instructing all 4 directions of travel to pass through the intersection at the same time. The MMU/CMU would query the safety of the configuration displayed and cross-check the shown status

with its own configuration for permitted safe states of the intersection. If the ATC displayed configuration is found not to be permitted by the MMU/CMU, the MMU/CMU overrides the ATC and places the intersection into the conflict flash state. Thus, the MMU/CMU will not the let the ATC display traffic signal patterns which would pose a risk to the vehicles passing through. This is done as a fail safe to prevent ATC configuration mistakes by technicians.

2.4 MMU Programming Card and CMU Datakey

The configuration of the MMU relies upon an interchangeable programming card, and 120 pairs of 1.09 mm (0.043 in.) diameter holes on this card are used to configure the compatibility between traffic lights. The configuration is achieved by soldered wire jumpers, and the defined compatibilities are further used for conflict detection.

The fail-safe management of CMU is based on a configuration saved to what is called a "Datakey". The Datakey is an EEPROM memory device configured by a transportation engineer with a configuration that contains what parameters and states are valid for a respective vehicle traffic intersection. The Datakey is then inserted into the CMU unit which the configuration is read and then placed into operation. If a Datakey is not inserted into a CMU, the CMU will place the intersection into conflict flash [23].

3 Attack Surface Analysis

In this section, we analyze the security of existing vehicle traffic signal systems and summarize potential security implications. **Note that the summarized implications are based on the study in the partnering municipality, and they may also apply to other municipalities using the same device.**

3.1 Access to the Traffic Signal System

When breaching the perimeter to access traffic signal systems, an attacker will encounter both physical access and/or remote access restrictions. In the case of a network intrusion, an attacker will likely gain access to more than one ATC due to the uniform use of network restriction mechanisms. With a physical intrusion, an attacker would first need to breach a traffic signal cabinet or operation center, then proceed to escalate privileges through a regional transportation network. In this section, both access methods are discussed to provide a through pathway to regional traffic signal access.

(1) Physical Access
As mentioned in Sect. 2.2, the hardware devices in the traffic signal system are normally placed in a roadside cabinet. To avoid unauthorized access or destruction, the cabinet is protected by a Corbin #2 lock and key. This key is held by technicians who maintain the technology inside the cabinet. To assist with

physical monitoring, surveillance cameras may be deployed to monitor potential access to the traffic cabinet.

Cabinet Keys. According to the cabinet specifications [23,28], both the ITS and TS-2 cabinets shall be provided with a Corbin #2 Type key. Due to the large amount of deployed cabinets under these standards, we looked to verify this within our testing municipality. Through inquiry and testing, we verified that all of our testing municipalities traffic signal cabinets can be opened with the default Corbin #2 key.

With further research, we found that the Corbin #2 master key is sold online. For the price of $5 USD, the key is marked with the ability to open most traffic signal cabinets in the United States. Upon further examination, the purchased key was proven to be an exact match to the cabinets that are used by our partnering municipality and standards that we are investigating. This key would allow us to open all traffic signal cabinets deployed by the municipality.

Implication 1 : A large number of traffic signal cabinets can be opened with a Corbin #2 key purchased online.

Surveillance Cameras. Prior research has commented on the difficulty of beating surveillance cameras when gaining physical access to traffic cabinets [20]. However, our analysis shows a different result. In the municipality we investigated, there are 750 vehicle intersections. According to the municipality officials, only 275 vehicle intersections are covered by traffic cameras, which leaves more than 60% intersections of the traffic network unsurveilled. Without a surveillance camera, physical access to the traffic cabinets would be undetectable.

Implication 2 : Physical access to the traffic signal cabinets is out of watch of surveillance cameras in more than 60% *intersections of the investigated municipality.*

Door Status Monitoring. In the ITS cabinets, the status of the door can be monitored by the CMU [23]. Specifically, the ATC sends a Type 61 query command [23] to the CMU, and then the current status of the cabinet door is returned in the 31st byte of the response. In real-world deployments, we learn that the Model 2070 ATC [24], which is deployed in the investigated municipality, writes the door alarm message to log file, then after some time, the log file is forwarded to the parties who are monitoring the system. However, we are informed by our test municipality that the forwarding of the log files is kept to a low frequency (typically every one-to-five minutes) to reduce network congestion. This one-to-five minute gap offers a perpetrator a chance to clean up the log files before they get forwarded through the Model 2070's user interface. According to the test municipality, none of the cabinet door alarms are currently being monitored across the 750 vehicle intersections that they encompass.

Implication 3 : The door status of traffic signal cabinets may not be monitored in real-time or at all. Alarms may be cleared from the system by an attacker.

(2) Remote Access
As shown in previous work [20], a number of transportation systems use the insecure IEEE 802.11 wireless access points for network communications. The insecure wireless network would allow a perpetrator to remotely connect to a traffic network and access networked hardware inside.

While the network of some traffic signal systems is isolated from the Internet for security concerns, we do find that the public IP addresses of traffic signal systems are publicly accessible. The Shodan [31] website provides a search engine for internet-connected devices where reports can be generated containing IP addresses and signatures of devices meeting search criteria. With keywords such as `NTCIP` or `Econolite`, we are able to identify the IP address of a number of ATCs. Note that the keyword `Econolite` is traffic signal system manufacture who makes ATCs for ITS cabinets.

Additionally, due to the engineering efforts required for system updates, the Linux kernel in the ATCs is normally out-dated and is vulnerable to multiple existing attacks [15]. The ATCs used at our partnering municipality were confirmed to running the Linux 2.6.39 kernel network wide. Moreover, since the SSH/FTP connection is required in the ATCs [7], a perpetrator may also leverage known attacks [35] to gain access to the system. During our analysis, we found that both the deployed Intelight Model 2070 ATCs and Siemens Model 60 ATCs use default credentials for the SSH and Telnet connections. According to our partnering municipality, they were not aware of the ability to login to the ATC over SSH. This poses an interesting predicament as it appears that there may be additional municipalities that may not have an understanding that they are vulnerable to network attacks conducted via SSH.

Implication 4 : SSH connections to ATCs are possible via the publicly exposed IP addresses and default credentials.

3.2 Traffic Signal Control

As described in Sect. 2.1, the ATC is used to configure traffic signal patterns and timing. In the devices we investigated, the Intelight Model 2070 ATC uses D4 software [19] for configuration while the SEPAC software [33] is used in the Siemens Model 60 ATC. Since the ATCs follow the same standard [7], the basic functionalities of the different software are the same.

(1) With Physical Access
To reduce the complexity of using the software, the ATCs are equipped with a series of control buttons on the front panel. With the buttons and configuration menus, one can easily specify the configurations of the ATC including different traffic signal patterns, the internal clock, and the status of MMU/CMU.

In our investigation, we found out that the configuration of the ATC does not require authentication. In other words, it requires no credentials to access the front control panel of the ATC, that can be used to configure the ATC freely. While access codes can be set to control access to this front panel, our partnering municipality did not do so. Therefore, once physical access is gained to the ATC, a perpetrator may modify the configuration of the ATC without any restrictions.

(2) With Remote Access
In the ATC system, the D4 and SEPAC work as traffic control software in the Linux system. Naively, an attacker can gain remote access to the front panel controls by connecting into the Linux subsystem of controller. With the D4 software, an attacker that launches a connection will be displayed a remote terminal with the same controls that are offered on the Model 2070 front panel. With the SEPAC software an attacker can gain access to the front panel of by launching the front panel binary contained in the `/opt/sepac/` directory.

With remote access to the ATC via SSH, one can also control the traffic signals following the specification described in [7]. Specifically, the ATC is provided with seven serial communication ports, which are mapped as devices in the Linux `/dev` directory. According to the specification, Serial Port 3 (`/dev/sp3s`) and Serial Port 5 (`/dev/sp5s`) are used for in-cabinet device communications. Thus, directly writing a `Type 0` [28] command frame to the Load Switch relays achieves control of the traffic signal. To avoid conflict with the D4/SEPAC software, an attacker can stop the control software and their actions.

Similar to the aforementioned configuration with physical access, writing commands to the serial ports does not require any authentication in the investigated devices.

> *Implication 5 : The configuration of ATCs and the communication between the ATC and the traffic control signal do not require any authentication.*

3.3 Conflict Status Control

Recall that the MMU/CMU is in charge of detecting the conflict between the ATC configuration and predefined forbidden patterns. The forbidden patterns in the MMU and CMU are specified by the Programming Card and Datakey, respectively. Thus, to control the conflict status, a perpetrator needs to override the configuration in the Programming Card or Datakey.

(1) MMU Programming Card
The conflict status on the MMU is defined by the compatibility between channels on the Programming Card [28]. Configuration is accomplished through the use of soldered wire jumpers. Therefore, to override the configuration, the perpetrator needs to resolder the wire jumpers to specify the required status.

(2) CMU Datakeys
According to the specification [23], the CMUs in ITS cabinets use the LCK4000 Datakey [11]. We find that the LCK4000 is a 4 KB serial EEPROM memory chip molded into a plastic form-factor resembling a house key. Designed and manufactured by ATEK Access Technologies [9], the Datakey serves as an unencrypted configuration storage unit for the CMU that includes the known safe-states for an intersection housed in a defined byte-array [23]. Located on the ATEK Access Technologies website, we find that the company offers memory flashing devices for the LCK4000, and also instructions for making your own reader and writer based upon the Microwire serial communication protocol [10].

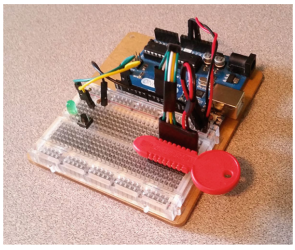

Fig. 1. The Datakey LCK4000 Microwire Flasher built upon the Arduino Platform. The red key is the LCK4000 Datakey. (Color figure online)

To configure the Datakey, we would be able to buy an EEPROM memory flashing unit directly from ATEK. However, to learn the bar of overriding the configuration, we built a customized Datakey access tool by using an Arduino Uno starter-kit [8]. Following the Microwire serial protocol specification [10] found on ATEK's website, we are able to construct our own flashing device as shown in Fig. 1. Similar to the control of traffic signals, the configuration of the Datakey requires no authentication, and our simple device would allow us to read and write configurations on-demand without any restriction.

Implication 6 : The configuration of the conflict status control does not require any authentication.

3.4 Troubleshooting of the Traffic Signal System

Wireless 802.11 deployments in traffic networks are generally linear in communication flows. That is, due to the geography that must be covered in these

networks, the use of redundant protocols such as spanning-tree is not seen due to the extra cost needed to design and install additional equipment. If there are no redundant loops in the network architecture, one can easily disable network communications across a linear communication chain by disabling an upstream communication node (i.e., *an intersection*). Thus, each wireless network connection can be seen as a dependency to its parent station as we work our way further from the centrally headquartered location.

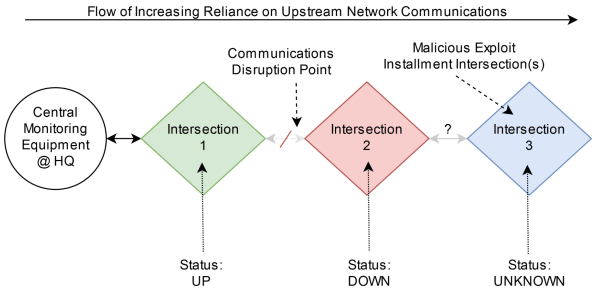

Fig. 2. Diversionary cabinet access tactic. The circle on the left most represents the central headquarter. An attacker can disable the communication between intersections 1 and 2, and conduct the malicious exploitation at intersection 3 where is a few miles away.

Consequently, a diversionary tactic would seriously affect the troubleshooting process of the traffic signal system. For example, one would covertly or explicitly break upstream network communications, thus leaving downstream traffic intersections with no network access to the rest of the traffic network. This would disable any sort of central monitoring including surveillance cameras and cabinet door alarms as there would be no network path to these devices. Figure 2 shows the diversionary cabinet access tactic.

To achieve the needed disruption to the network, one of the methods is to use radio frequency jamming techniques since the wireless 802.11 equipment is widely used to connect vehicle traffic networks [20]. As shown in works by Grover et al. [30] and Pelechrins et al. [21], 802.11 networks can be completely or selectively jammed to block communications between end devices. The use case for us would be to disrupt the communications pathway between a selected vehicle intersection and the traffic control master server located miles away. In reality, our partnering municipality informs us that network outages are already a common occurrence due to the interference generated by the deployment of wireless 802.11 access points in homes and business that are near traffic intersections. In short, they would likely disregard our radio frequency disruption and jamming attacks as another common case of co-channel interference.

To traffic system monitoring staff, the statuses of intersections that lay on the other side of the network disruption or breakage would fundamentally be unknown due to the lack of network communications to the downed intersections. It is at this point that an attacker would access one of the downstream traffic cabinets with an unknown status. Throughout the period of unknown status, the attacker would have completely unmonitored access to the cabinet.

At some point, the municipality will have to troubleshoot the outage. We learn through our partnering municipality that the troubleshooting process could occur anywhere between instantaneously and 64 h (if the attack is orchestrated outside of normal business hours during the weekend). Upon inspection, the maintenance staff would focus on the direct location of the network outage itself and not any of the unknown status intersections behind the disrupted connection. Once they managed to resolve the disruption at the first disrupted intersection, it is unlikely that they would investigate any of the previously unknown status intersections if all network communications return to normal.

Implication 7 : The troubleshooting process of the real-world traffic signal systems makes it possible for the attacker to achieve stealthy access/control to the system.

4 Attacks Implementation and Testing

To learn the impact of the implications discussed in Sect. 3, we have crafted several attacking scenarios in which we test with our partnering municipality.

4.1 Environment Setup

We first partner with a local municipality to gain access to their traffic signals test lab, which is equipped with ITS cabinets, Intelight Model 2070 ATCs [24], and CMU-212 [16]. This lab is a mock-up of their operational traffic network and is used for their own testing and burn-in of equipment before deploying the devices to the field. The devices that were used in the lab are shown in Fig. 3. The Intelight Model 2070 ATC is running the Linux 2.6.39 kernel as specified by ATC standard [7].

Moreover, we obtain a TS-2 cabinet and set up an environment that fulfills the NEMA standard. In this cabinet, the widely used Siemens Model 60 ATC [32] and EDI MMU-16LE [17] are deployed. The entire traffic signal system is shown in Fig. 4 Like the Intelight Model 2070 ATC, the Siemens Model 60 also runs upon a Linux 2.6.39 kernel as specified by the ATC Standard.

Fig. 3. Traffic Signal System in the municipality test lab. The left side shows a group of Model 2070 ATCs. The right top of the figure shows the traffic signal bulbs while the right bottom of the figure shows the CMU-212.

4.2 Thread Model

We assume the target traffic signal system follows the ITS cabinet standard or the TS-2 cabinet standard. In both standards, we assume the ATC deployed inside the cabinet follows the ATC standard released by AASHTO, ITE, and NEMA. In our attack, we assume the access to the traffic signal system is gained via *Implications 1-4*. Specifically, in most scenarios, we only require the remote access achieved by *Implication 4*. In the all-direction green light attack, we provide two different attacking policies with physical access gained by *Implications 1-3* and remote access gained by *Implication 4*, respectively.

4.3 Attack Scenarios

(1) Stealthy Manipulation and Control
As demonstrated in previous research [14,20,26], the monitoring and control of the traffic signal system could be used in a series of attacks such as Denial of Service (DoS) and causing traffic congestion. However, previous attack approaches control the traffic signals by either changing the configuration of the ATC or by injecting messages to the transportation system that are easy to be detected. For example, the transportation engineers may simply pull the configuration of the ATC remotely to identify the abnormal configuration.

In our attack, we achieve a stealthy manipulation and control via intercepting the communication between the ATC and Load Switch relays that control the traffic signal lights. As discussed in Sect. 3.2 and *Implication 5*, the in-cabinet device communication between the ATC and traffic lights is performed via the serial port `/dev/sp3s` and `/dev/sp5s`, and the communication requires no authentication mechanism. To monitor and manipulate the communication, we replace the driver of these two devices in the system with a customized driver, and the customized driver records/modifies the message sent to the serial port before it is transmitted to the hardware.

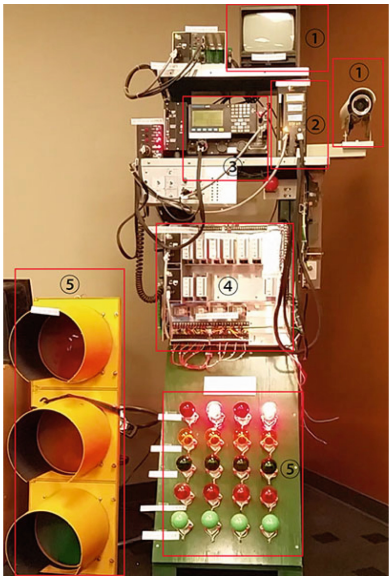

Fig. 4. Our Traffic Signal System of TS-2 Standard Equipment. The ① is a vehicle detection and surveillance system. The ② shows the MMU-16LE while the ③ shows the Siemens Model 60 ATC. The ④ and ⑤ indicate the Load Switch relays and traffic signal bulbs, respectively.

With the customized driver, our attack is launched with a stealthy style since it modifies no configuration of the ATC and involves no additional messages. For example, we can increase the duration of the red light to introduce a traffic congestion. More serious congestion would be caused if we place all the traffic signals into a flashing red style. Even worse, malicious signal patterns such as all-direction flashing yellow may spark a critical accident. According to *Implication 7*, existing troubleshooting process of the traffic signal system can hardly detect the attack if the malicious traffic signal pattern is carefully designed.

(2) Ransomware Deployment

One of the most crippling scenarios for a traffic network is the deployment of ransomware across all traffic control devices contained in the network. Using methodologies as described in Sect. 3, we design the most simplistic path towards a ransomware deployment across a traffic network. In this scenario, all ATCs on a traffic control network would have their internal traffic control program processes disabled and all root login access to the internal Linux operating system would be denied, thus, each ATC would be held at ransom.

As specified by the ATC specification [7], the ATC stores all startup instructions in the Linux `/etc/inittab` daemon file. While investigating this file, we found an instruction to launch a shell script file that handles setting up the runtime environment and processes for the traffic control software. If one is to remove this shell script file, it completely disables the traffic control software

from launching thus leaving its respective intersection uncontrolled. Rebooting the ATC devices will not resolve the issue, and the only way to resume the traffic control software is to replace the correct script that is responsible for launching the traffic control software. To further consolidate the attack, we can change the credentials of the SSH connection to prevent the transportation engineers from accessing the ATC system.

To extend the attack, we launch a ransomware deployment Python script in the partnered lab, which includes a large number of Intelight 2070 ATCs on the test traffic network. With the script, we are able to make a list containing IPs of all known ATCs on the traffic network then deploy our ransomware engagement shell commands issued over SSH.

The Destruction. In the network of the road agency that we partnered with, this exploit would allow us to take control of 400 ATCs running the known traffic control software. To fix a ransomware affected traffic signal system at an intersection, a transportation engineer would need to drive to the intersection and physically update the firmware of the ATC. If we assume that it will take 1 h to fix each ATC (it might take more time because of the traffic congestion and none of the ATC traffic signals operating), and assuming that 10 workers have the expertise to do this. The time for resuming the complete traffic signal system would be: 400 controllers * 1 h/10 engineers = 40 h/engineer = 1 week (if an engineer works 8 h/day). If we assume that an engineer is paid $40/h, the estimated cost for fixing this would be 400 controllers * 1 h * $40/h = $16,000 USD. A previous study [12] also shows that simply reconfiguring the timings of 60 intersections in one district of Boston could save $1.2 million per year. Additionally, we also identify that many states (e.g., California, Florida, Michigan, Missouri, Ohio, Oregon, South Carolina, Texas, Virginia, Wisconsin, etc.) currently use the 2070 ATC [1,2,34], which means that our attacks might be deployed in these states as well.

(3) All-Direction Green Lights

When considering the most dangerous state for an intersection, we conceived the idea of all-direction green lights. An intersection displaying green lights in all directions would leave drivers defenseless to vehicle cross traffic traveling at speed as they passed through. In order to make this happen, one would have to override the fail-protection of the MMU/CMU, then program the all-direction green light pattern into the traffic controller. We chose to investigate this possibility heavily as the MMU/CMU was not shown to be tested in previous work.

The MMU/CMU plays the role of policing traffic patterns shown by the ATC. If a traffic pattern is displayed that would be dangerous, such as all-direction greens, the MMU/CMU steps in and places the intersection into conflict flash. Furthermore, if serial communication fails amongst any of the traffic cabinet's devices, the intersection is placed into conflict flash. This made for a difficult process as any event that placed the cabinet ecosystem out-of-balance would trigger a conflict flash state. Due to the differences in attack policies, we discuss this attack with and without physical access, respectively.

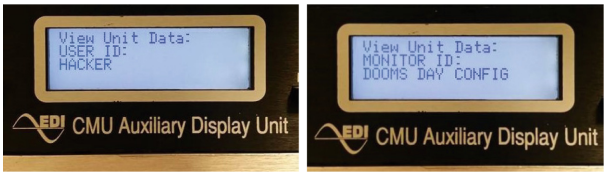

Fig. 5. CMU-212 display unit showing the Datakey configuration for USER ID and MONITOR ID which was written using the home-made Arduino Datakey writer to allow for full-permissive configuration.

With Physical Access. As discussed in Sect. 3.3, the configuration data such as unsafe states is defined by the Programming Card and Datakey in MMU and CMU, respectively. To overcome accidentally triggering conflict states, we can directly override the configuration of the MMU/CMU with physical access according to *Implication 6*. Since the configuration of the Programming Card is simply achieved by soldered wire jumpers, here we only show how to override the configuration in the CMU Datakey.

While we find the CMU's specification for the address layout and configuration parameters for the Datakey in the ITS Cabinet Specification [23], we believe that the parameter selection would be difficult for someone without traffic device configuration experience. In order to combat this, we look for configuration generation programs on the CMU manufacturers website. It does not take us long to find one as we quickly discovered a free program [18] offered which would allow us to create the configuration files for the Datakey using a wizard-style approach. This wizard would handle parameter setup, leaving us to only to configure the nullification of conflict states for the intersection which was as simple as selecting a group of checkboxes called `permissives`. A `permissive` is a setting that which specifies what individual traffic signal light bulb is permitted to be turned on with each other light bulb of the intersection. This is done as a method to prevent two cross-directions of travel from receiving concurrent green lights which would cause passing vehicles to enter a potentially dangerous situation. If two signal light bulbs try to turn on that is not set to be permissive with each other, the CMU will engage and place the intersection into conflict flash. After using the key generation program to generate a configuration file allowing for all-direction green permissives, we use our Arduino Uno LCK4000 flasher to write the configuration to the key as shown in Fig. 1. Figure 5 shows the CMU Datakey configuration on a display screen. This Datakey is written via the home-made Arduino writer using our generated key file.

The last step in configuring all-direction greens lights is to place the correct settings in the ATC traffic control program. However, during our experiment, it is discovered that the Intelight Model 2070 ATC must maintain nearly constant contact with the CMU over serial communications, and this contact periodically shares the configuration of the LCK4000 Datakey and the ATC with each other. If the configurations do not match, the CMU will trigger a conflict flash. To combat this issue, the traffic controller must be configured to match the all-direction green permissive configuration on the CMU.

Fig. 6. All-direction green lights being displayed on traffic signal test equipment. The left 4 green LEDs represent the through directions of travel at an intersection while the right 4 green LEDs represent the corresponding left-turn lanes. (Color figure online)

In order to set up the ATC with a matching configuration to the CMU, all that we required is the front panel controls and display screen located directly on the unit. Navigating through the front panel menu controls, we find that the traffic control software features a similar parameter setup to what we saw on the CMU. In this menu, we are able to explicitly state the permissives of the intersection then construct an all-directions green traffic pattern. We are then able to schedule for an all-directions green pattern to run in another menu. Shortly after scheduling the pattern to run and waiting for the transition to occur, we are greeted with the all-directions green configuration. A test displaying all-direction greens is shown in Fig. 6.

With Remote Access. Although the configuration of the ATC could be modified via remote access, the aforementioned approach requires physical access to reconfigure the unsafe states of MMU/CMU. Since the configuration in devices like the MMU programming card is achieved by soldered wire jumpers, it would be difficult to override the configuration without physical access.

To bypass the fail-safe units, we implement an attack called the transient avoidance attack tactic. The root feature of this attack is that the fail-safe unit does not trigger a conflict state until conflicting control signals exist for 200 ms or greater. Following this 200 ms wait period, the fail-safe requires up to an additional 300 ms to place the intersection into a conflict state (all-directions flashing red lights). Figure 7 shows the details of the transient attack. From the top, the first line and second lines represent the on/off signal of two green lights that conflict at an intersection. The third line represents the presence on a conflict situation. The fourth line displays if an intersection has entered a conflict flash failure state. The state designation, seen on the bottom of the graphs, is described as the following: (1) An intersection is in a conflict free running state; (2) A conflict has occurred. The conflicting signals must exist for 200 ms before triggering a conflict state; (3) Conflicting signals have been shown for more than 200 ms. In the next 300 ms timespan, the fail-safe unit must place the intersection into a conflict flash state; (4) The intersection is currently in the conflict flash state.

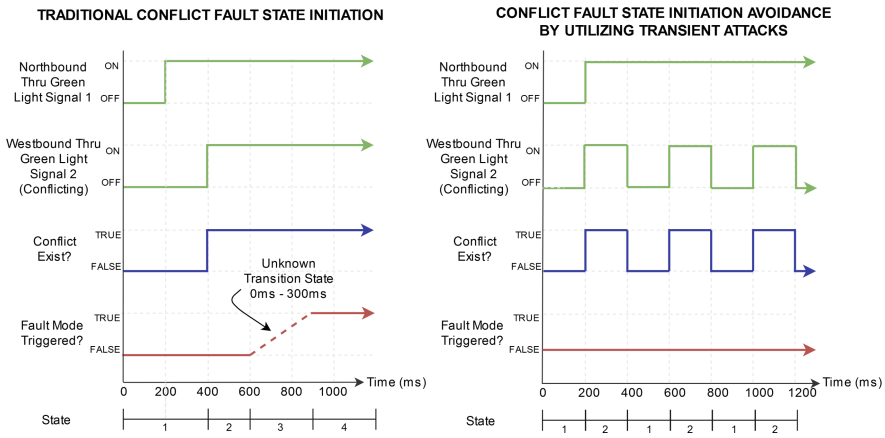

Fig. 7. Comparison of a typical all-directions green conflict flash state initiation versus transient avoidance attack tactic. (Color figure online)

Another challenge that we will have overcome is the fail-safes' use of Recurrent Pulse Detection (RPD) [16,17]. This mechanism is used to detect failures resulting in voltage leaks from a traffic signal's Load Switch relays. This mechanism looks for voltage leaks lasting 1 to 200 ms and triggers a conflict flash state if they meet a certain criteria level in regards to power, duration, and frequency. In practice, our experiment shows that the RPD mechanism will not trigger a fault if an off time of 24 ms or greater duration is used to separate conflicting signals such as in the procedure of triggering each green light bulb during the all-directions green attack. Note that the transient attack places the traffic lights into a flicker status. Considering the high flicker frequency, the influence of the real-world ambient light, and the long distance between the real-world traffic lights and the drivers, the flickering green lights are likely to be recognized as constant green lights.

5 Recommendations

To mitigate the security implications, we propose recommendations for the existing and future traffic signal systems. While some of the recommendations can be implemented by the municipality directly, many fixes, rethinks, and implementations will have to come from at the discretion of manufacturers and organizations that design and manufacture traffic signal equipment.

5.1 Default Passwords and Master Keys

While the standardization of passwords and physical keys comes with a convenience factor for parties involved, organizations must question to what extent is the standardization worth the risk. The default ATC SSH credential is what

opened up the door for our region-wide deployment of ransomware. Likewise, cabinet master keys allowed us to confidently know we had access to any traffic cabinet to implement all-direction greens. Both of these vulnerabilities are what quickly allows a small attack to grow quite large. By using different passwords and door keys, an organization can significantly lower their attack surface. Thus, an immediate recommendation to any agency operating the Model 2070 ATC and Model 60 ATC is to change the credential of the SSH connection.

5.2 Open Access to Traffic Specifications

We applaud the transportation industry's push to publish open specifications. Open specifications allow for professionals and communities across all fields to verify, correct, and improve designs. Where this process goes wrong is with its cybersecurity implications. An example would be the Datakey exploit. While this may have been an acceptable strategy behind closed doors, the moment that this information becomes public available, it becomes vulnerable. However, this does not mean that open information increases the attacking probability. The root problem is that these specifications need to be designed with security in mind. Instead, we encourage the industry to move forward with standards such as the APIRI [6], but we ask that they scrutinize components that may allow for malicious exploit. Furthermore, we ask that they reach out to professionals in the cybersecurity field to help implement protocols that can be shared in the public domain without increasing the attack surface of infrastructure.

5.3 Redaction of Software Distribution

Though software tools are needed to help municipalities and their technicians perform maintenance and configurations, this software should not be provided freely and openly on the Internet. By us having open access to these tools, such as the Datakey key file creator, we were easily able to reverse engineer critical configurations and safety components. To combat this problem, manufacturers should distribute these pieces of software with newly purchased equipment or through online access portals in which download recipients have to be registered and verified. Moreover, we would ask that policies and agreements should be strictly enforced between Original Equipment Manufacturers (OEMs) and third-party companies for the software and specification distribution.

6 Related Work

Previous work [20] investigated the security of vehicle traffic signal systems. In their analysis, the researchers identified vulnerabilities about the deployed wireless network and operating system of the traffic controller. Exploiting these vulnerabilities, the researchers were able to control intersections on-demand to give them the ability to completely manipulate vehicle traffic progression. Cerrudo [13] presented vulnerabilities on the wireless sensors of the vehicle traffic

signal systems. These vulnerabilities allow attackers to take complete control of the devices and send fake data to vehicle traffic signal systems. By leveraging these flaws, adversaries can cause traffic jams in a city. Laszka *et al.* [26] developed a method for evaluating the transportation network vulnerability, and their method is tested on randomly generated and real networks. Their approach can further identify critical signals that affect the congestion. Li *et al.* [27] presented risk-based frameworks for evaluating the compromised traffic signals and provided recommendations for the deployment of defensive measures in the vehicle traffic signal systems. [14] focuses on the vulnerability of the I-SIG system and shows that traffic congestion could be introduced by data spoofing attack from even a single attack vehicle. Unlike these work, we target the ATCs featuring two standards (i.e., ITS and TS-2) and advance their work in the following aspects: (1) We analyze the security of the entire traffic signal system in both ITS and TS-2 standards and summarize the security implications; (2) we show that stealthy manipulation to the traffic signal system is feasible via a diversionary cabinet access tactic; (3) we demonstrate the feasibility of the all-direction greens attack via bypassing the MMU/CMU.

7 Conclusion

In conclusion, we presented a comprehensive vulnerability analysis of the vehicle traffic signal systems with both ITS and TS-2 standards. By leveraging these vulnerabilities, attackers can conduct a variety of attacks against vehicle traffic signal systems such as the region-wide deployment of ransomware. Moreover, to our best knowledge, our work is the first one to demonstrate the all-direction greens attack via bypassing the MMU/CMU. In our experiments, we test and verify the designed attacks in our lab and the municipality's test lab. We provide our security recommendations and mitigation plans for addressing these threats. Furthermore, we would like to raise the attention in the transportation community for the critical cybersecurity threats against the vehicle traffic signal systems.

Acknowledgements. This work is partially supported by the National Science Foundation Grant No. IIS-1724227. Opinions, findings, conclusions and recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the US Government.

References

1. 2070 ATC (Advanced Transportation Controllers) Equipment Parts and Accessories. http://esbd.cpa.state.tx.us/bid_show.cfm?bidid=139594. Accessed 31 May 2017
2. Search Federal, State, and Local Government Contracts, Government Bids, and RFPs. https://www.bidcontract.com/government-contracts-bids/search-government-Bids-Contracts.aspx?s=2070&t=FE&is=0. Accessed 31 May 2017

3. The Traffic Signal Museum: Eagle Signal Model EF-15 Traffic Controller (2015). http://www.trafficsignalmuseum.com/pages/ef15.html
4. 1103 v03 - NTCIP Transportation Management Protocols (TMP) v03 (2017). https://www.ntcip.org/library/standards/default.asp?documents=yes&qreport=no&standard=1103%20v03
5. AASHTO: American association of state highway and transportation officials. https://www.transportation.org/
6. AASHTO, ITE, and NEMA: Advanced transportation controller application programming interface reference implementation software user manual (2015). https://www.ite.org/pub/?id=31058d5b-ccfb-5b00-30d1-61c715ada9a4
7. AASHTO, ITE, and NEMA: Advanced Transportation Controller (ATC) standard version 06 (2018). https://www.ite.org/pub/?id=acaf6aca-d1fd-f0ec-86ca-79ad05a7cab6
8. Arduino: Starter kit. https://store.arduino.cc/usa/arduino-starter-kit. Accessed 1 Dec 2018
9. ATEK Access Technologies: Access the power of technology. http://atekcompanies.com/access-technologies. Accessed 31 May 2017
10. ATEK Access Technologies: Datakey microwire protocol specification (2014). http://datakey.com/downloads/223-0017-003_REVI_MWInterfaceSpec_SBM.pdf
11. ATEK Access Technologies: Datakey LCK series specification sheet (2015). http://datakey.com/downloads/LCK_Series_DS_REV.D.pdf
12. Boston Transportation Department: The benefits of retiming/rephasing traffic signals in the back bay. https://www.cityofboston.gov/images_documents/The%20Benefits%20of%20Traffic%20Signal%20Retiming%20Report_tcm3-18554.pdf. Accessed 1 Dec 2018
13. Cerrudo, C.: Hacking US (and UK, Australia, France, etc.) Traffic Control Systems. IOActive Blog (2014)
14. Chen, Q.A., Yin, Y., Feng, Y., Mao, Z.M., Liu, H.X.: Exposing congestion attack on emerging connected vehicle based traffic signal control. In: Network and Distributed Systems Security (NDSS) Symposium 2018 (2018)
15. Cozzi, E., Graziano, M., Fratantonio, Y., Balzarotti, D.: Understanding Linux malware. In: IEEE Symposium on Security & Privacy (2018)
16. Eberle Design, Inc.: CMU-212. https://www.editraffic.com/wp-content/uploads/888-0212-001-CMU-212-Operation-Manual.pdf. Accessed 1 Dec 2018
17. Eberle Design, Inc.: MMU-16LE series SmartMonitor. https://www.editraffic.com/wp-content/uploads/888-0116-001-MMU-16LE-Operation-Manual.pdf. Accessed 1 Dec 2018
18. Eberle Design, Inc.: Traffic control software. https://www.editraffic.com/support-traffic-control-software/. Accessed 1 Dec 2018
19. Fourth Dimension Traffic: The D4 traffic signal controller software. https://fourthdimensiontraffic.com/about/about.html. Accessed 1 Dec 2018
20. Ghena, B., Beyer, W., Hillaker, A., Pevarnek, J., Halderman, J.A.: Green lights forever: analyzing the security of traffic infrastructure. In: 8th USENIX Workshop on Offensive Technologies (WOOT 2014). USENIX Association, San Diego (2014). https://www.usenix.org/conference/woot14/workshop-program/presentation/ghena
21. Grover, K., Lim, A., Yang, Q.: Jamming and anti-jamming techniques in wireless networks: a survey. Int. J. Ad Hoc Ubiquit. Comput. **17**(4), 197–215 (2014)
22. Institute of Transportation Engineers: About the institute of transportation engineers. http://www.ite.org/aboutite/index.asp

23. Institute of Transportation Engineers: Standard specification for roadside cabinets (2006). https://www.ite.org/pub/E26A4960-2354-D714-51E1-FCD483B751AA
24. Intelight: 2070 ATC controllers. https://www.intelight-its.com/product-categories/2070-type-controllers/. Accessed 1 Dec 2018
25. Koonce, P., et al.: Traffic signal timing manual. Technical report (2008)
26. Laszka, A., Potteiger, B., Vorobeychik, Y., Amin, S., Koutsoukos, X.: Vulnerability of transportation networks to traffic-signal tampering. In: Proceedings of the 7th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS 2016), pp. 1–10. IEEE (2016)
27. Li, Z., Jin, D., Hannon, C., Shahidehpour, M., Wang, J.: Assessing and mitigating cybersecurity tisks of traffic light systems in smart cities. IET Cyber-Phys. Syst. Theory Appl. **1**(1), 60–69 (2016)
28. National Electrical Manufacturers Association: Standards publication TS 2-2003 (2003). https://www.nema.org/Standards/ComplimentaryDocuments/Contents%20and%20Scope%20TS%202-2003%20(R2008).pdf
29. NEMA: National electrical manufacturers association. https://www.nema.org/pages/default.aspx
30. Pelechrinis, K., Iliofotou, M., Krishnamurthy, S.V.: Denial of service attacks in wireless networks: the case of jammers. IEEE Commun. Surv. Tutor. **13**(2), 245–257 (2011)
31. Shodan: Search engine for Internet-connected devices. https://www.shodan.io/. Accessed 1 Dec 2018
32. Siemens: M60 series ATC. https://w3.usa.siemens.com/mobility/us/en/road-solutions/Documents/m60%20Series%20ATC%20Data%20Sheet%20FINAL.pdf/. Accessed 1 Dec 2018
33. Siemens: SEPAC local controller software. https://w3.usa.siemens.com/mobility/us/en/road-solutions/Documents/SEPAC%20Local%20Controller%20Software.pdf. Accessed 1 Dec 2018
34. Spencer, D.: The Advanced Transportation Controller and Applications for Oregon Department of Transportation (2013). https://www.oregon.gov/ODOT/HWY/TRAFFIC-ROADWAY/docs/pdf/2013_conference/ATCforODOT.pdf. Accessed 31 May 2017
35. The MITRE Corporation: Common vulnerabilities and exposures. https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=SSH. Accessed 1 Dec 2018
36. The White House: FACT SHEET: Announcing Over $80 million in New Federal Investment and a Doubling of Participating Communities in the White House Smart Cities Initiative (2016). https://obamawhitehouse.archives.gov/the-press-office