Caught in the Act: Running a Realistic Factory Honeypot to Capture Real Threats

Stephen Hilt, Federico Maggi, Charles Perine, Lord Remorin, Martin Rösler, and Rainer Vosseler

TREND MICRO™ | research

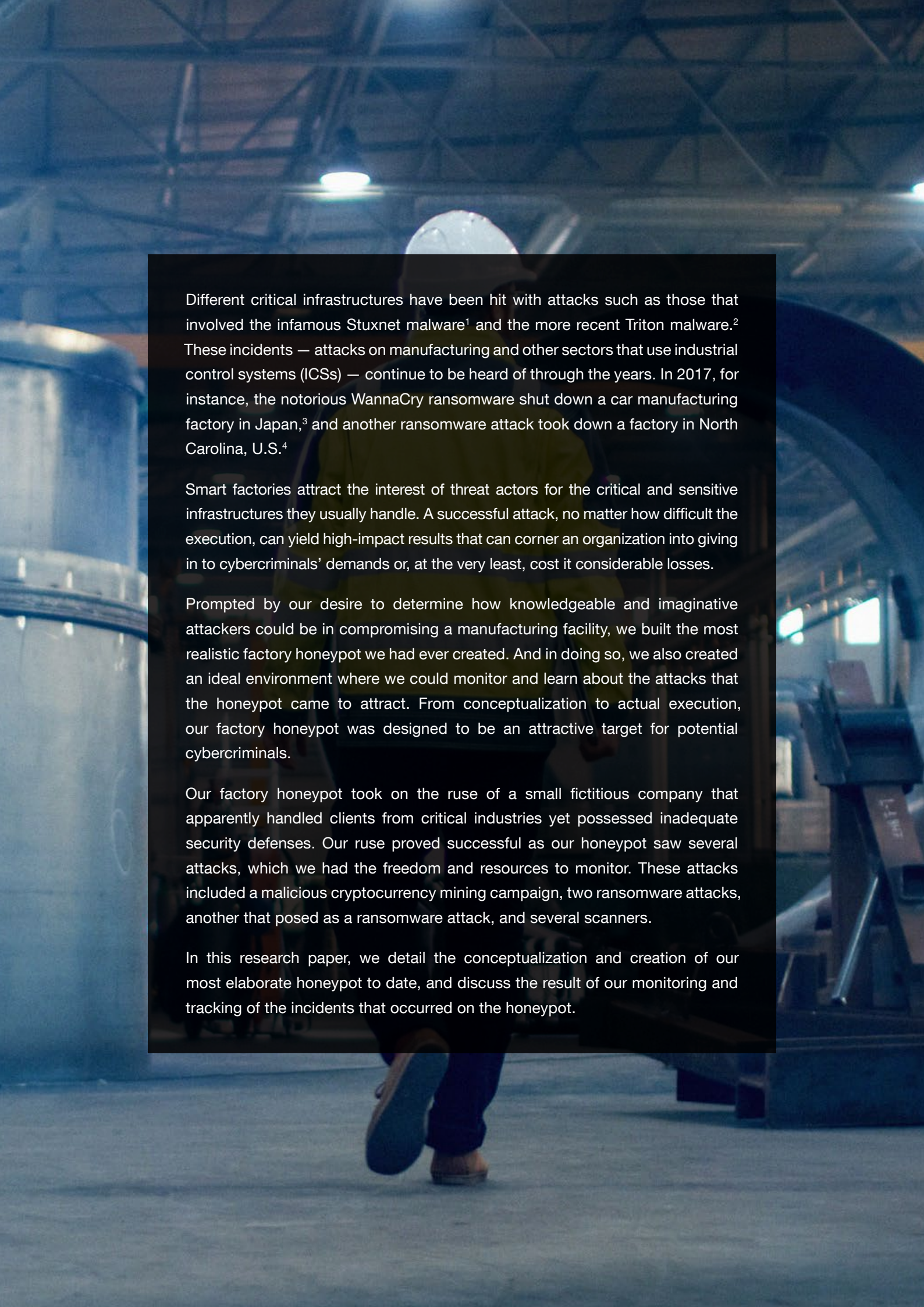
Published by
Trend Micro Research

Written by
Stephen Hilt, Federico Maggi, Charles Perine, Lord Remorin, Martin Rösler, and Rainer Vosseler

Stock images used under license from Shutterstock.com

For Raimund Genes (1963-2017)

Contents



Different critical infrastructures have been hit with attacks such as those that involved the infamous Stuxnet malware[1] and the more recent Triton malware.[2] These incidents — attacks on manufacturing and other sectors that use industrial control systems (ICSs) — continue to be heard of through the years. In 2017, for instance, the notorious WannaCry ransomware shut down a car manufacturing factory in Japan,[3] and another ransomware attack took down a factory in North Carolina, U.S.[4]

Smart factories attract the interest of threat actors for the critical and sensitive infrastructures they usually handle. A successful attack, no matter how difficult the execution, can yield high-impact results that can corner an organization into giving in to cybercriminals' demands or, at the very least, cost it considerable losses.

Prompted by our desire to determine how knowledgeable and imaginative attackers could be in compromising a manufacturing facility, we built the most realistic factory honeypot we had ever created. And in doing so, we also created an ideal environment where we could monitor and learn about the attacks that the honeypot came to attract. From conceptualization to actual execution, our factory honeypot was designed to be an attractive target for potential cybercriminals.

Our factory honeypot took on the ruse of a small fictitious company that apparently handled clients from critical industries yet possessed inadequate security defenses. Our ruse proved successful as our honeypot saw several attacks, which we had the freedom and resources to monitor. These attacks included a malicious cryptocurrency mining campaign, two ransomware attacks, another that posed as a ransomware attack, and several scanners.

In this research paper, we detail the conceptualization and creation of our most elaborate honeypot to date, and discuss the result of our monitoring and tracking of the incidents that occurred on the honeypot.

Conceptualization

Trend Micro had already created several honeypots, specifically ones that ran ICSs. In 2013, Trend Micro released a research that centered on a honeypot for a water system.[5] This research was focused on a pure-production honeypot that mimicked a real system, including a human-machine interface (HMI) and other components of an ICS. And in 2015, Trend Micro released research around the Guardian AST monitoring system using a honeypot called GasPot,[6] which simulated a gas tank monitoring system. The purpose of this honeypot was to deploy multiple unique systems that did not look the same but nonetheless responded like real deployed systems.

We evolve our honeypots by making them more and more realistic each time we build them. This is why for this research we wanted to build a honeypot that not only mimicked a real system but could also start making products. The goal was to build a honeypot that appeared so real that not even a well-trained control systems engineer would be able to tell that it was fake without diving deeply into the system.

First, we decided on what services and ports would be exposed to the internet to make our honeypot attractive to attackers. At the same time, we would maintain a minimal number of exposed services to prevent our honeypot from being identified as such. Second, we created a backstory for our fictitious company, which included made-up employee names, working phone numbers, and email addresses — anything and everything that a real company would need to run a day-to-day business. Third, we created a strategy to build the factory with equipment we already owned and equipment we needed to purchase.

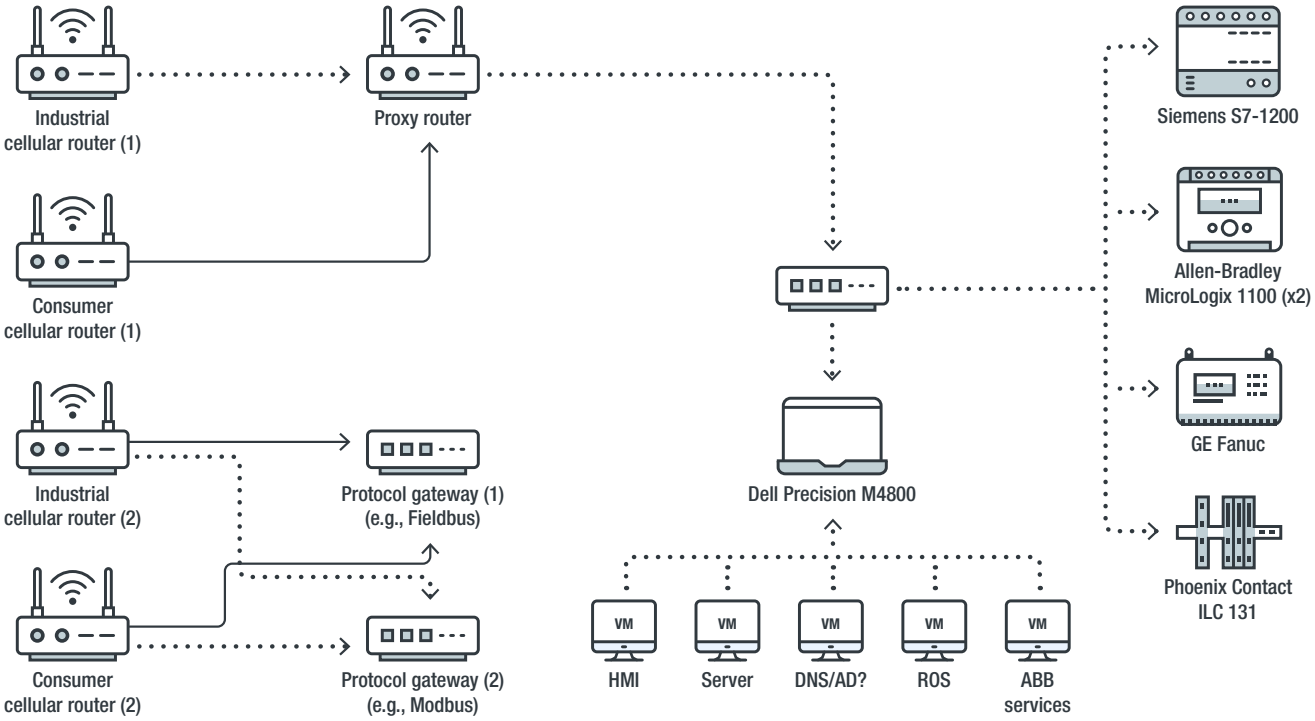


Figure 1. Our original layout plan

Looking at the equipment that we had and what we wanted to achieve, we replaced the Modbus programmable logic controllers (PLCs) with ones from Omron. This was to see if there would be any other attacks that we did not observe in our 2013 research on water facilities; in that research, we received 12 targeted attacks out of 39 total attacks. After we finalized our build-out, we decided on what type of products we wanted to make and how we were going to design the logic, HMI screens, and other ICS components.

Building the ICS Environment

In building our ICS environment, one of our primary goals was to prevent attackers from simply flagging our system as a honeypot, which would of course drive them away. Advanced attackers could be very picky in choosing systems they wanted to compromise and would check every small detail that they could before conducting an attack. With this is mind, we decided to use real ICS hardware and a mixture of physical hosts and hardened virtual machines (VMs).

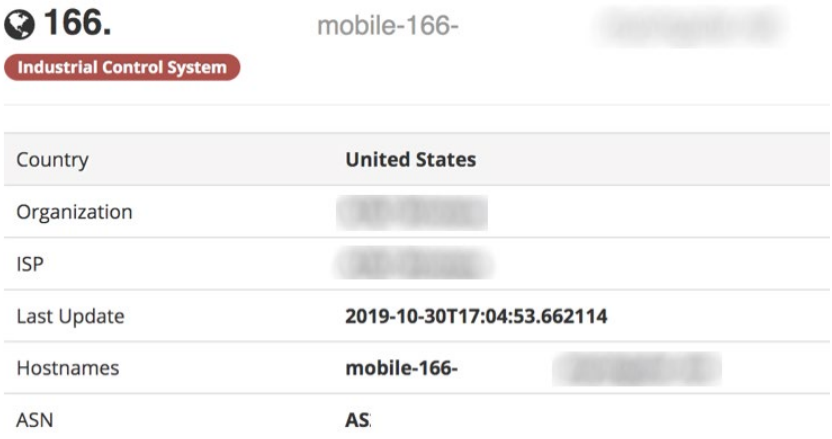


Figure 2. Shodan data classifying our honeypot as ICS

Hardware

For our ICS hardware, we ran four PLCs from three different brands: one Siemens S7-1200, two Allen-Bradley MicroLogix 1100 units, and one Omron CP1L. These PLCs were chosen for their popularity in the control systems markets from around the world. Also, each PLC brand uses a different protocol, allowing us to see if there would be any attacks on any of these PLCs that we could monitor.

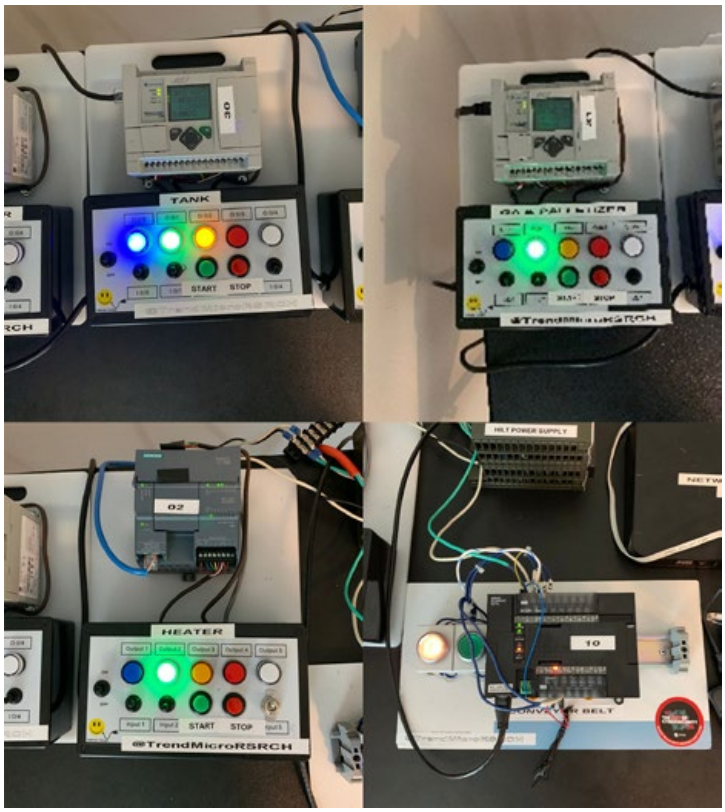


Figure 3. Hardware equipment that ran our factory

Each PLC was loaded with logic and performed specific and associated tasks that together ran the manufacturing facility. These roles were agitator, burner control, conveyor belt control, and palletizer, which used a robotic arm. To make our manufacturing process realistic, we used incremental and decremental functions through logic to vary the feedback values, which imitated the starting and stopping seen in motors or heaters. Random generator functions were also created to make slight fluctuations in the feedback values.

Machines

We had three VMs and one physical machine running our factory. The three VMs included an HMI to control our factory, a robotics workstation to control our palletizer, and an engineering workstation to program our PLCs. The physical machine was used as a file server for our factory.

Human-Machine Interface

Aside from the different brands of PLCs that we used in our production environment, we wanted to monitor the status of the logic we loaded on our devices. To mimic a realistic manufacturing environment, we created an HMI to quickly identify whether the states of our “virtual” actuators, motors, and feedback values were being modified.

During our planning stage, we found out that starting with the layout of our HMI would be much easier than starting with the PLC logic. The HMI machine of our honeypot was exposed through Virtual Network Computing (VNC) without control access.

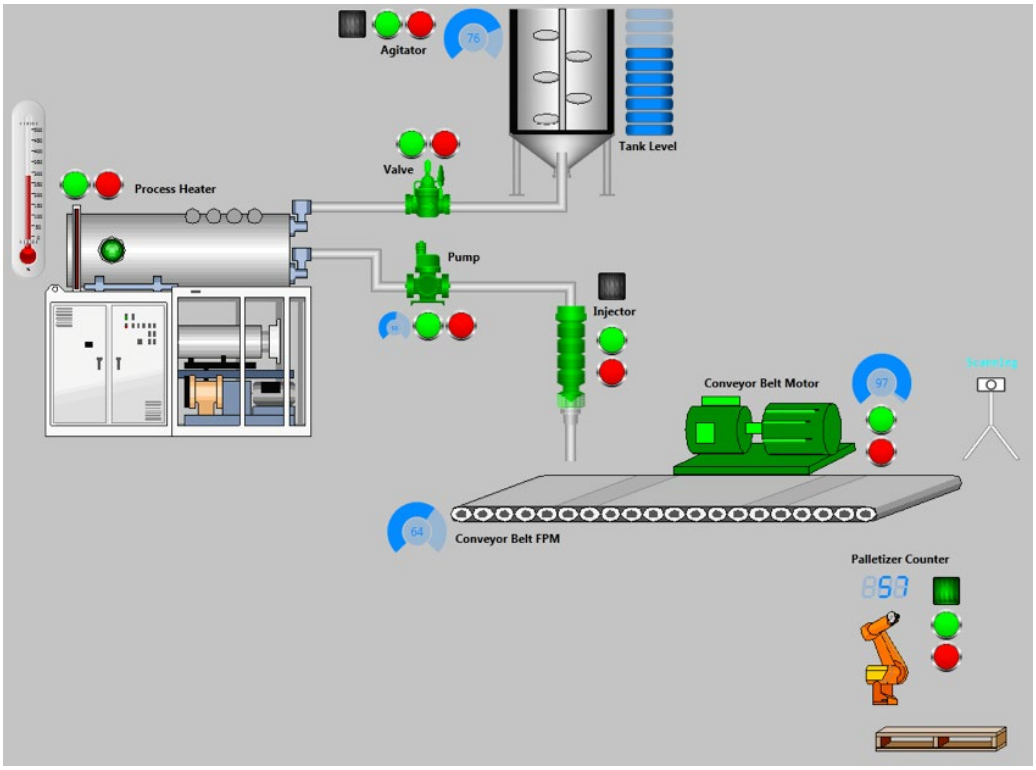


Figure 4. A screenshot of the HMI used to control the plant in our factory

Robotics Workstation

Industrial robots are a key component of modern smart manufacturing. They are used for automating not only simple pick-and-place tasks but also complex ones. Because of this, we decided to include a robotics station and its corresponding engineering workstation in our factory. To further make our factory realistic, we included a robotics workstation that would be used by engineers to graphically write the automation logic. Since actual industrial robot machines are normally isolated in an internal network, we decided to expose only the workstation via VNC.

Figure 5. Industrial robots typically found in modern factories

Given our experience with the ABB Robotics ecosystem,[7] we opted for the programming environment of RobotStudio. We downloaded this from the ABB Robotics website, installed it on a dedicated VM, and configured it accordingly. We then opened a simulation file and saw to it that the rendered 3D digital twin of the robot was visible on screen to ensure that VNC scans — such as those used by Shodan and similar search engines — would grab it and display it to whoever might be interested in exploring VNC targets. In addition, we collected some code and left it on the machine.

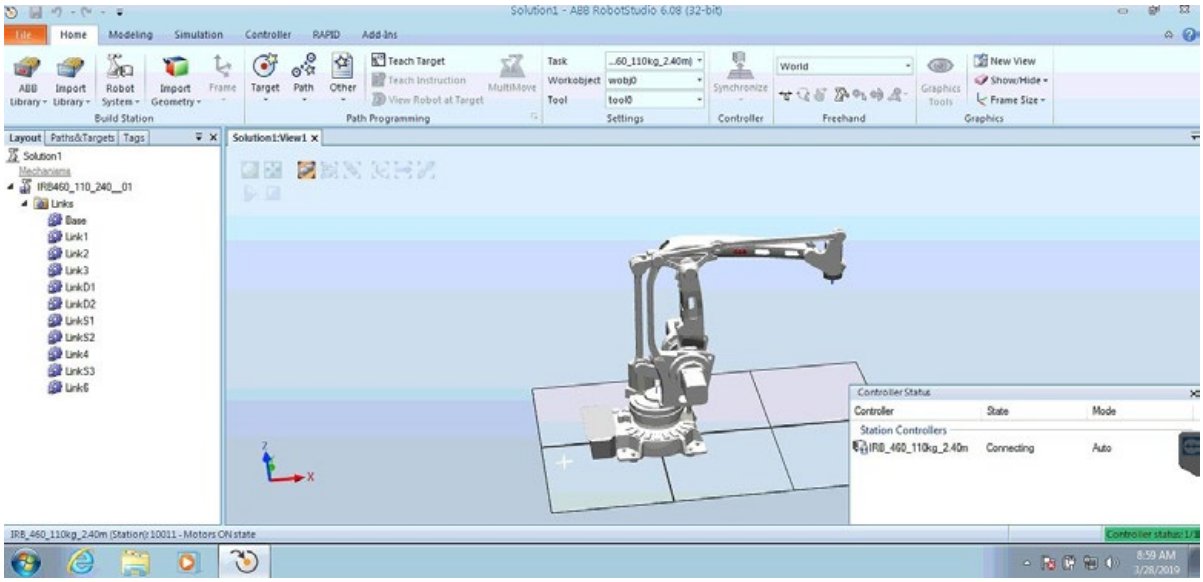


Figure 6. A screenshot of the robotics workstation, used to write automation routines for the robotics station

The effectiveness of our design and implementation in creating an attractive and realistic target was confirmed by the several attempts to attack the robotics workstation that we later observed, some of which were successful, with the attackers coming back several times.

Engineering Workstation

To program logic on the PLCs that would perform tasks similar to those required in a production line, we added an engineering workstation to our honeypot network. Like the robotics workstation, this workstation would be used by engineers to create logic for the PLCs. Accordingly, we installed industrial software for programming the PLCs we used: Totally Integrated Automation Portal (TIA Portal) for Siemens, MicroLogix for Allen-Bradley, and CX-One for Omron.

We decided that the engineering workstation would not be exposed outside of our network. Rather, we used the same admin password as that of the exposed HMI and robotics workstation. This mimicked a common setup in companies maintained by an administrator.

Unfortunately (for our purpose, that is), the engineering workstation did not receive any attacks, even though we purposely used the same admin password as that of the other exposed machines.

File Server

We set up a file server to lure attackers and also to serve as a backup for some of our own “work” in our simulated company. This provided us with the ability to sneak net files in and out of the network using a multi-USB method to ensure that we were not leaving any traces of other systems on our honeypot network. The file server was a Windows 7 Professional build that had a shared directory and allowed anyone read and write access.

At first, we built it with no files or structure and intended to build up the structure over time. But once we found that actors were actively looking at the structure, we decided to populate it with false files. To do this, we created a script that would create a file from a list of extensions. Using multiple words from a dictionary, the script would create a file of a random size.

```
38 lines (34 sloc)  1.32 KB                                   Raw  Blame  History

1  import random
2  import string
3  import sys
4  import os
5
6  # Randomly choose words from 10000 words in words .txt
7  def random_line(fname):
8      lines = open(fname).read().splitlines()
9      return random.choice(lines)
10
11 if len(sys.argv) != 2:
12         print "*\n*\n*USAGE python random_file.py NUMBER_OF_FILES\n*\n*"
13         sys.exit(0)
14 # number of files to make
15 numfile = int(sys.argv[1])
16 for x in range(0,numfile):
17         number = random.randint(100,2000)
18         # extension from extensions.txt
19         extension = random_line('extensions.txt').lower()
20         #word list used to create random files
21         file_name = "{}{}.{}".format(random_line('words.txt').capitalize(),random_line('words.txt').capitalize(),extension)
22         #use the os command mkfile to create the file
23         os.system('mkfile -n %sk %s' % (number, file_name))
24         # Randomize the Month
25         month = str(random.randint(1,7)).zfill(2)
26         #Randomize the day
27         day = str(random.randint(1,30)).zfill(2)
28         #Randomize the time
29         time = str(random.randint(0,2359)).zfill(4)
30         #Randomize the Seconds
31         seconds = str(random.randint(0,60)).zfill(2)
32         # Put it all together (19 so everything was created this year, can randomize this between years if needed)
33         date = "19%s%s%s.%s" % (month, day, time, seconds)
34         #touch the file to update the timestamp with created, modified, dates to be the same.
35         os.system('touch -a -m -t %s %s' % (date, file_name))
36
37 sys.exit(0)
```

Figure 7. The script we created to generate random file patterns

After we ran the script for a given number of files, we populated our file server with the generated "honeyfiles."

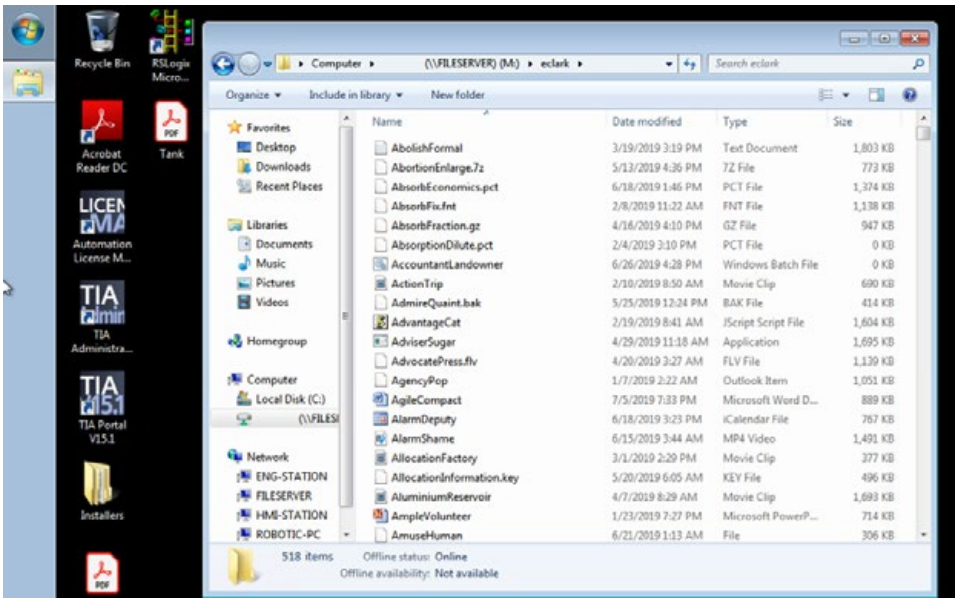


Figure 8. Honeyfiles placed in the file server to make it look realistic

Building the Company

The proliferation of honeypots used to collect attack traces has significantly raised the bar for future honeypots. Not only are current attackers already accustomed to encountering honeypots virtually everywhere, but advanced actors also typically perform in-depth investigation — using open-source intelligence (OSINT), for example — before attacking a target system to make sure that they are not about to be "caught" by a honeypot system. For this reason, our honeypot did not only need to look realistic from a design and technical implementation standpoint, but it also had to reflect a system that a real company would use.

Trying to put ourselves in the shoes of an advanced attacker, we took advantage of our OSINT knowledge to better anticipate how an attacker would act, which information they would cross-check, and from which sources. Would they check used IP addresses in reputation systems? Would they try to match the reverse lookup history of an IP address? Would they search for names and keywords, trying to find online evidence of a real company linked to that system? With such questions in mind, we started to create a list of ideas.

In this section, we discuss the details we came up with to make our honeypot more convincing as a real company.

An Attractive Target

Although it would have resulted in a very attractive target for attackers, posing as an existing company could have posed legal issues. We ruled out this option immediately because the potential reputational damage in case of an attack would be out of proportion. If existing companies would like to run honeypots, they would need to do it on their premises, under their legal responsibility. However, we still wanted to be an attractive target. Fortunately for us, many large companies rely on external workforce, especially for bleeding-edge applications and technologies (e.g., smart manufacturing). This presented a perfect position for us to take in the market.

We decided to pose as a small industrial prototyping boutique working for special customers. Considering that some small companies offer consultancy services to larger ones, we thought we could invent one such company. In this way, we would not need to claim ownership of any existing brand. After some brainstorming, we decided to pose as a small industrial prototyping boutique: a consultancy firm that specialized in advanced prototyping, serving very large anonymous customers in the military, avionic, and manufacturing sectors.

Vision and Brand Image

At this point, we started thinking like an entrepreneur. We needed to pick our business mission and name, and build an image around those ideas. We decided that we would be a company with only a small number of employees, all founding members. We wanted to convey the message that we were highly specialized, focused on our business, and for this reason worked for large and important enterprises. This was to lure the attackers into thinking that we were dealing with sensitive projects, despite being a small company. In other words, we wanted our company to appear "weak" in terms of cybersecurity, as this was not its focus, and at the same time show that it handled important assets in our target systems.

After weeks of discussing the company among ourselves, we started to feel it was real. Embracing the idea, vision, and brand of our faux company was very important because we had to depict a realistic company image while reducing room for mistakes, such as leaking details about our real affiliation. To avoid leaving any trace of our actual purpose, from the day we decided to start developing assets (e.g., images, brand material, domain registration, website), we created an isolated VM on a specific laptop, which we used exclusively to perform any activity related to our faux brand.

Online Presence

Even small companies are nowadays expected to maintain a bare minimum of presence online. Creating the illusion of a real company through social media was our first thought, but we discarded it quickly as it could be a very tedious and error-prone process. While creating social network profiles was actually easy, keeping them active was the hard part. And inactive, freshly created social profiles would just be red flags. We decided that we did not want to invest time and resources in maintaining social media profiles. Anyone who might be planning on recreating our setup should consider making the same decision, as they could fall into thinking that it would only take a few posts once in a while to maintain a believable social media account.

With this in mind, our first asset was a website. In creating a website, we first needed a company name. We chose a name that reflected our rapid prototyping specialization: MeTech. MeTech was a fictitious company name that we came up with to give the honeypot additional credibility. It is not now, nor has it ever been, affiliated in any way with any real company that uses the word "MeTech" or any similar words or phrases as part of its company name.

We then decided on a motto and a logo, for which we simply used a free logo generator. Similarly, we used a royalty-free web template to design a professional-looking website with only a few pages. The front page was the most important one. We needed to quickly convey a clear message in only that one page, which was that we were contractors for large enterprises. Our advertised business (i.e., rapid prototyping) needed to be in line with our honeypot system. The finished homepage of our website, metech.co, is shown in Figure 9.

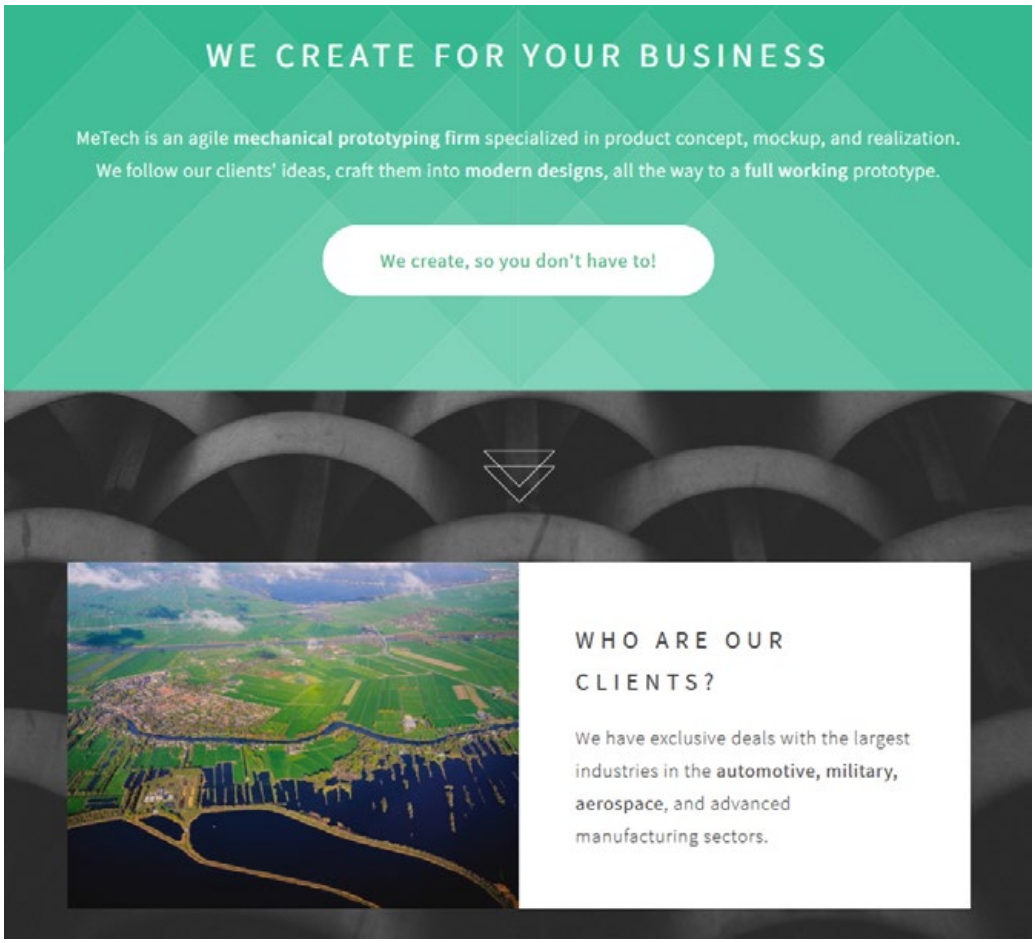


Figure 9. A screenshot of our company's homepage

We hand-picked royalty-free images from various sites. We focused on large, low-popularity pictures, from which we could crop relevant objects (e.g., machines, prototypes, labs). This also reduced the chances of anyone finding the same photos should they search for keywords related to our business. Similarly, cropping and rotating avoided the possibility of someone finding the real source of an image using, for instance, Google reverse image search.

People and Contacts

We wanted to look like a real company composed of real people, with professional profiles in line with the expertise needed by our job. We came up with the full names of four fictitious people and looked for stock images that would fit their ethnicities. Initially, we focused on group photos or photos from which we could crop and slightly alter good head shots.

We did not want a casual attacker to be able to use the reverse image search features of some web services (e.g., Google Images) to discover that we had used photos taken randomly off the web. Eventually, we looked for good-quality AI-generated photos of nonexistent people. We then came up with a short profile for each name and photo. We kept the profiles short and essential, without providing too many details, which could be used by an attacker to cross-check the profiles. For this reason, we avoided putting any affiliation (e.g., school or city names) and kept the profiles rather generic and focused on technical skills. Since we were just a small company working for important customers, we thought that discretion would fit our overall image. The result is shown in Figure 10.

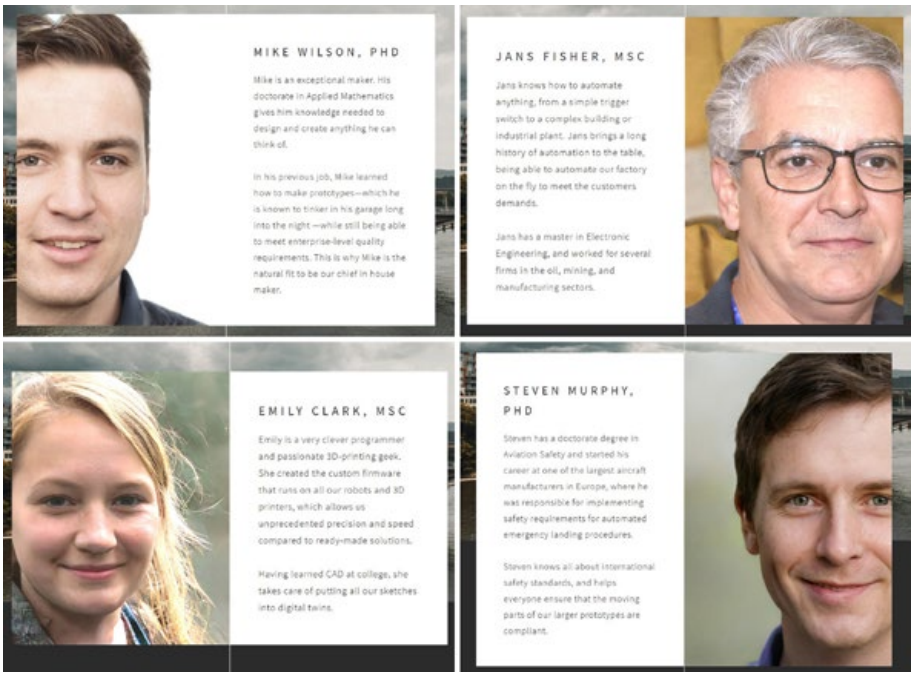


Figure 10. Screenshots of the contacts page of our company website

Creating a contact email address was very easy. We used our hosting provider's mail server. On top of that, we used a simple online service to build a web form that we could put on our website for it to look even more professional. The form that we created had a simple handler that forwarded any contact request to our email inbox.

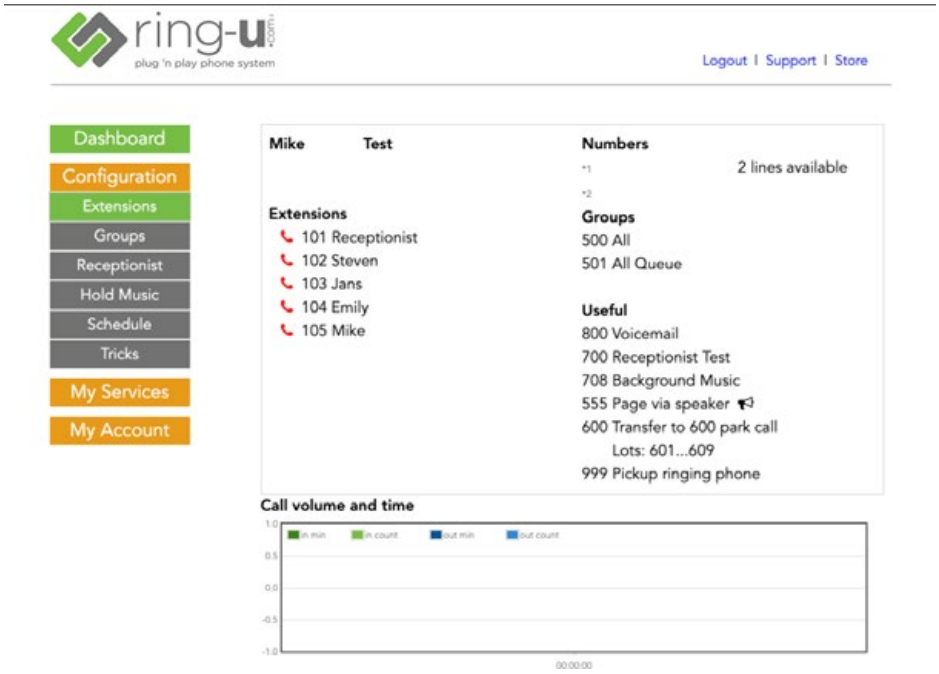


Figure 11. Virtual receptionist configuration

Creating phone numbers required an external VoIP provider. Considering the many options available nowadays (a simple web search for “e-receptionist” or “cloud phone service” would yield a number of results), we opted for a service run by a personal contact of our own. We wanted someone whom we could trust and who would allow us a little control over the system, e.g., allow us to customize the service if needed.

We created two phone numbers under the U.S. country code, one per site (the engineering lab and the front office). We wanted someone who could answer professionally, so we configured the e-receptionist system with a distinct recorded message, with a voice guide and the option to leave messages.

Building the Honeypot

With a clear concept of what we wanted for our honeypot, we were able move on to the execution of our plans. In this section, we discuss the infrastructure and tools we used to create our honeypot. These include actual operational technology (OT) software and monitoring tools that would allow us to record incidents in our honeypot in real time. We also illustrate the actions we took to lure potential threat actors into attacking our system.

Monitoring Environment

To get the most out of our honeypot, we had to carefully design our monitoring environment. We needed to get relevant data in real time while keeping the evidence of our monitoring to a minimum so as not to deter threat actors from naturally conducting an attack.

Infrastructure

The monitoring infrastructure consisted primarily of a Raspberry Pi 3, four USB Ethernet adapters, four SharkTap Ethernet taps, and a large external drive. We inserted the Ethernet taps into four specific locations in the network, as illustrated in Figure 12.

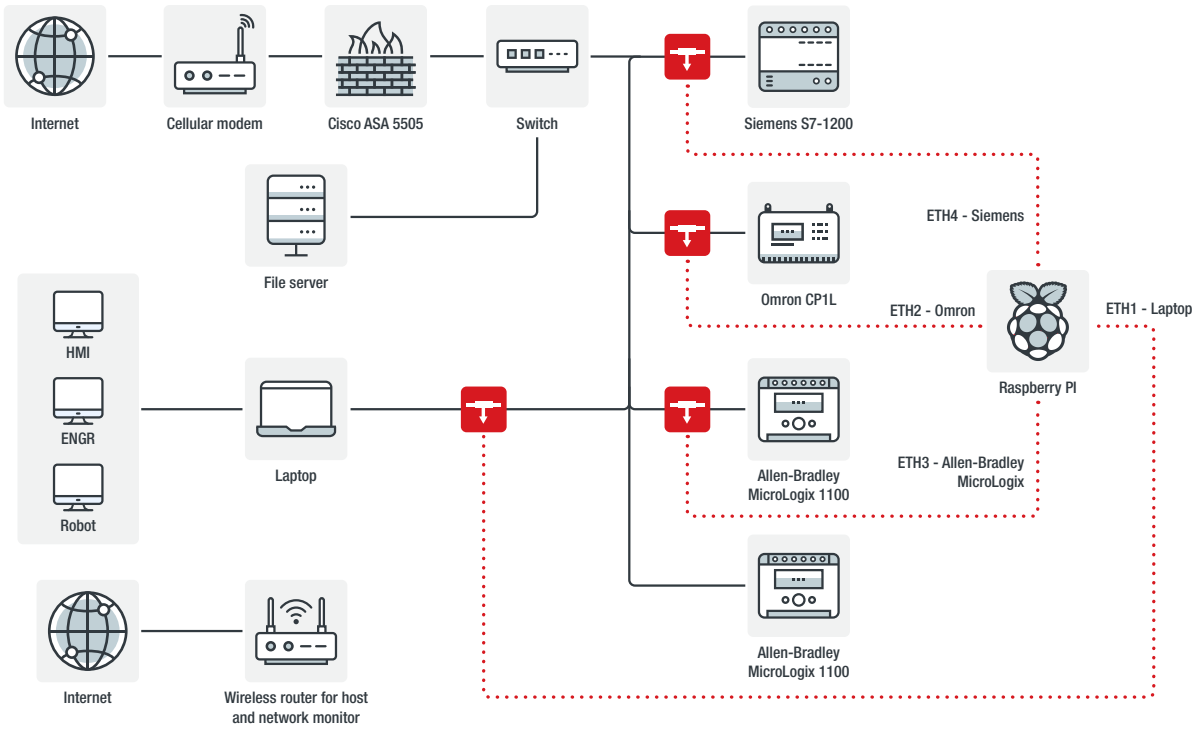


Figure 12. The full honeypot design with the red boxes depicting the SharkTap Ethernet taps

The location of three of the PLC Ethernet taps allowed us to monitor external traffic sent specifically to the three internet-exposed PLCs, with one PLC accessible only from the local network. The fourth Ethernet tap was used to monitor the exposed VirtualBox guests and any traffic from the guests to any of the other systems on the network.

Figure 13. A SharkTap device and the Sierra Wireless AirLink RV50 that we used

The Raspberry Pi 3 was able to handle our packet load with few dropped packets, although the new Raspberry Pi 4 would have worked better as it supports USB 3.0. The Raspberry Pi could be accessed only via the VirtualBox host. We set up the Raspberry Pi to capture the traffic in 24-hour increments, which made it easy to parse what was happening on a daily basis. For some lightweight analysis, the summary scripts described in the "Tools" subsection below was performed on the Raspberry Pi, while other analysis required the packet capture (PCAP) files to be off-loaded to a secondary server.

The internal infrastructure was connected to the internet via an industrial cellular router (as illustrated in Figure 12), the Sierra Wireless AirLink RV50. We chose this router since its predecessor, the AirLink Raven, is commonly used in the U.S. An industrial cellular router is similar to a consumer cellular router built for industrial environments in that it also has features such as indoor and outdoor use and greater operating temperatures.

Firewall

During our research, we found out a limitation of our industrial router, which was that it would not let us do selective blocking to prevent attacks that ran counter to what we were trying to encounter with our honeypot. To remedy this, we deployed a firewall in transparent mode between the router and the switch. Our firewall was a Cisco ASA 5505, which we had on hand, but this could have just as well been any firewall that could operate in transparent mode. Adding the firewall allowed us to block things with little impact to the network.

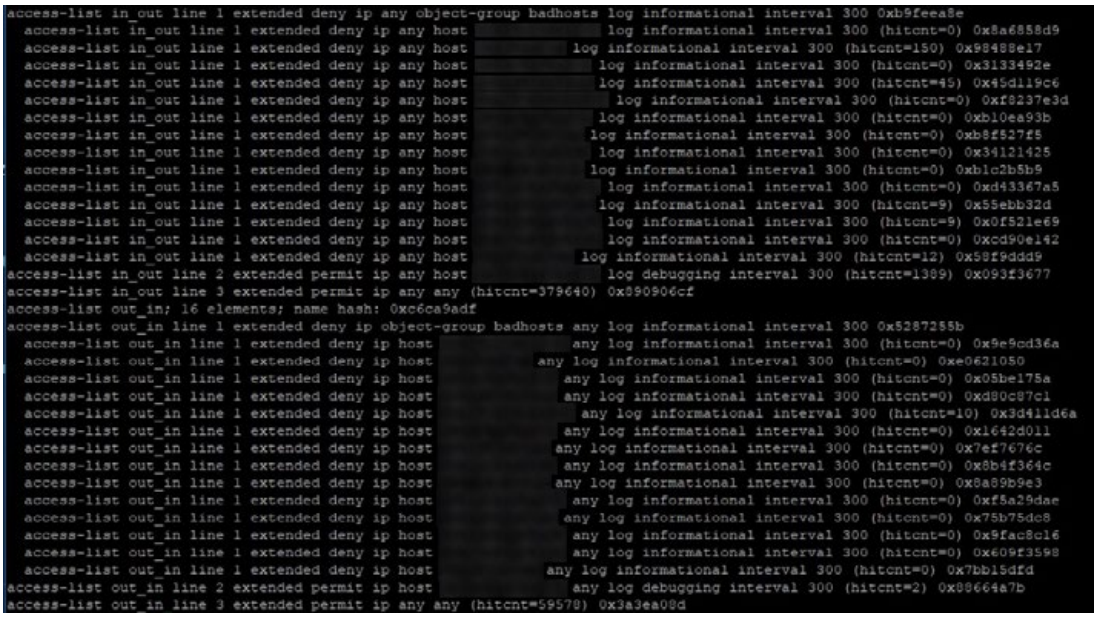


Figure 14. Access lists that we ran on our network appliance

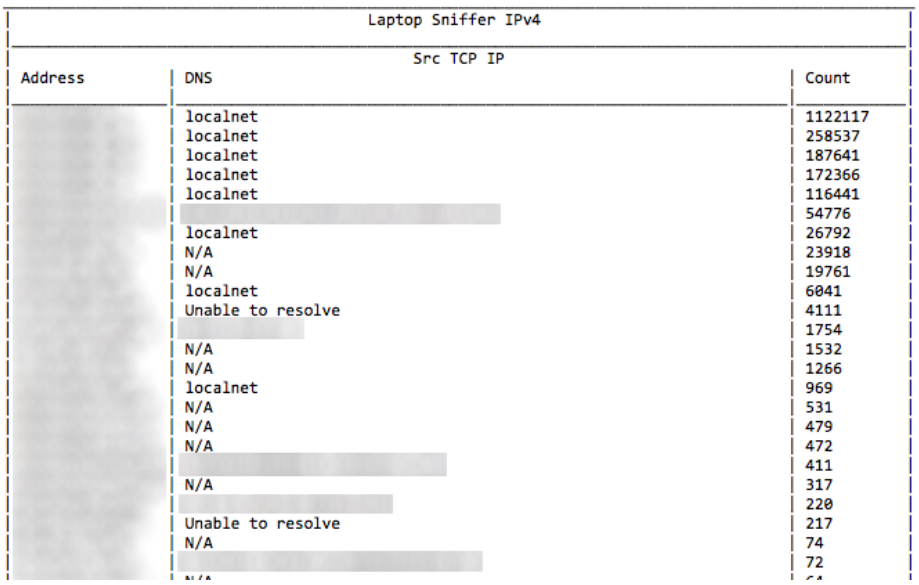
Figure 14 shows an example of the access lists that we ran on our network appliance. We had an object group called “bad hosts” where we put hosts that we did not want communicating with our honeypot and vice versa. These included known good hosts that for the purpose of our honeypot we tagged as malicious. Adding these known good hosts to our firewall also prevented fraud from originating from our systems.

Tools

Scripts

Any internet-facing system nowadays immediately receives a lot of traffic, especially from scanners. For our purpose, it was not straightforward to enumerate and whitelist all the scanner sources as they vary over time. In addition, not all scan traffic comes from legitimate scanning services such as Shodan, which advertises its scanners in such a way that people could filter out their source IP addresses. Initially, we had to perform some manual work to tell “signal” and “noise” apart. For this reason, we wanted to receive daily summary emails with traffic statistics, rankings, and other analytics that could help us spot interesting patterns.

We generated three different scripts to assist in monitoring traffic. The first script created a list of IP addresses that connected to our PLCs and VirtualBox guests, created the corresponding reverse Domain Name System (DNS) information for each IP address, and counted the number of packets from each IP address. The list is then emailed to the members of our team.



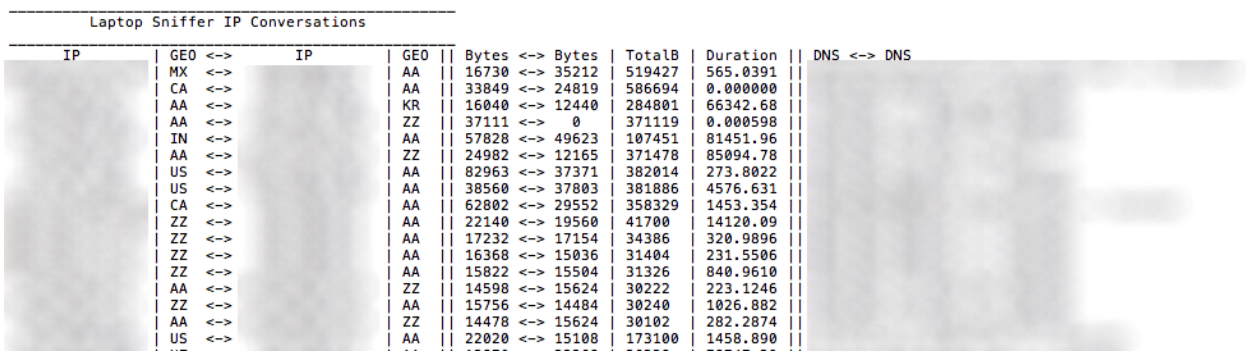
Laptop Sniffer IPv4

Src TCP IP

Address	DNS	Count
	localnet	1122117
	localnet	258537
	localnet	187641
	localnet	172366
	localnet	116441
		54776
	localnet	26792
	N/A	23918
	N/A	19761
	localnet	6041
	Unable to resolve	4111
		1754
	N/A	1532
	N/A	1266
	localnet	969
	N/A	531
	N/A	479
	N/A	472
		411
	N/A	317
		220
	Unable to resolve	217
	N/A	74
		72

Figure 15. Output of the first script, containing actual IP addresses and host names

The second script looked at external conversations. It used Wireshark's tshark CLI application to gather the statistics of the conversations, perform a reverse DNS and GeoIP lookup, and email the list to the team.



Laptop Sniffer IP Conversations

IP	GEO <->	IP	GEO	Bytes <-> Bytes	TotalB	Duration	DNS <-> DNS
	MX <->		AA	16730 <-> 35212	519427	565.0391	
	CA <->		AA	33849 <-> 24819	586694	0.000000	
	AA <->		KR	16040 <-> 12440	284801	66342.68	
	AA <->		ZZ	37111 <-> 0	371119	0.000598	
	IN <->		AA	57828 <-> 49623	107451	81451.96	
	AA <->		ZZ	24982 <-> 12165	371478	85094.78	
	US <->		AA	82963 <-> 37371	382014	273.8022	
	US <->		AA	38560 <-> 37803	381886	4576.631	
	CA <->		AA	62802 <-> 29552	358329	1453.354	
	ZZ <->		AA	22140 <-> 19560	41700	14120.09	
	ZZ <->		AA	17232 <-> 17154	34386	320.9896	
	ZZ <->		AA	16368 <-> 15036	31404	231.5506	
	ZZ <->		AA	15822 <-> 15504	31326	840.9610	
	AA <->		ZZ	14598 <-> 15624	30222	223.1246	
	ZZ <->		AA	15756 <-> 14484	30240	1026.882	
	AA <->		ZZ	14478 <-> 15624	30102	282.2874	
	US <->		AA	22020 <-> 15108	173100	1458.890	

Figure 16. Output of the second script, showing external conversations

The third script was added later to search for IP addresses that connected to the factory honeypot multiple times over the course of days, weeks, or months. This script crawled through all of the PCAP files we had collected, which were segregated by date. It aggregated IP addresses, reverse DNS information, the last date an IP address connected to the honeypot, and the number of days an IP address had connected to any of our systems.

ipStr	count	dns	last_seen
Filter	Filter	Filter	Filter
[redacted]	67	[redacted]	2019-09-23
[redacted]	67	[redacted]	2019-10-01
[redacted]	64	[redacted]	2019-09-23
[redacted]	59	[redacted]	2019-10-01
[redacted]	57	[redacted]	2019-09-23
[redacted]	53	[redacted]	2019-09-30
[redacted]	50	[redacted]	2019-07-29
[redacted]	50	[redacted]	2019-08-03
[redacted]	49	[redacted]	2019-09-29
[redacted]	49	[redacted]	2019-10-01
[redacted]	49	[redacted]	2019-10-01
[redacted]	46	[redacted]	2019-10-01
[redacted]	46	[redacted]	2019-10-01

Figure 17. Output of the third script, showing aggregated statistics on each IP address

These scripts made it easier for us to hunt for and monitor specific or potentially interesting actors using Moloch.

Network Traffic Analysis and Investigation With Moloch

We needed an effective way to manually dig into the network traffic recordings. While using command-line tools (e.g., tshark and tcpdump) along with some shell scripts is great for quick tasks, such an approach would not be enough to fully handle an environment where multiple people need to collaboratively analyze and look at many gigabytes of traffic. We turned our attention to Moloch,[8] an open-source traffic analysis system developed by AOL. It provides a user experience similar to that of Wireshark but has more features for collaborating, annotating packets, tagging, exporting, drilling, and other tasks.

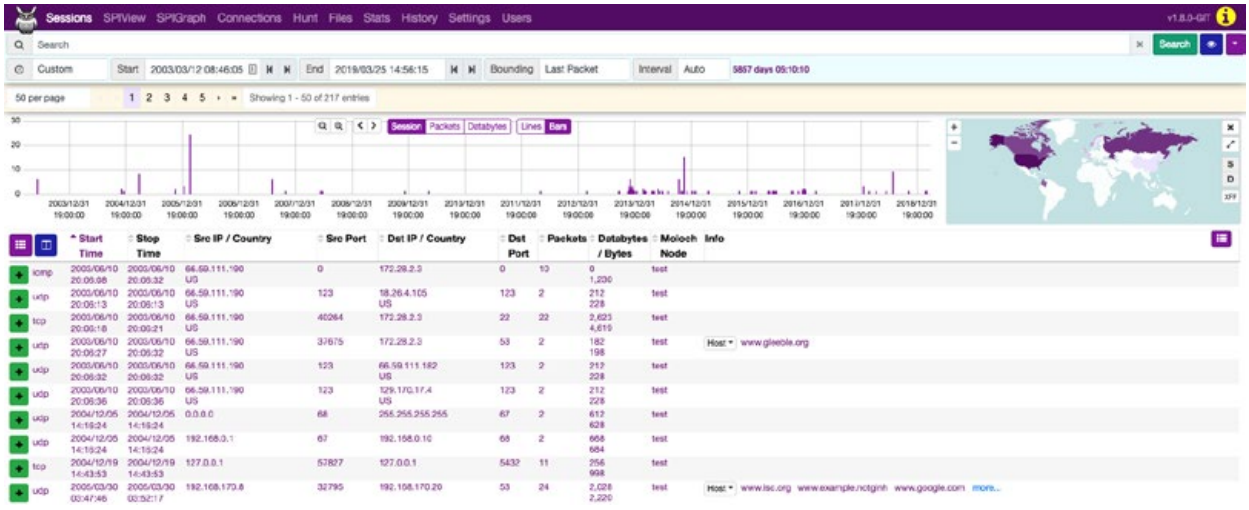


Figure 18. A screenshot of Moloch, purposely picked from Moloch's official website to avoid revealing any detailed information about the real traffic that our honeypot had received

Installation was rather easy, but we needed some fine-tuning. Since we did not want to process the network data using the same machines used for our honeypot, we exported the PCAP dumps every day to an Amazon Simple Storage Service (S3) bucket. We then set up a daily cron script that imported the files from the S3 bucket to the Moloch machine, which we located on the Amazon Web Sevices (AWS) cloud. Moloch's default setup is to process live network traffic from the system interfaces. However, it is possible to configure it to process offline data and move it away from a "queue" folder once done.

VirtualBox Screen Recording

To properly document the attacks that we would be receiving on our honeypot system, we decided to record video of the screen every time there was a change made or every time there was VNC access to the opened robotics workstation. We created a script that would monitor the VirtualBox guest's screen through our host machine and take a screenshot on a fixed interval. We then compared two images to check if there were any big differences — an indicator that someone had been trying to access our honeypot through unauthenticated VNC. We started screen-recording the monitored guest VM once a screen change was detected, continuously checking whether the VM was still being accessed before stopping the recording.

The VirtualBox screen recording proved effective. We minimized any VirtualBox installation footprint on our guest VM by not installing any guest additions. We also tried replaying VNC traffic from the PCAP to see what had been changed on our system, but using the custom screen recording was an easier route for us to go with.

VNC Utilities

We investigated both VNC and Remote Desktop Protocol (RDP), and determined that VNC was easier to monitor from a pure network perspective than RDP. To make sure the honeypot looked as real as possible, we did not have additional tools running on the VirtualBox guests that might tip off an attacker. The "remote framebuffer" (RFB) protocol VNC is built on was easier to extract information from — information such as keystrokes and clipboard data. In some cases, the VNC session could even be replayed.

The main tools we used for VNC monitoring were Chaosreader[9] and VNCLogger.[10] Chaosreader can extract VNC sessions and keystrokes from a PCAP file. While the VNC session replay worked well with internal testing, it was difficult to get working for much of the real-world VNC connections, which was why we relied on the VirtualBox screen recording. The keystroke extraction worked, but we found that it did not show certain keystrokes (e.g., backspace, enter, control), so it was not as useful as VNCLogger for keylogging.

VNCLogger did a better job of extracting all of the keystrokes, but there were a couple of drawbacks to it. One drawback was that it is designed to listen on an interface instead of processing PCAP files. This issue was easily overcome using tcpreplay. The other drawback to VNCLogger, one shared by Chaosreader, was that it does not show clipboard data. In order to obtain clipboard data, we used Wireshark. A few minor updates to VNCLogger would address the gaps we noticed.

Zeek and Intel Stack

We looked into the network security monitor Zeek,[11] formerly known as Bro, combined with Intel Stack, a marketplace of free threat intelligence optimized for it.[12] While some of the output was interesting, we determined that Trend Micro's own tools and the Suricata threat detection engine[13] provided data that was more relevant to our purposes.

Syslog Feed From Router, Firewall, and Other Appliances

As mentioned earlier, we implemented a transparent firewall, allowing us to log our access lists and syslog them to a host. This host did not need to have any syslog systems running as we configured our syslog to use User Datagram Protocol (UDP). We then pointed syslog to the file server IP address so that we could pick up the syslog messages on our previously described full packet captures.

```
logging enable
logging buffered notifications
logging trap informational
logging host inside FILESERVER
```

Figure 19. Enabled syslog logging on the Cisco ASA 5505

We also logged our AirLink router for any events that were possibly caused by an attacker to the router itself. Figure 20 shows examples of messages we saw from the router and the transparent firewall.

Source

```
3c31 333e 3120 3230 3139 2d31 302d 3135 <13>1.2019-10-15
5430 353a 3538 3a34 322b 3030 3a30 3020 T05:58:42+00:00.
5256 3530 5820 414c 454f 535f 5345 5256 RV50X.ALEOS_SERV
4943 4553 5f4d 6375 4d61 6e61 6765 7220 ICES_McuManager.
2d20 2d20 5b6d 6574 6120 7365 7175 656e -.-.[meta.sequen
6365 4964 3d22 3231 3739 225d 2050 726f ceId="2179"].Pro
6365 7373 506f 6c6c 696e 670a           cessPolling.
```

Source

```
3c31 3636 3e25 4153 412d 362d 3330 3230 <166>%ASA-6-3020
3136 3a20 5465 6172 646f 776e 2055 4450 16:.Teardown.UDP
2063 6f6e 6e65 6374 696f 6e20 3432 3230 .connection.4220
3336 2066 6f72 206f 7574 7369 6465 3a31 36.for.outside:1
3932 2e31 3638 2e30 2e32 3534 2f35 3137 92.168.0.254/517
3931 2074 6f20 696e 7369 6465 3a31 3932 91.to.inside:192
2e31 3638 2e30 2e39 392f 3531 3420 6475 .168.0.99/514.du
7261 7469 6f6e 2030 3a30 323a 3031 2062 ration.0:02:01.b
7974 6573 2031 3038 0a3c 3136 363e 2541 ytes.108.<166>%A
5341 2d36 2d33 3032 3031 353a 2042 7569 SA-6-302015:.Bui
6c74 2069 6e62 6f75 6e64 2055 4450 2063 lt.inbound.UDP.c
6f6e 6e65 6374 696f 6e20 3432 3230 3435 onnection.422045
2066 6f72 206f 7574 7369 6465 3a31 3932 .for.outside:192
2e31 3638 2e30 2e32 3534 2f35 3137 3931 .168.0.254/51791
2028 3139 322e 3136 382e 302e 3235 342f .(192.168.0.254/
3531 3739 3129 2074 6f20 696e 7369 6465 51791).to.inside
3a31 3932 2e31 3638 2e30 2e39 392f 3531 :192.168.0.99/51
3420 2831 3932 2e31 3638 2e30 2e39 392f 4.(192.168.0.99/
3531 3429 0a                            514).
```

Figure 20. Examples of messages we saw from the router and the transparent firewall

Luring Attackers

One of the main goals of a honeypot is to be attacked. But how does one go about setting up a honeypot to be attacked? The approach we took was to stage leaking information, to put us in such a position that attackers might find our honeypot interesting enough to play around in and eventually attack. To do this, we started by opening specific ports when we brought the honeypot online, as listed in Table 1.

Port Number	Service
102	Siemens S7
3389	RDP
5900	VNC
5901	VNC
9600	Omron FINS
44818	EtherNet/IP

Table 1. Exposed services on the external router

These were the ports that we started with in opening our honeypot to the internet through our AirLink Wireless gateway. However, because of the increased amount of scanning on RDP TCP/3389, we saw huge performance issues on our network and we decided to not run RDP as it at times made the internet unusable from our stance. With RDP open on a certain network, there may be performance issues on the network due to the amount of traffic that is coming at the network to brute-force RDP, including ones that use exploits to get into the network with RDP. After we removed RDP, the final setup looked as shown in Figure 21.

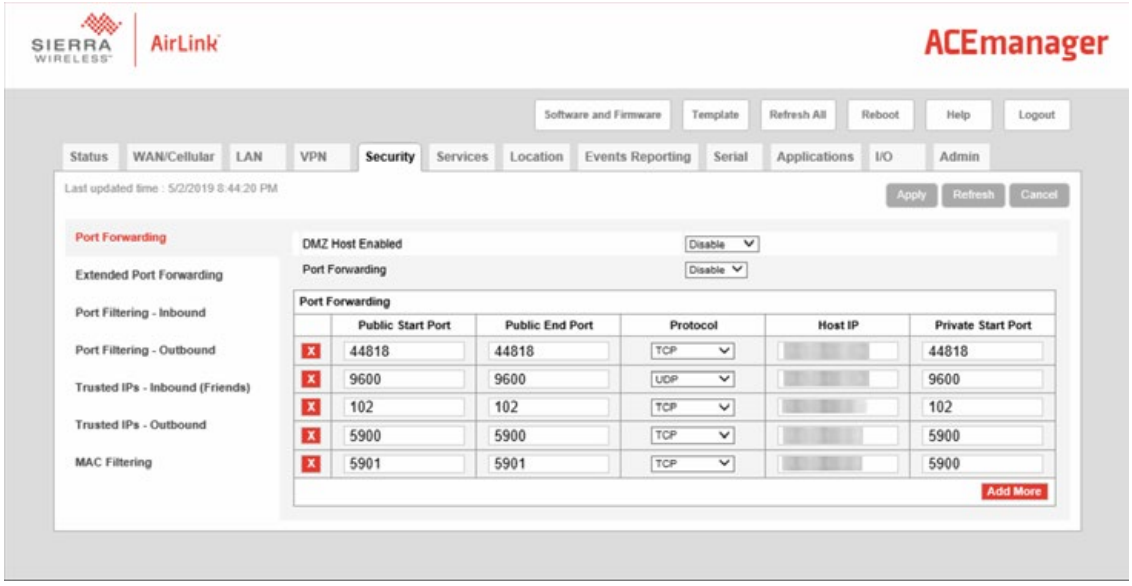


Figure 21. The final setup on the external cellular router

This was how we ran the system for months, with both of the VNC services in view-only mode, which required no password. Here was where we tried to make things look realistic and see how fast it would take for an attacker to notice if something got changed on the network and take advantage of it.

About a month into running our honeypot, we "misconfigured" VNC to allow remote input on the robotics workstation. We did this over the HMI just to see whether anyone would try to pivot from machine to machine.

Later, we acted like a victim infected by malware and uploaded several items to an online antivirus aggregation service, including network diagrams of our factory and some other sensitive information. This was to see whether an attacker might be using credentials to search for information leakage. However, we saw no attacks related to this information leakage in the online antivirus aggregation service.

Shortly afterward, we posted some information about our honeypot on Pastebin, as shown in Figure 22. This included a link to our submission to the antivirus aggregation service and basic information, to again see whether we could lure attackers with information leakage in one of the typical places where it happens.

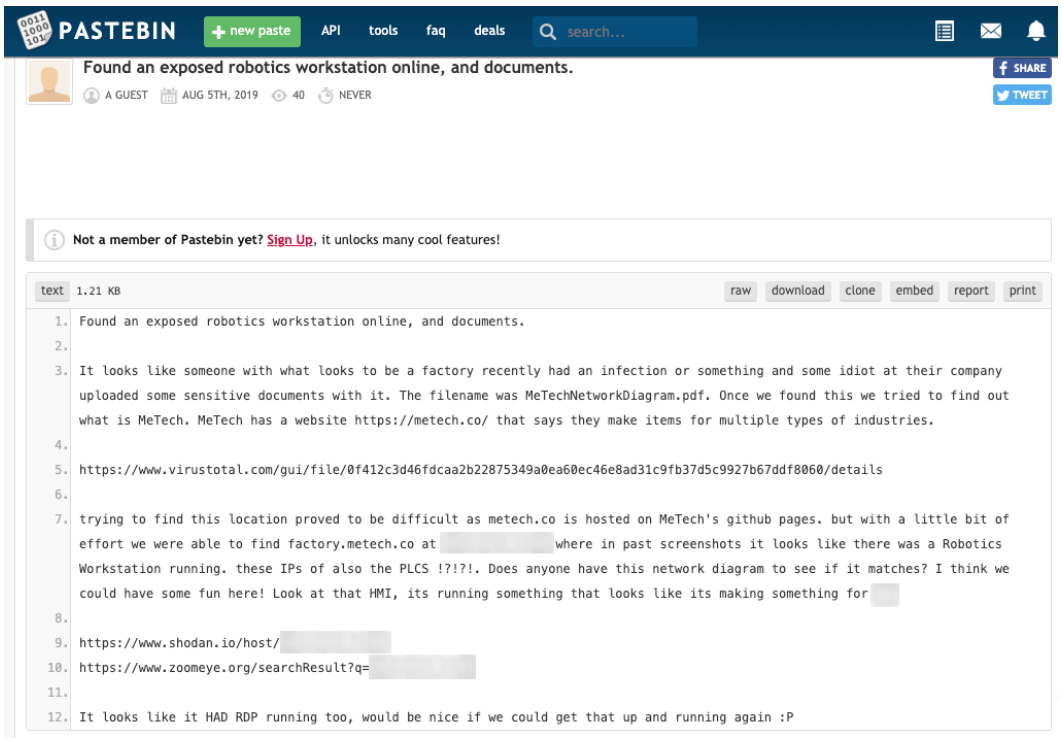


Figure 22. Information about our honeypot, which we posted on Pastebin to attract attackers

We then wanted to see whether there would be an increase in attacks if we exposed the web HMI from the vendor of the HMI that we were using. We made a second Pastebin post, shown in Figure 23, about this a few days later. The post included an update that made it sound as though the post came from a group that was actively monitoring the system.

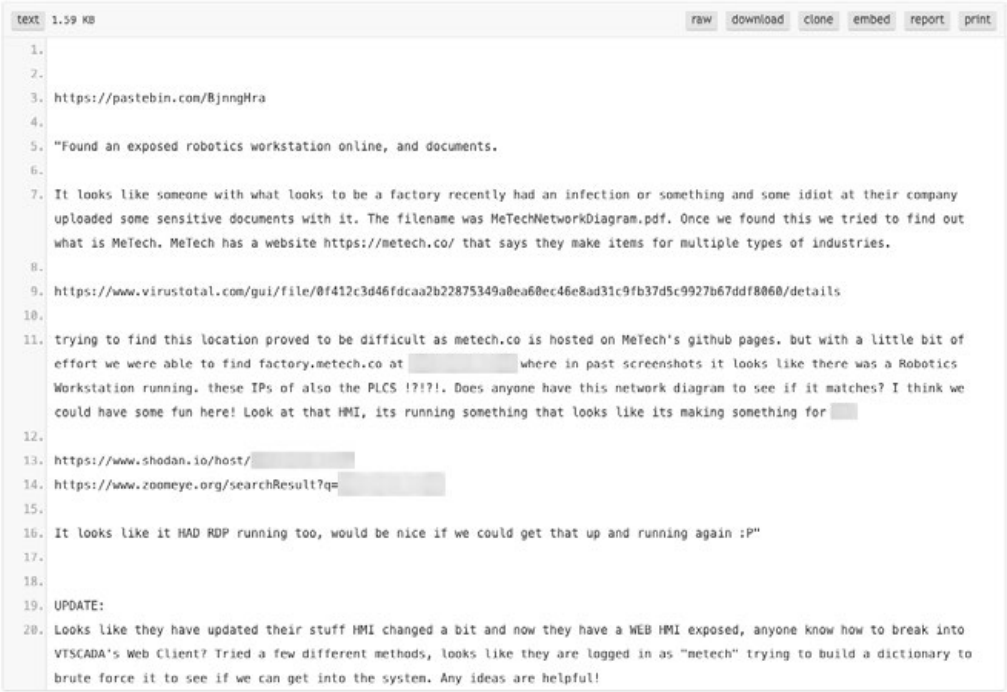


Figure 23. Our second Pastebin post, including an update that made it sound as though the post came from a group that was actively monitoring our honeypot

We then monitored underground forums and other online locations for any communications about our honeypot to see whether anyone else was discussing it. However, we did not find anything related specifically to our honeypot.

Lessons Learned

During the time that we ran our honeypot, we learned several things that needed to be checked to make our honeypot more appealing, realistic, and resilient to attacks. This subsection discusses a short list of details that need to be accomplished before running a complicated factory honeypot:

- **Remove all VirtualBox artifacts.** On the first few weeks we went live, a member of our team checked whether there was a VirtualBox tray icon on the Windows task manager.
- **Back up files.** We did this for resiliency to ransomware attacks, which our honeypot would encounter often.
- **Frequently take snapshots of VirtualBox images.** We took a snapshot when the system was clean, after an attack, and/or after a Windows update. Ideally, snapshots should be taken daily if storage space is not an issue.

- **Design the HMI before creating the logic.** We searched for the normal operating values of the actuators and motors needed on the manufacturing line. From there, we decided on the layout of the HMI and created the appropriate logic. We then refined the logic and HMI as needed.
- **Install a firewall.** Filtering fraud-related attacks and other unwanted attacks can take some time. Unfortunately, it is difficult to find the right balance between "attacks that you want to filter" and "attacks that may turn out to be useful and related to the honeypot." For this reason, we had to adjust the firewall rules over time, as we learned of new attacks.

Incidents

Our honeypot went online in May 2019. For seven months, we maintained the image of a real company and monitored the honeypot closely. The first attack we encountered came a month after the honeypot went live, with several others following in its wake. This showed that our ruse as a small business with critical clients and inadequate security was effective in luring threat actors.

Some of the attacks we saw had been briefly mentioned in the previous section detailing the conceptualization and creation of our honeypot. In this section, we discuss each type of incident that we saw. We also provide a summary of our findings in a single timeline illustrating the order in which the attacks occurred and which of them overlapped.

Scanners

As mentioned earlier, when we started looking at incidents we specifically wanted to exclude any traffic that was generated by scanners of well-known, reputable companies. These included ip-ip, Rapid 7, Shadow Server, Shodan, and ZoomEye,. However, during our review of all the traffic, we found that there were many other scanners from companies that performed internet figure monitoring and related services, which we also needed to exclude. To do so, we did reverse lookups of IP addresses that were observed hitting our honeypot in a scanner-like manner. We excluded IP addresses that resolved to sites that, when visited, explained the nature of their scanning.

Of the 9,452 unique IP addresses observed over the period our honeypot was online, 610 were linked to scanners. That was 6.45% of all unique IP addresses.

Frauds

One of the biggest risks we ran into, as with anyone else who has VNC or RDP open to the internet, was misuse of the system and resources by third parties to engage in fraudulent activities. We observed that third-party actors had used the resources in the honeypot to engage in and obscure possibly abusive and inappropriate activities, such as buying smartphones by upgrading mobile subscriber accounts and cashing out airline miles for gift cards.

Malicious Cryptocurrency Mining

One of the first uses our honeypot took on shortly after we opened remote control on VNC was as a cryptocurrency miner. A threat actor came into our system, opened a web browser, went to a website, and downloaded a PyInstaller[14] bundle file called host.exe.

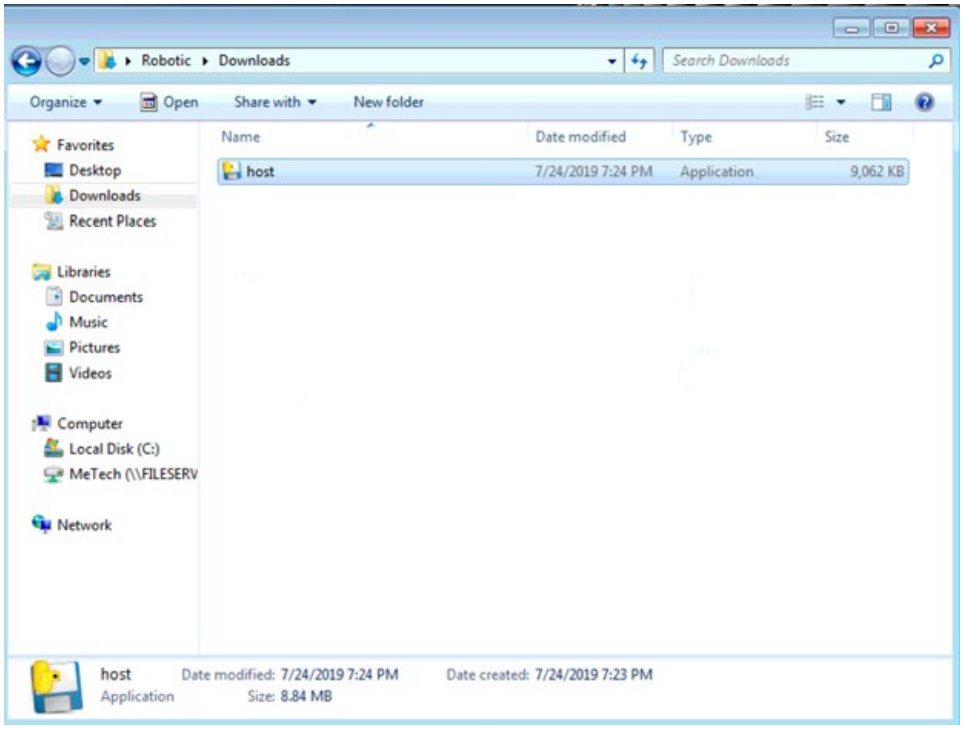


Figure 24. A PyInstaller file called host.exe, which was downloaded by one of the threat actors who accessed our honeypot

Using a combination of PyInstxtractor[15] and Uncompyle6[16] to decompile the file, we learned that this file was an open-source tool called Ares[17] that had a hard-coded command-and-control (C&C) server. Based on these findings, we then looked into what information was being sent to the C&C server. This was where we discovered that the threat actor had installed and joined our system to a well-known Monero-mining system. We found cryptocurrency mining on a VM strange as it seemed that it would not yield much. However, if this was going on with many other machines around the world, the threat actor behind it could cash in on the attack well.[18]

Ransomware Attacks

During the period we were running our honeypot, we came across two ransomware infections on our system. As mentioned earlier, we purposely exposed an accessible VNC service on our robotics workstation and recorded videos of the attackers carrying out their campaigns. These two separate incidents were most likely carried out by two unrelated individuals or groups, but the execution flows of the ransomware attacks were quite similar.

Crysis Ransomware

The first ransomware attack we encountered happened in late September. We were able to document the entire duration of the attack. We also responded to the attackers, still posing as our organization, to gain further insight into how similarly threat actors might conduct their deals.

On Sept. 22, a threat actor began looking around our system. Typical of threat actors, they first investigated the system, likely looking for important and sensitive files. They looked at a few items such as the shared drive. Their next actions were to close the robotics workstation application and to go back to the shared drive to see how much information was on it.

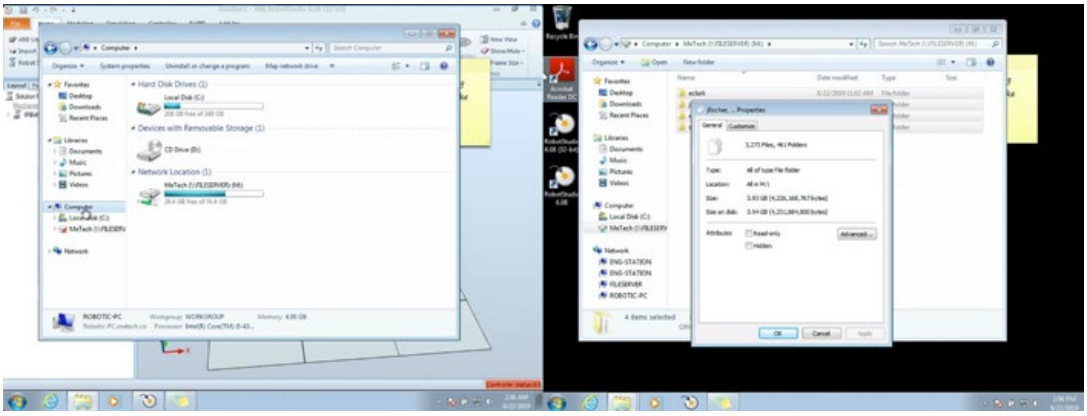


Figure 25. Investigation of the system carried out by the threat actor

After these initial actions on our system, they then downloaded the remote desktop software TeamViewer. In fact, they opened Bing and searched for “timeviwer” to do so. Then they ran the TeamViewer installer. They chose to run TeamViewer only once and chose the option to use it for personal use.

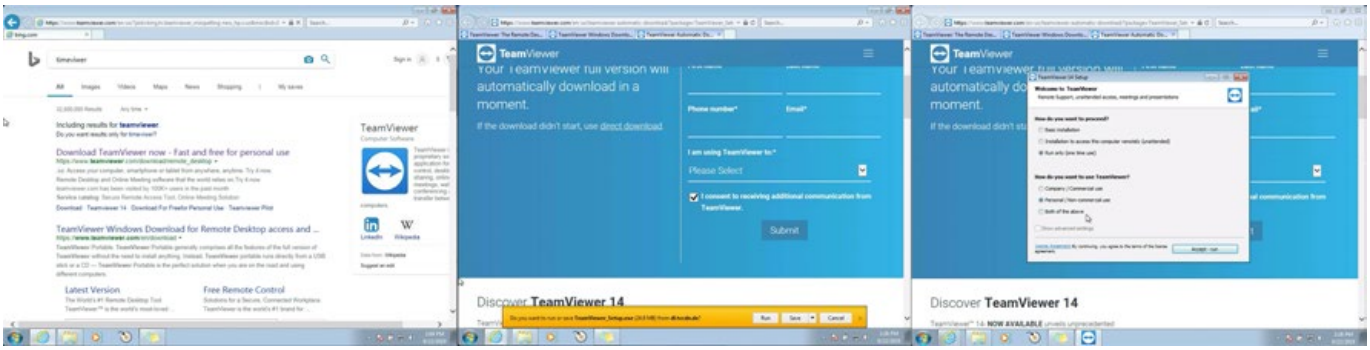


Figure 26. Downloading and running of TeamViewer performed by the threat actor

Once they started connecting to our system using TeamViewer, we lost further keystrokes from the PCAPs. This did not stop us from monitoring this attack, however, especially since at this point the threat actor had started to take more interesting actions. They then transferred three files over TeamViewer, which included the ransomware file:

- 1btc.exe
  - The ransomware file, a variant of Crysis
  - Detected by Trend Micro as Ransom.Win32.CRYSIS.SM[19]
  - SHA1: ddf8c065d45c734b5b58e770e4f1ea086a293f19
  - First submission from VirusTotal: 2019-07-24 10:14:26 UTC
- Everything.exe
  - A normal application that lists all files on a file system. It allows an attacker to check whether a system is already infected by another piece of ransomware using the search function.
  - SHA1: c8107e5c5e20349a39d32f424668139a36e6cfd0
- NS.exe
  - A tool used to scan mounted and unmounted physical and network drives. Its ability to scan unmounted drives makes it very effective for ransomware attacks.
  - Detected by Trend Micro as HackTool.Win32.NetTool.A[20]
  - SHA1: 629c9649ced38fd815124221b80c9d9c59a85e74
  - It is highly similar to a sample analyzed by Hybrid Analysis.[21]

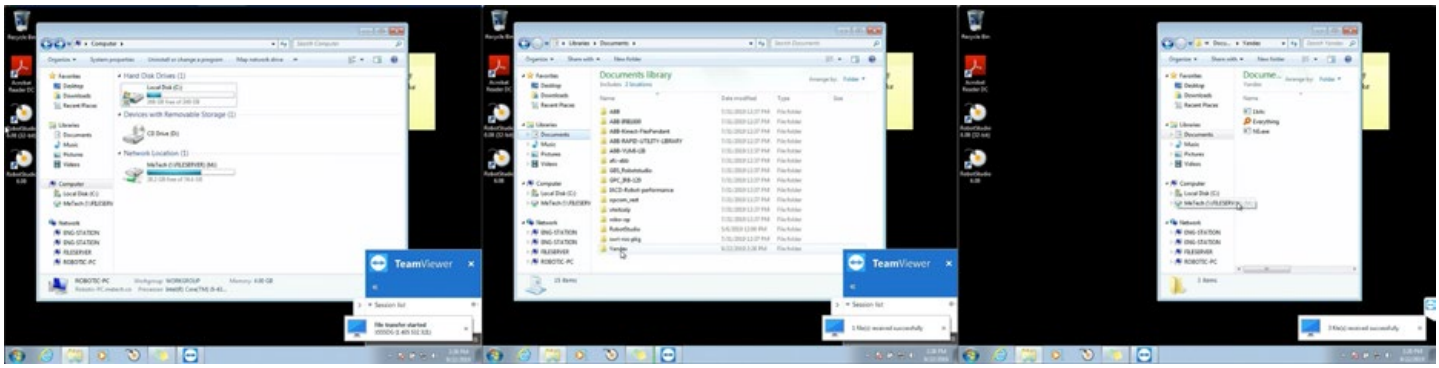


Figure 27. Downloading of files by the threat actor using TeamViewer

After downloading the files, they connected to the system using the computer name “X555DG” with a TeamViewer ID of “1 405 532 321”. They then started transferring the files to the Documents library under the subfolder they named Yandex.

After this point, the threat actor began running each of the downloaded files, beginning with NS.exe, the tool used to scan for mounted and unmounted drives. Next, they ran the Everything.exe file as an administrator. While this was running, they opened a command window and typed in the command

"vssadmin delete shadows /all", which is commonly used in ransomware attacks. Finally, they ran the 1btc.exe file, the Crysis ransomware variant, as an administrator. We were able to record all of these activities, as shown in Figures 28 through 30.

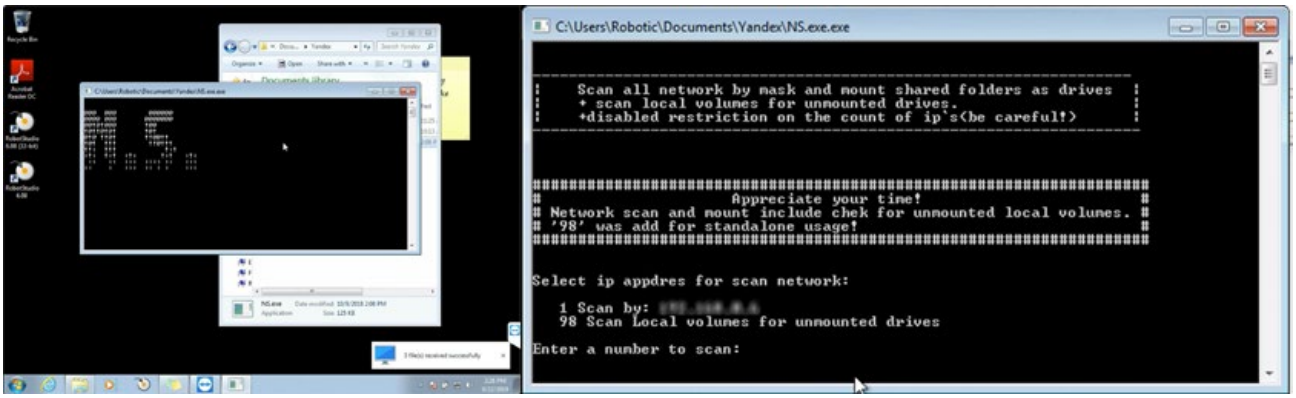


Figure 28. The NS.exe file being run

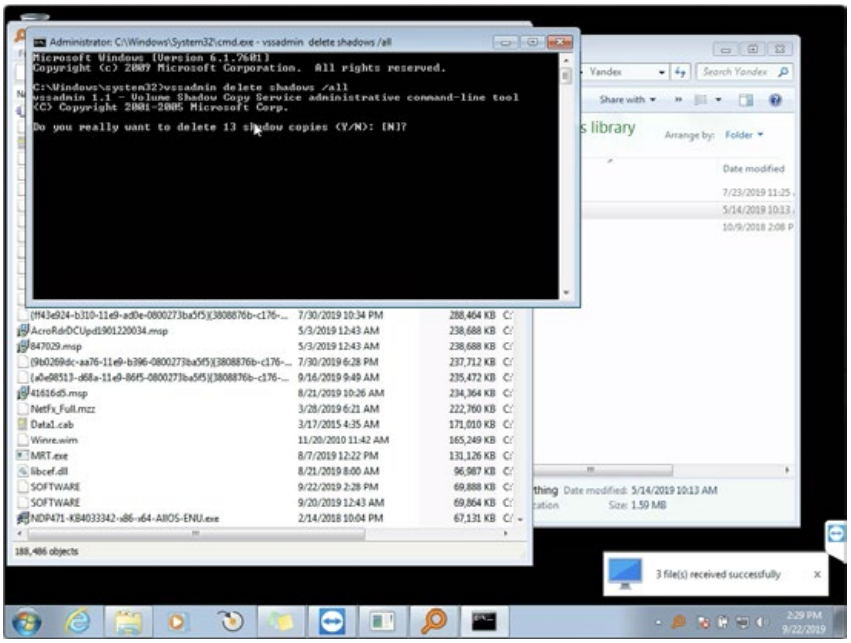


Figure 29. The command "vssadmin delete shadows /all" being run

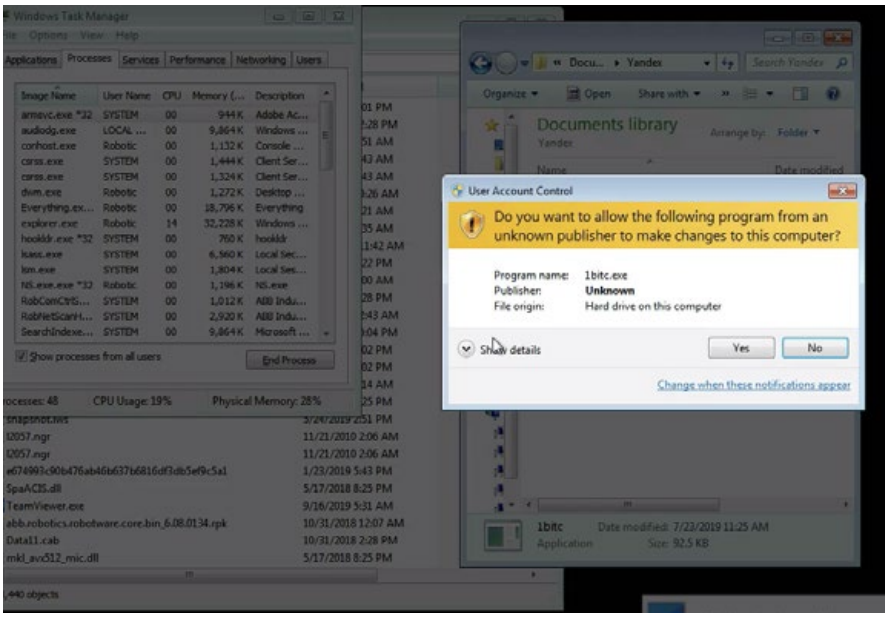


Figure 30. 1bitc.exe being run as an administrator

After setting all of these in motion, the threat actors watched and waited by opening the task manager. They even stopped other services to give their activities more processing power, as shown in Figure 31.

They then checked the result of their work by looking at all of the files listed in Everything, the otherwise legitimate tool used for listing files on a file system. As shown in Figure 32, the ransomware seemed to have successfully affected the files in our system. The threat actors even looked at a particular file (AcroRdrDCupd1901220034.msp.id-7C24B999) and checked its properties to confirm that the ransomware had worked.

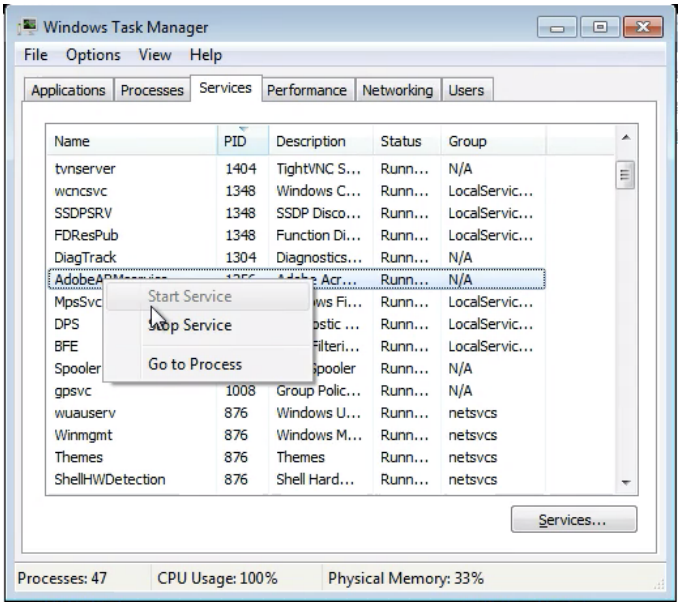


Figure 31. The threat actor viewing task manager and stopping services

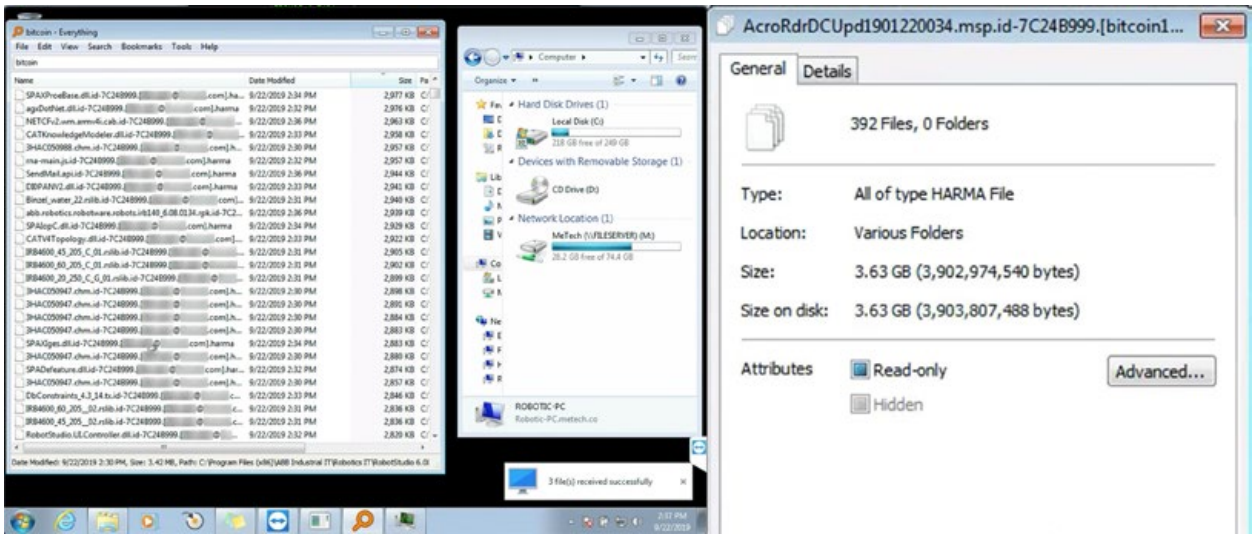


Figure 32. The threat actor checking files to see whether the ransomware had worked

Finally, with their work done, they closed TeamViewer. A ransom message then popped up, containing the typical content like the contact details of the threat actor, how to pay them in bitcoin, and the usual warning not to attempt to tamper with the encrypted files.

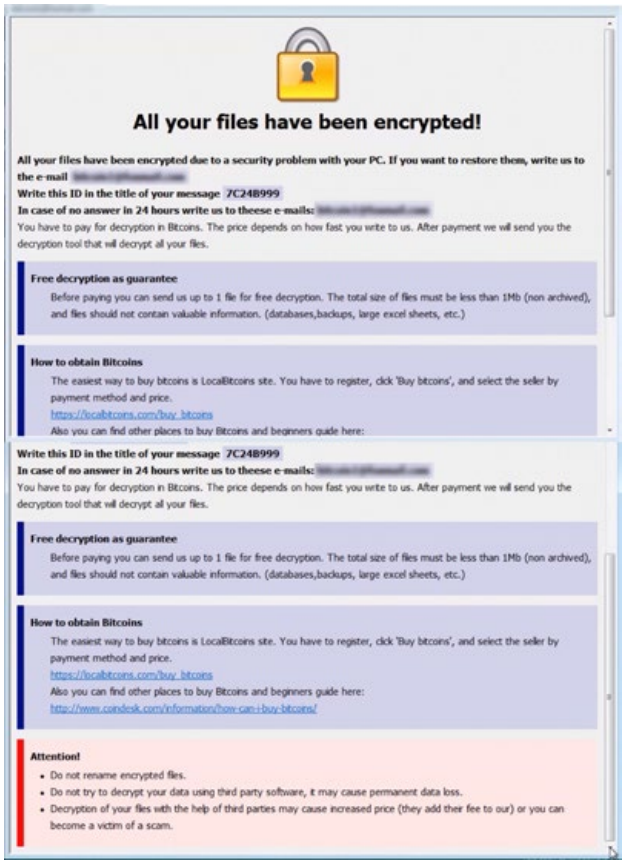


Figure 33. The ransom note that appeared after the threat actor closed TeamViewer

An actual company, upon realizing that its files have been encrypted and reading the ransom note, would have to go through several decision-making processes to handle such a situation. In our case, still posing as our cover company, we emailed the threat actor using the contact information they had left behind. Our first email was meant simply to engage the threat actor behind the provided email address. The reply we received was an obviously automated response, and it was followed a day later by an email asking for further details. These first few exchanges can be seen in Figure 34.

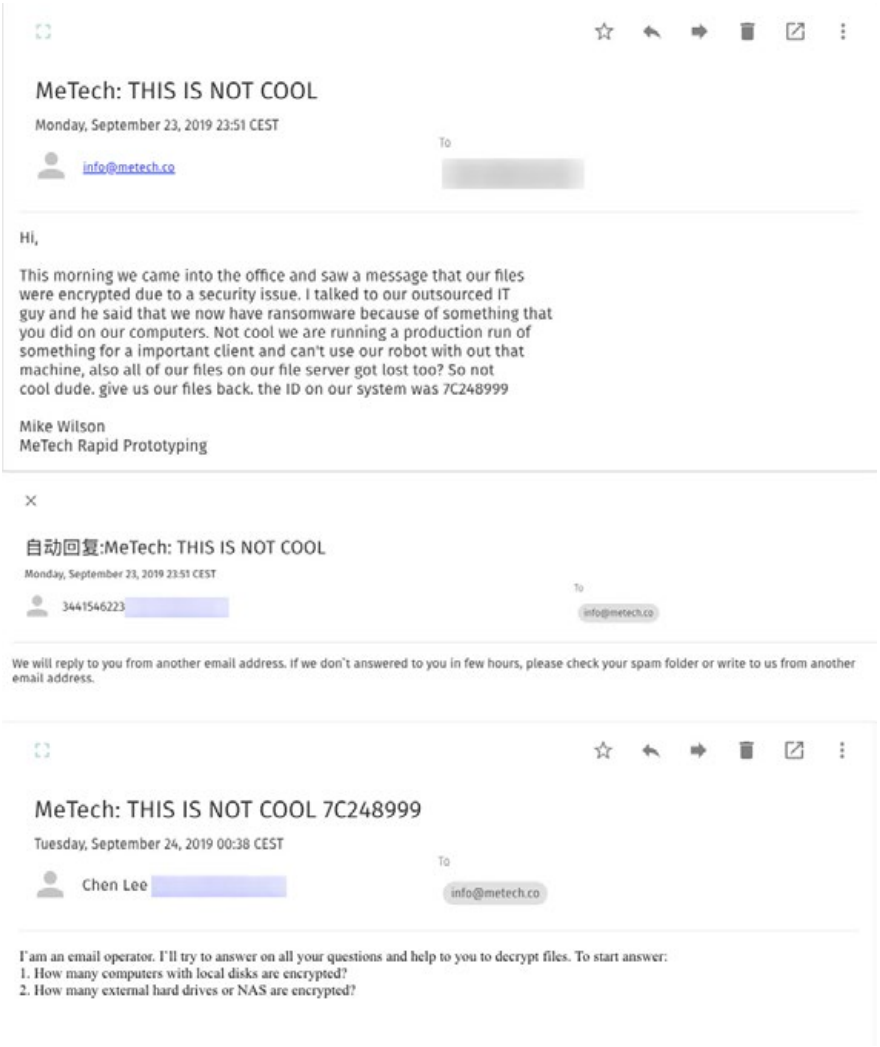
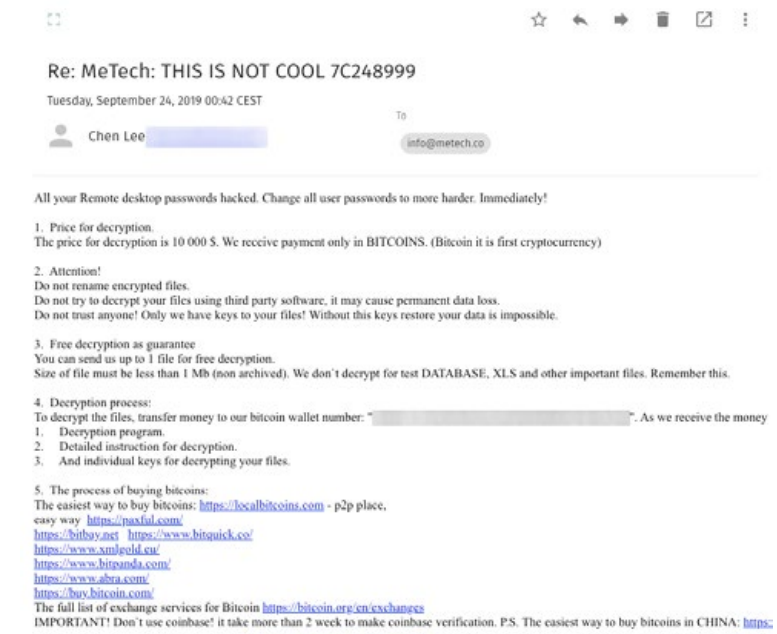


Figure 34. The first few emails exchanged with the threat actor

We responded to the email shown in the last image in Figure 34 by saying that one computer and one file server were affected in the attack. The next email from the threat actor contained a list of instructions and, more significantly, their demand for US$10,000 worth of bitcoin in exchange for having our files returned to normal, to be transferred to their wallet address, also specified in the email.



Re: MeTech: THIS IS NOT COOL 7C248999

Tuesday, September 24, 2019 00:42 CEST

To

Chen Lee [redacted] info@metech.co

All your Remote desktop passwords hacked. Change all user passwords to more harder. Immediately!

1. Price for decryption.
The price for decryption is 10 000 $. We receive payment only in BITCOINS. (Bitcoin it is first cryptocurrency)

2. Attention!
Do not rename encrypted files.
Do not try to decrypt your files using third party software, it may cause permanent data loss.
Do not trust anyone! Only we have keys to your files! Without this keys restore your data is impossible.

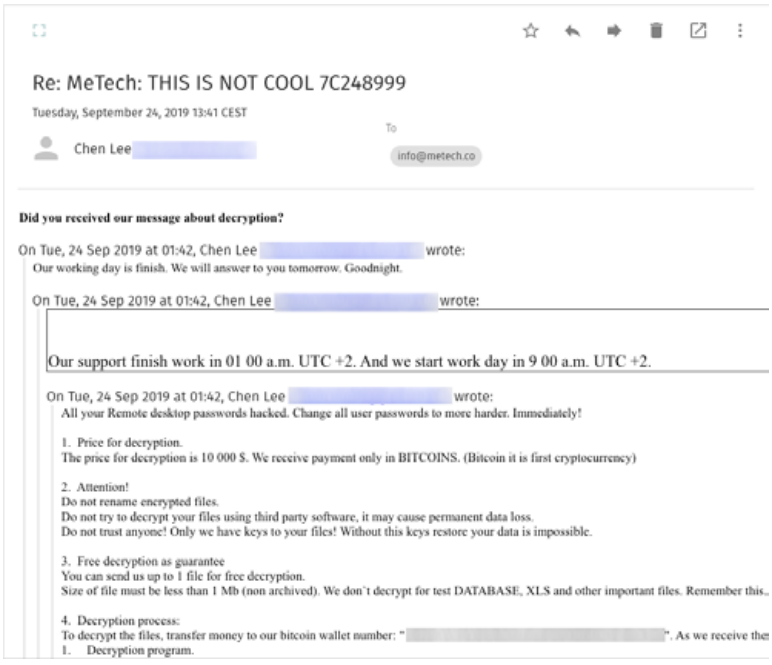
3. Free decryption as guarantee
You can send us up to 1 file for free decryption.
Size of file must be less than 1 Mb (non archived). We don`t decrypt for test DATABASE, XLS and other important files. Remember this.

4. Decryption process:
To decrypt the files, transfer money to our bitcoin wallet number: "[redacted]". As we receive the money
1. Decryption program.
2. Detailed instruction for decryption.
3. And individual keys for decrypting your files.

5. The process of buying bitcoins:
The easiest way to buy bitcoins: https://localbitcoins.com - p2p place,
easy way https://paxful.com/
https://bitbay.net https://www.bitquick.co/
https://www.xmlgold.eu/
https://www.bitpanda.com/
https://www.abra.com/
https://buy.bitcoin.com/
The full list of exchange services for Bitcoin https://bitcoin.org/en/exchanges
IMPORTANT! Don`t use coinbase! it take more than 2 week to make coinbase verification. P.S. The easiest way to buy bitcoins in CHINA: https:

Figure 35. The emailed list of instructions, including the ransom amount in bitcoin and the wallet address where the payment should be transferred

This email was followed by another email, this time informing us of the threat actor's working hours. This apparently served as a prelude to the next email, which simply stated that their working hours were done for the day and that further emails would be replied to the following day. It illustrated how organized the crime was from the end of the threat actor. When we did not respond for some time, they emailed us again to ask whether we had received their previous message, as shown in Figure 36.



Re: MeTech: THIS IS NOT COOL 7C248999

Tuesday, September 24, 2019 13:41 CEST

To

Chen Lee [redacted] info@metech.co

Did you received our message about decryption?

On Tue, 24 Sep 2019 at 01:42, Chen Lee [redacted] wrote:
Our working day is finish. We will answer to you tomorrow. Goodnight.

On Tue, 24 Sep 2019 at 01:42, Chen Lee [redacted] wrote:

Our support finish work in 01 00 a.m. UTC +2. And we start work day in 9 00 a.m. UTC +2.

On Tue, 24 Sep 2019 at 01:42, Chen Lee [redacted] wrote:
All your Remote desktop passwords hacked. Change all user passwords to more harder. Immediately!

1. Price for decryption.
The price for decryption is 10 000 $. We receive payment only in BITCOINS. (Bitcoin it is first cryptocurrency)

2. Attention!
Do not rename encrypted files.
Do not try to decrypt your files using third party software, it may cause permanent data loss.
Do not trust anyone! Only we have keys to your files! Without this keys restore your data is impossible.

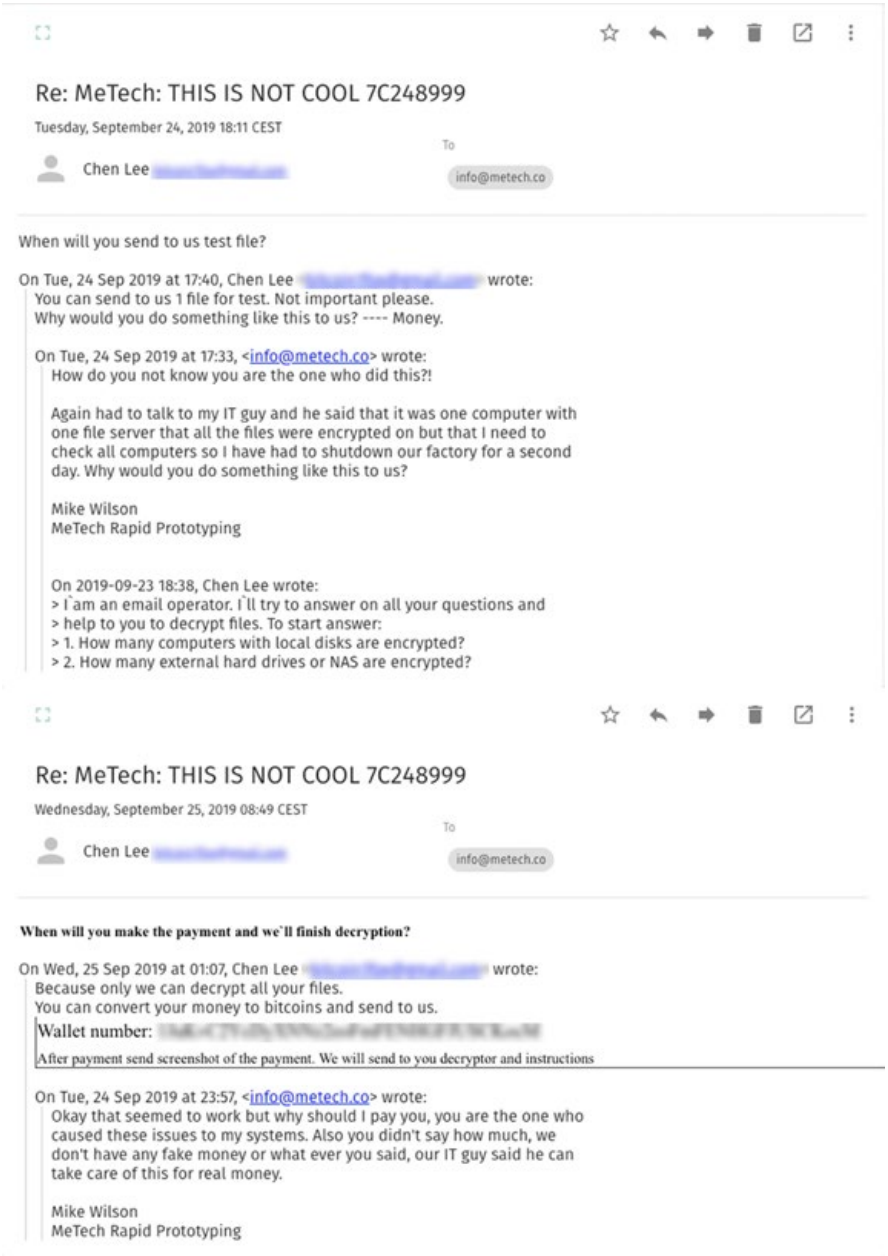
3. Free decryption as guarantee
You can send us up to 1 file for free decryption.
Size of file must be less than 1 Mb (non archived). We don`t decrypt for test DATABASE, XLS and other important files. Remember this..

4. Decryption process:
To decrypt the files, transfer money to our bitcoin wallet number: "[redacted]". As we receive ther
1. Decryption program.

Figure 36. Thread snippet showing the threat actor's working hours and follow-up email

In response, we sent an email asking them to decrypt a file as an example, to make sure that they did in fact have the decryption key. As shown in Figure 37, during this part of our exchange, we acted the part of a disgruntled company representative asking why the threat actor was doing this in the first place. They answered succinctly and obliged us by decrypting a sample file.

We sent them the conveyor belt PLC programing file (Omron CXP file), which they decrypted accordingly, suggesting that they were unaware that we had in fact sent them an important file. After resending the decrypted file, they reiterated their demand for and preferred mode of payment.



Re: MeTech: THIS IS NOT COOL 7C248999

Tuesday, September 24, 2019 18:11 CEST

To

Chen Lee [redacted] info@metech.co

When will you send to us test file?

On Tue, 24 Sep 2019 at 17:40, Chen Lee [redacted] wrote:

You can send to us 1 file for test. Not important please.
Why would you do something like this to us? ---- Money.

On Tue, 24 Sep 2019 at 17:33, <info@metech.co> wrote:

How do you not know you are the one who did this?!

Again had to talk to my IT guy and he said that it was one computer with one file server that all the files were encrypted on but that I need to check all computers so I have had to shutdown our factory for a second day. Why would you do something like this to us?

Mike Wilson
MeTech Rapid Prototyping

On 2019-09-23 18:38, Chen Lee wrote:
> I`am an email operator. I`ll try to answer on all your questions and
> help to you to decrypt files. To start answer:
> 1. How many computers with local disks are encrypted?
> 2. How many external hard drives or NAS are encrypted?

Re: MeTech: THIS IS NOT COOL 7C248999

Wednesday, September 25, 2019 08:49 CEST

To

Chen Lee [redacted] info@metech.co

When will you make the payment and we`ll finish decryption?

On Wed, 25 Sep 2019 at 01:07, Chen Lee [redacted] wrote:

Because only we can decrypt all your files.
You can convert your money to bitcoins and send to us.
Wallet number: [redacted]
After payment send screenshot of the payment. We will send to you decryptor and instructions

On Tue, 24 Sep 2019 at 23:57, <info@metech.co> wrote:

Okay that seemed to work but why should I pay you, you are the one who caused these issues to my systems. Also you didn't say how much, we don't have any fake money or what ever you said, our IT guy said he can take care of this for real money.

Mike Wilson
MeTech Rapid Prototyping

Figure 37. Thread snippet showing our request for a sample decrypted file and the other party agreeing to it

We continued the exchange by attempting to haggle. Ultimately, we managed to reduce their price to US$6,000 worth of bitcoin from the original US$10,000, as shown in Figure 38.

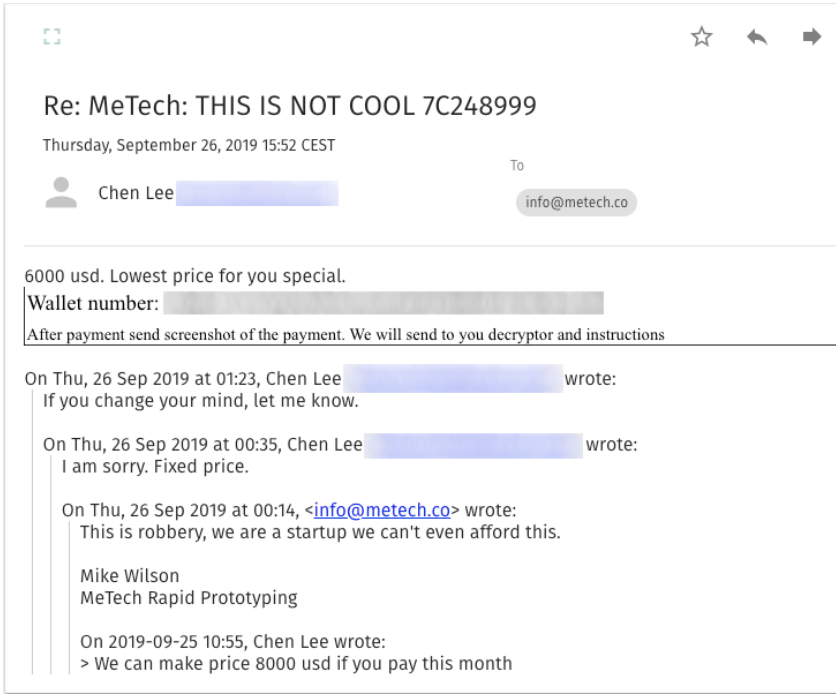
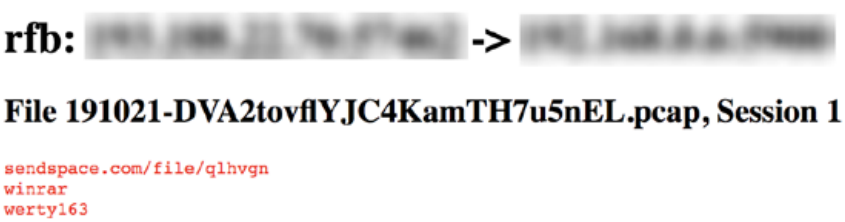


Figure 38. The last part of our exchange, where we haggled for the price of the ransom

When this attack had run its course, we simply reset the system after getting all the information we could.

Phobos Ransomware

On Oct. 21, another ransomware attack occurred on the honeypot. The event took almost an hour and consisted of the threat actor browsing the file system, scanning the network, and deploying the ransomware. We later found out that the ransomware used for this attack was named Phobos, detected by Trend Micro as Ransom.Win32.PHOBOS.SM.[22] We recorded notable keystrokes that the attacker typed in our honeypot network, as shown in Figure 39.



File 191021-DVA2tovflYJC4KamTH7u5nEL.pcap, Session 1

```
sendspace.com/file/qlhvgn
winrar
werty163
```

Figure 39. Notable keystrokes from the second ransomware infection

The threat actor visited the link “sendspace[.]com/file/qlhvgn” to download a RAR archive. This archive had the filename “remove backups.rar” (SHA1: ef1418e3fcdcca4410014948116a28fa47e74fe2) and contained the files for the attack, as shown in Figure 40 and detailed in Table 2. Perhaps noticing that there was no archiver utility installed on our system, they decided to download WinRar as well. They opened the archive, which was protected with the password “werty163”, as we found logged from the network traffic.

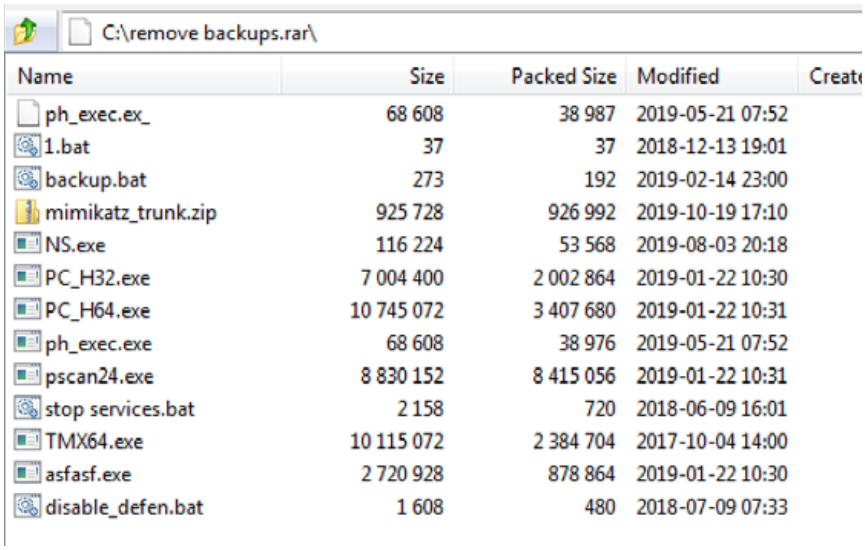


Figure 40. The contents of the RAR archive

Filename	SHA1	Trend Detection	Description
1.bat	8ecff105db88464edf548b542a7837e92e56fcbe		Deletes all shadow copies
NS.exe	f628f11e39d2ce90e49de8774df40a248a6abcff		Network scanner
PC_H32.exe	c4e2953509e9a47d9ee0ecfa8c886328d700ed7e		PC Hunter, an analysis tool for Windows
PC_H64.exe	d373052c6f7492e0dd5f2c705bac6b5afe7ffc24		PC Hunter, an analysis tool for Windows
TMX64.exe	5ce6f58f46dc8ab89fd8bfc994dabb50316e7202		Task Manager Deluxe
asfasf.exe	75ba2e4bfb47feed72deed2bed9b2ef698e3253f		Process Explorer
backup.bat	86f599090aa2c7c1df65dccccf00e1818e72246a		Deletes all shadow copies
disable_defen.bat	c17f4d57deb93050d094e5a09d2f9e58abc252f9		Disables Windows Defender
mimikatz_trunk.zip	ebababb9c5b723df0fde7fe02dc22145e39ba0502	HKTL_MIMIKATZ.component	Mimikatz files
ph_exec.exe	2be826b4864f86c37592a2e908638873b5ff093c	Ransom.Win32.PHOBOS.SM	Phobos ransomware used in the attack
pscan24.exe	47dfbbbce8170891ddfbdcdd4e6a24d465d847e1		Port scanner
stop services.bat	8b77e8888276c8ce99746a7c0d5ca3f93ea9dee8		Batch file that stops database services (e.g., MSSQL, MySQL, PostgreSQL) and Windows Defender

Table 2. Details of the files from the archive

Phobos has similar attributes to Crysis, which was the ransomware variant used in the previously discussed attack. The screenshot in Figure 41 shows the ransom note that was displayed after the malware was executed and also after the system was rebooted. Encrypted files were renamed with the file extension .actin, as shown in Figure 42.

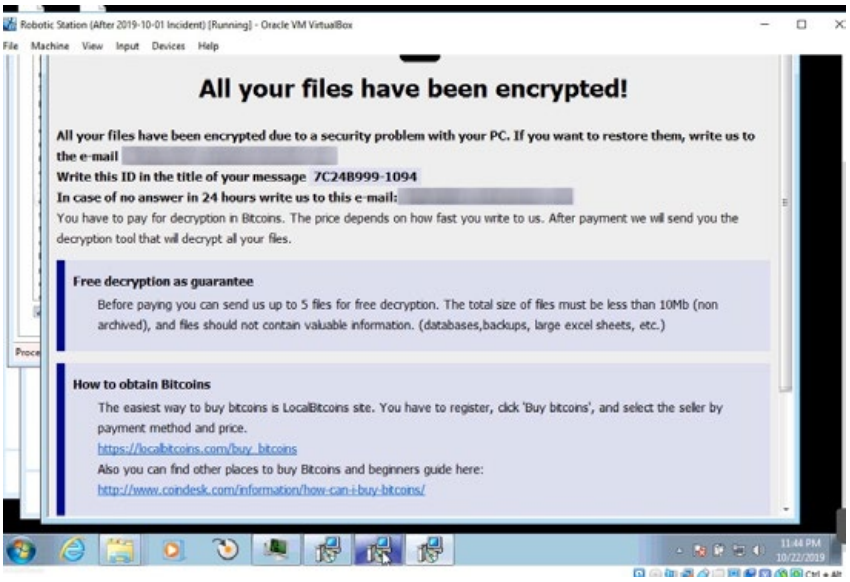


Figure 41. The ransom note left by the Phobos ransomware

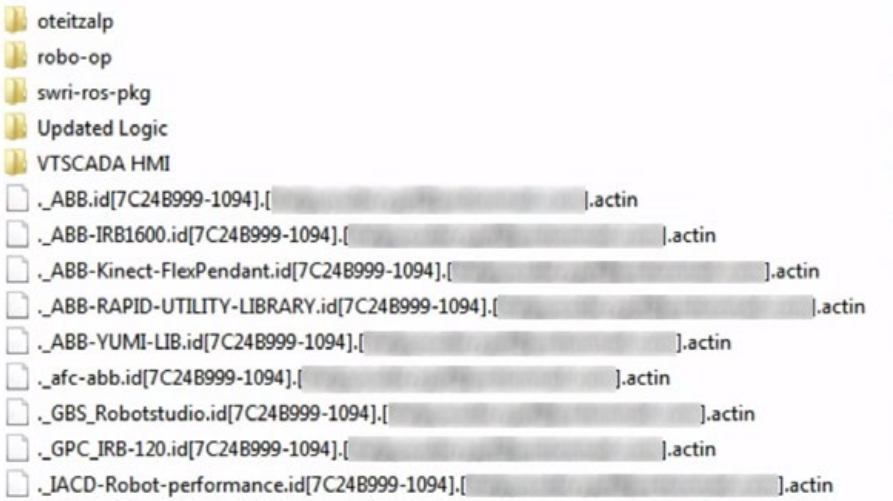


Figure 42. Encrypted files from the Phobos ransomware attack

A Fake Ransomware Attack

Several weeks after the second ransomware attack, on Nov. 12, another threat actor came in and dropped "ransomware" in our system. This threat actor fumbled around our system trying to get a PowerShell command to work.

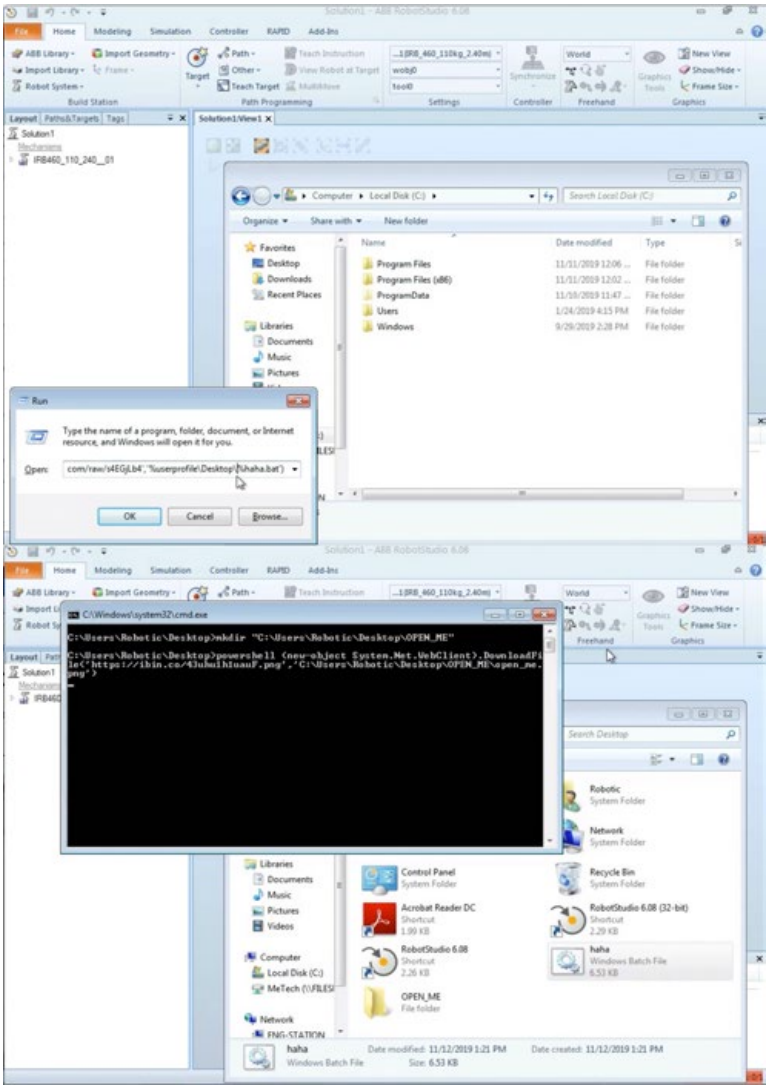


Figure 43. The threat actor attempting to run a PowerShell command

When they were able to get it to work, it downloaded a file called haha.bat. We watched them struggle to get this tool to work. They kept changing the bat file to "haha.rnsmwr", as shown in Figure 44, but they later renamed it back to "haha.bat". This confused us until we saw that the "ransomware" was really just using the REN or rename command.

At one point, as shown in Figure 45, the threat actor even edited haha.bat, which gave us a glimpse of the code as its Pastebin page was no longer active. Even though we were watching them perform this attack live, they were still quick to close the haha.bat code.

Interestingly, after all of these actions, they also made sure that the ABB directory was also “ransomed” or renamed, as shown in Figure 46.

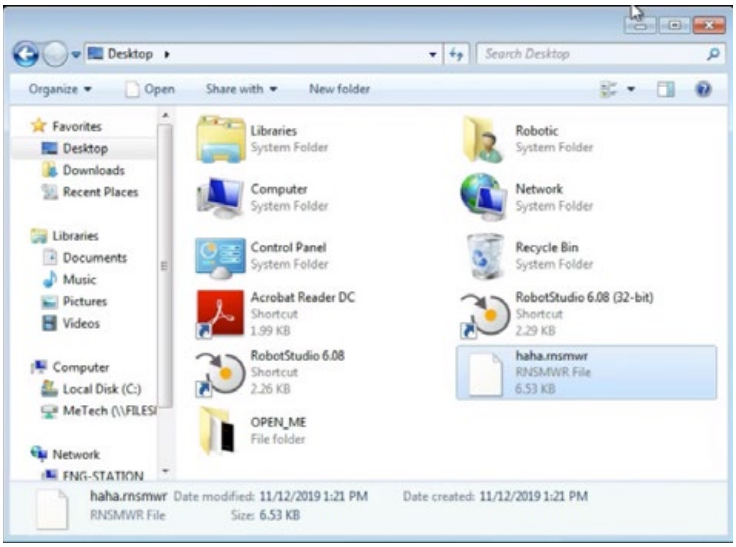


Figure 44. The threat actor renaming haha.bat to haha.rmsmwr

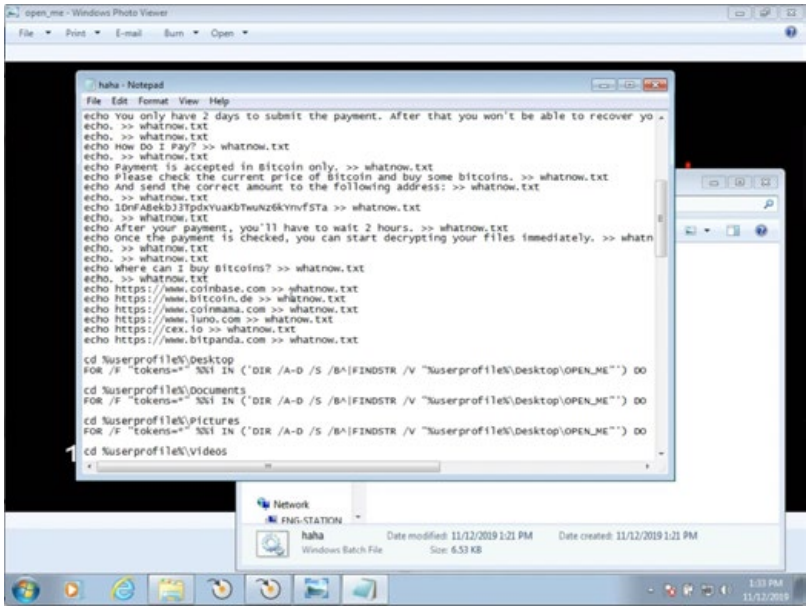


Figure 45. The threat actor editing the haha.bat file

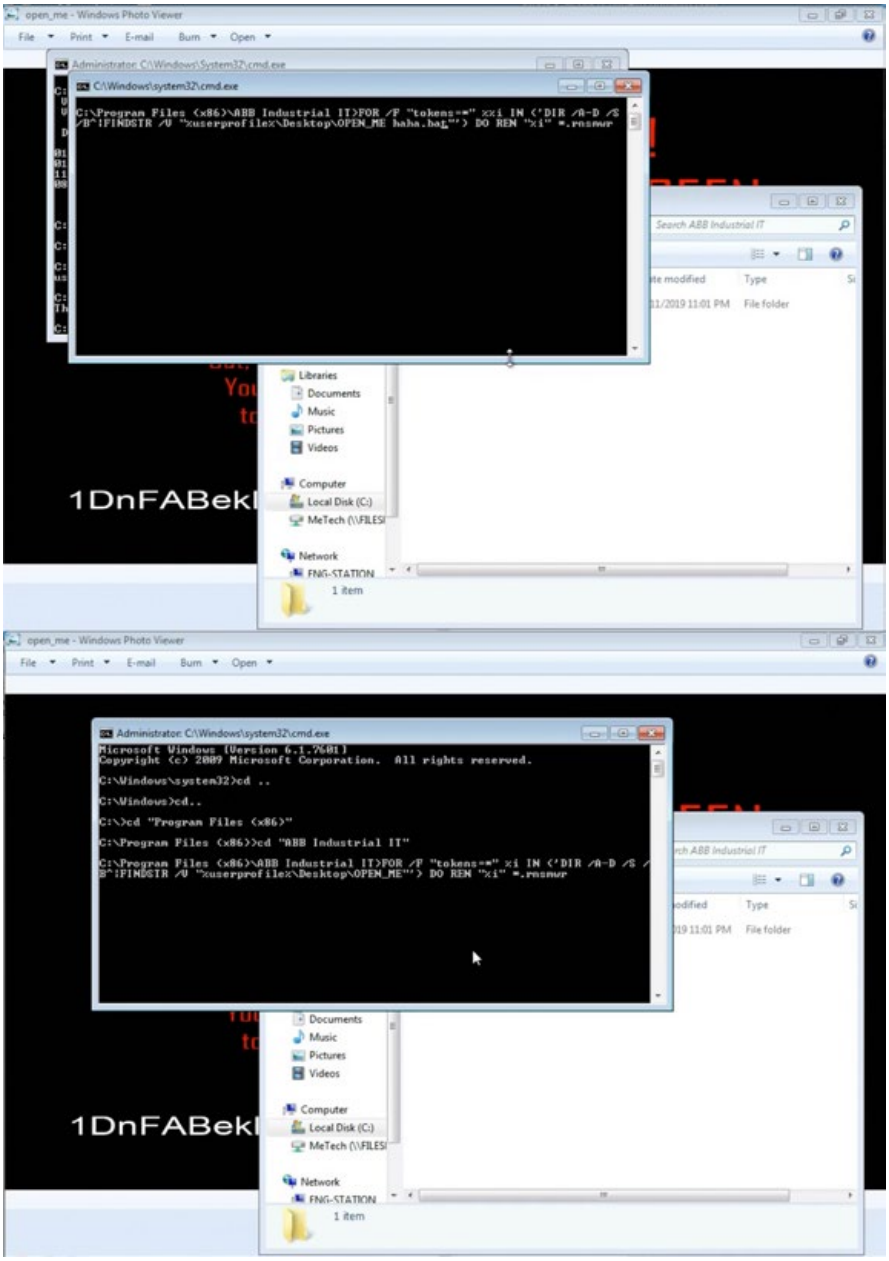


Figure 46. The threat actor renaming the ABB directory

They then moved on to editing the ransom message. First, they changed the wallet address. Then, they changed the payment they were demanding, from US$200 to US$750. They also assigned passwords to VNC so that only they would have access. They used the admin password “#concreTec” and the view password “serfcx”. These actions are reflected in Figures 47 through 49.

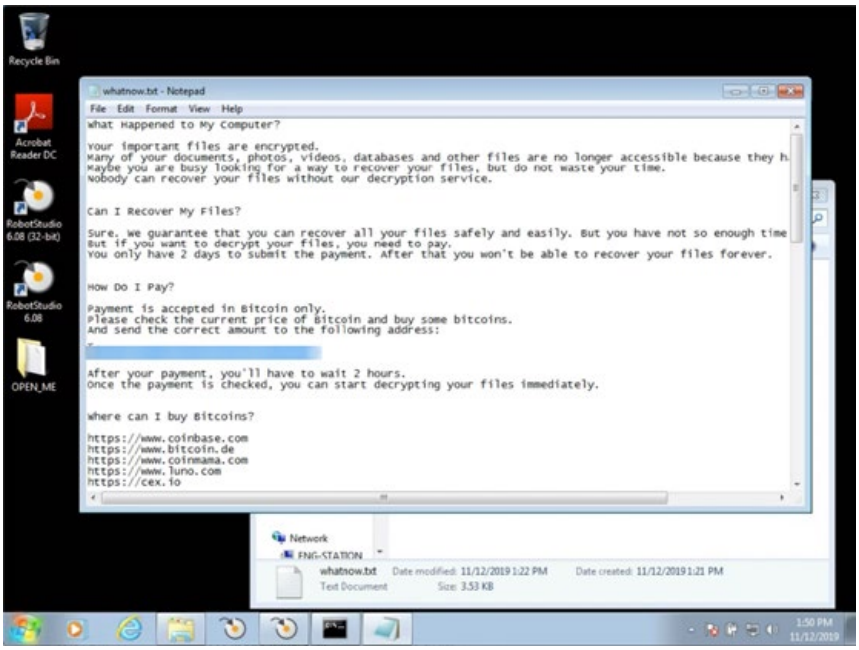


Figure 47. The threat actor changing the wallet address

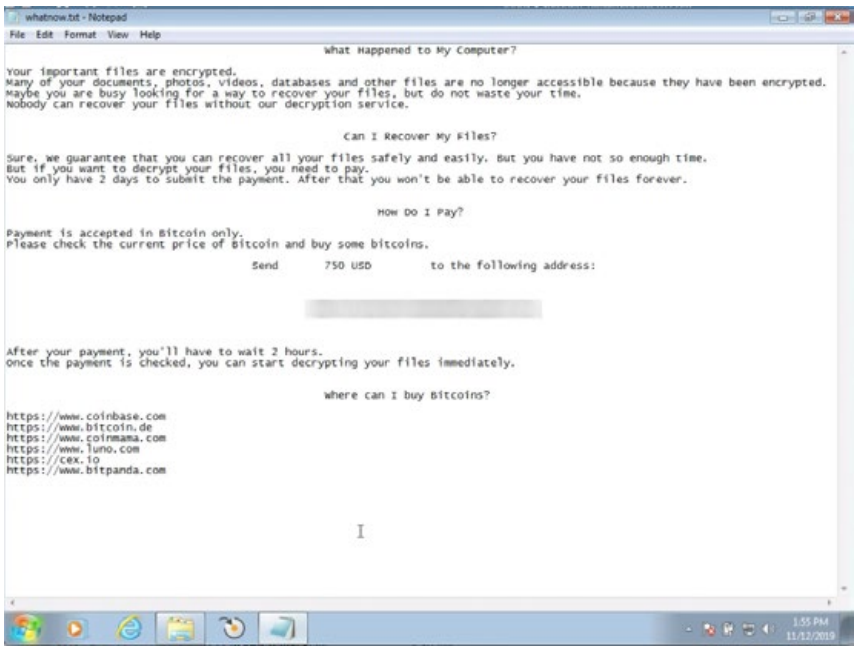


Figure 48. The threat actor deciding on the US$750 ransom

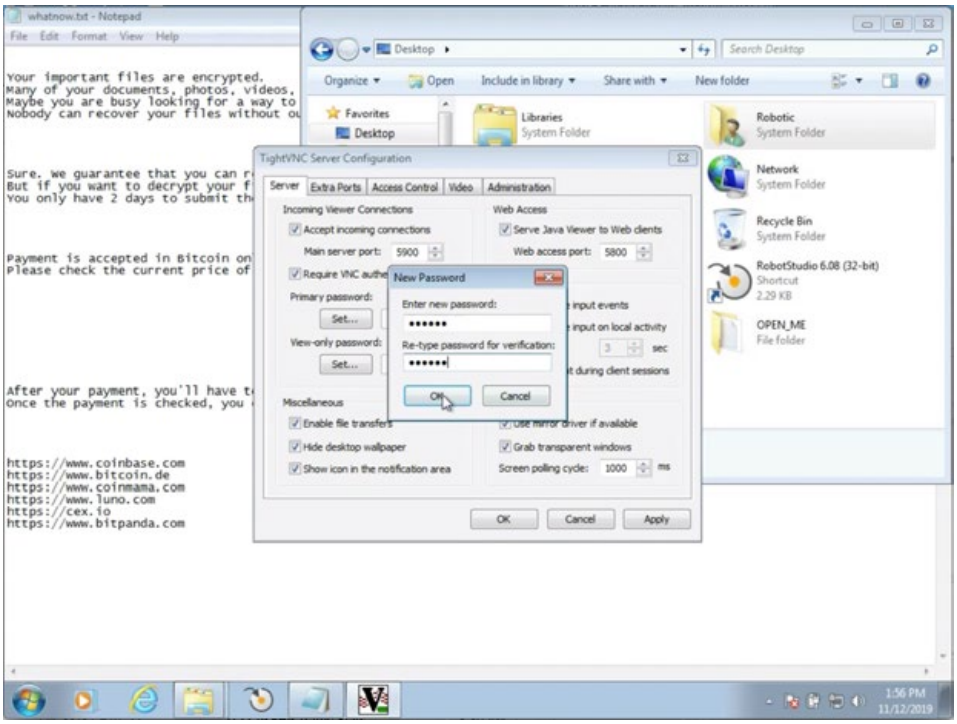


Figure 49. The threat actor assigning VNC passwords

They took a few final measures after all this had been set up. They again made sure that the ABB folder had been "encrypted." Then, they cleaned up the registry by editing some of the registry settings that were modified during the process. And finally, they changed the desktop background image into their ransom note before leaving our system. These actions are reflected in Figures 50 to 52.

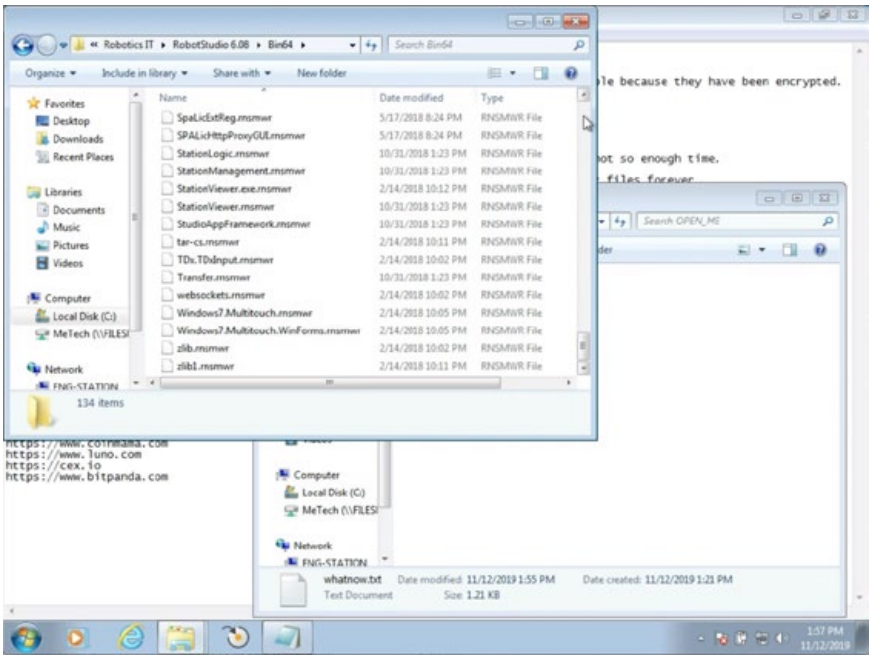


Figure 50. The threat actor checking that the ABB files had been "encrypted"

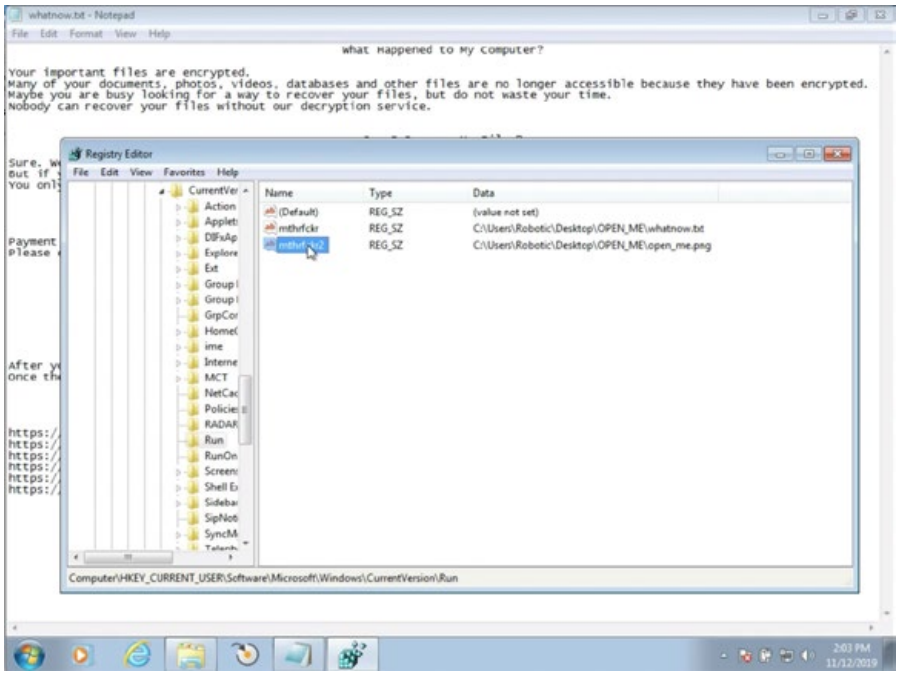


Figure 51. The threat actor cleaning up the registry

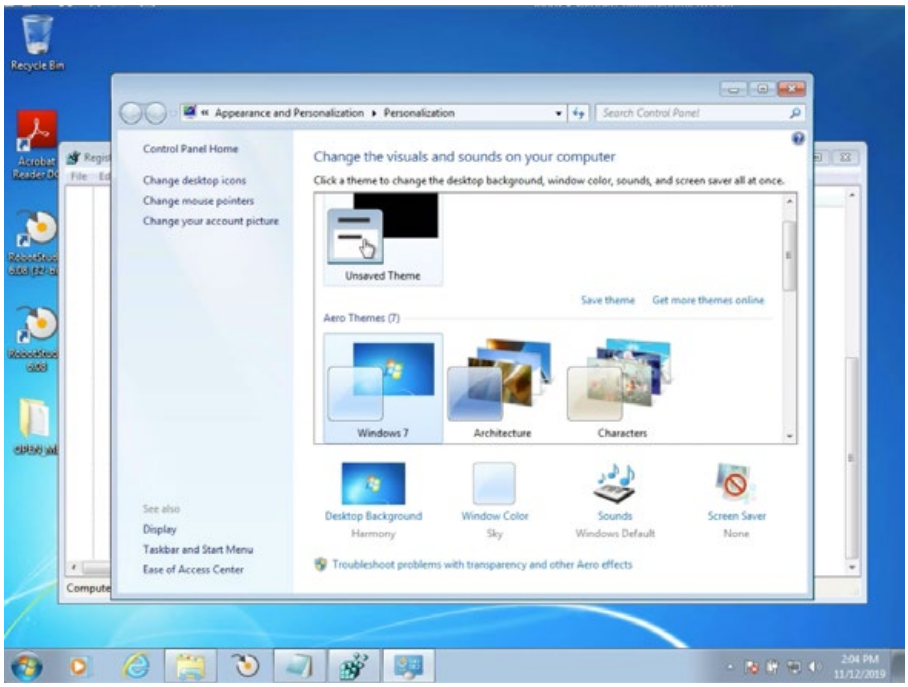


Figure 52. The threat actor changing the desktop background image to the ransom note

Two days later, on Nov. 14, they came back into our system — that is, we had assumed that it was the same threat actor based on the unfolding behavior. They went into the Documents library and deleted everything that was in it, as shown in Figure 53. It should be noted that this was the library where the "ransomware" actor spent some time on during their first visit to the system.

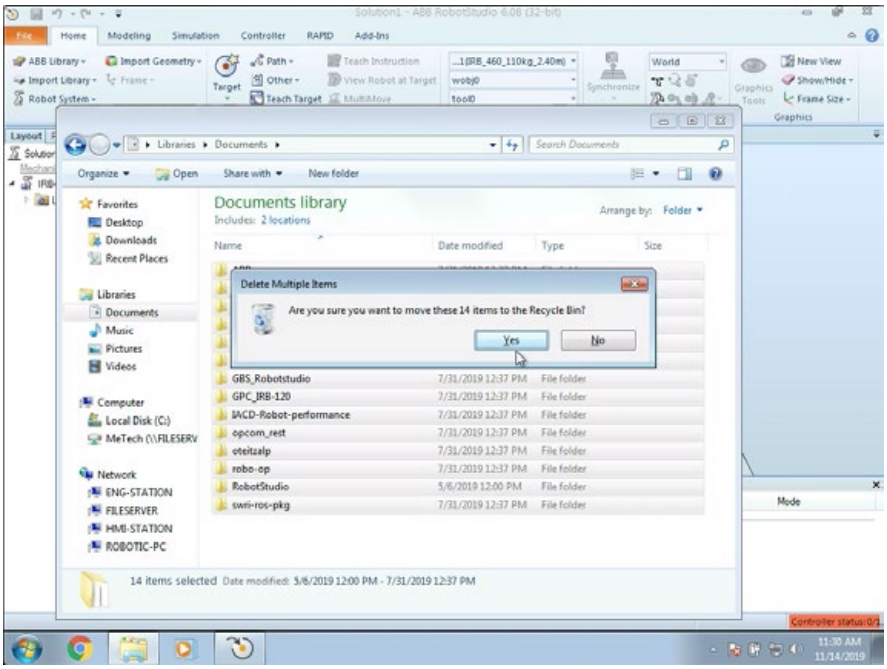


Figure 53. The threat actor deleting all of the files in the Documents library

They then created a program in Notepad that launched several de[.]youporn[.]com tabs. They executed this program before leaving our system again. And sure enough, several tabs of the porn site were opened. These actions, which are reflected in Figure 54, were likely meant to garner more attention, after the threat actor's first attempt had gone unnoticed for several days.

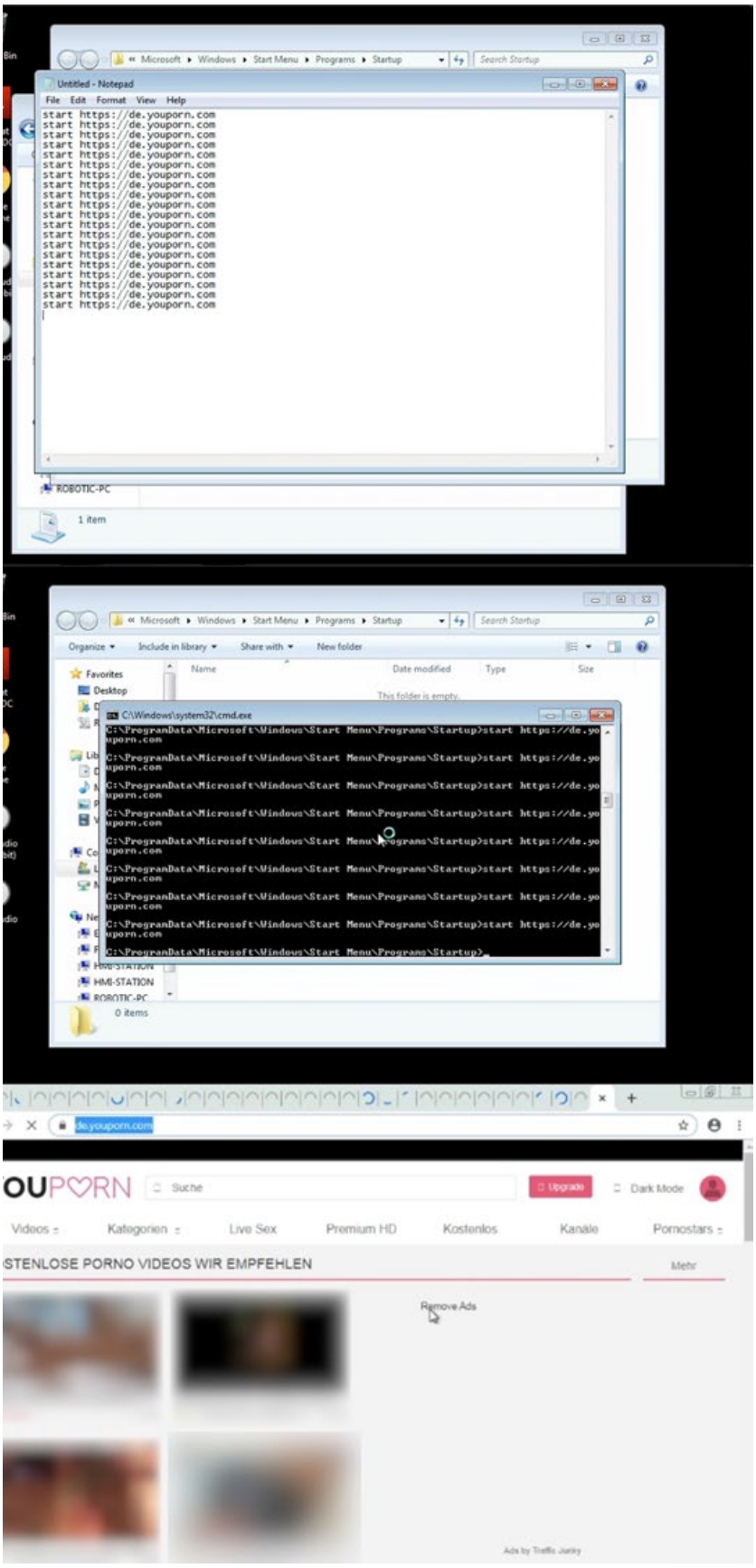


Figure 54. The threat actor leaving a wake of porn site tabs during their second visit to our system

Attack With a Beacon

On Oct. 16, an actor came into the robotics workstation via VNC. As it turned out, their intention was to send a beacon likely for lateral movement.

They went to https://www[.]sendspace[.]com/file/fjtdsk and downloaded a file called nsis.exe (SHA1: 00a31ed29c06c06dde3433a5d6fa0a5dc941f13e), a self-extracting archive with several encrypted files in it. We detail the contents of the downloaded file in Tables 3 and 4, the former pertaining to the encrypted version and the latter to the decrypted version.

Filename	SHA1	Encrypted	Description
$PLUGINSDIR\System.dll	f7543f9e9b4f04386dfbf33c38cbed1bf205afb3	No	
$TEMP\System.Data.SQLite.dll	42d5708ee9b662fae73e78f0fd0c5228090c3b40	No	Legitimate SQLite library for retrieving stored passwords in Chrome browser
$TEMP\ak.tmp	1775f9cb1829910dce7b412c2e7b1b701c23709e	Yes	
$TEMP\ak_1.tmp	b5931a99036a9a874cb917b6992e7c4510f063c2	Yes	
$TEMP\config.tmp	e355b51cf1b98c5d9513ff0752b59e8ab09e93d4	Yes	
$TEMP\installer.ps1	552c69ab13fbc4ed770b4bed69474fbf32ba6f4b	No	Main script that will be executed after extraction of archive. This is responsible for decrypting the component files and installing the backdoor malware. Detected by Trend Micro as Trojan.PS1.CREDSTEAL.SM.
$TEMP\migwiz.tmp	d5d02092dd453185f94f5882ffa090a0358be774	Yes	
$TEMP\migwiz_1.tmp	a2ca90c6b6efce5b85335b0cc3ecca07c024dcc0	Yes	
$TEMP\rdpclip.tmp	7da837d644123e3547464273756800f22b0ed034	Yes	
$TEMP\rfxvmt64.tmp	1885f2a4a58fb77c49763e09189aa3c1ec4eaa27	Yes	
$TEMP\termsvc.tmp	4a6ab099aec72b4ca6b82db088e308d5542e1242	Yes	
$TEMP\termsvc_1.tmp	e774f3e8379615eaffb7c998c743ec119aa7b481	Yes	

Table 3. The list of encrypted files from the downloaded nsis.exe file

Filename	SHA1	Description
ak.bin	3192ad3118b8c1eb5ee46764920a7d9120ca02e1	UAC bypass binary
ak_1.bin	61a6b265bc612d97589dddd65e8d31cc9f0625ea	64-bit version of ak.bin
config.bin	91c24a33a616168604645aacc01f32c9beac92aa	RDP config
migwiz.bin	fd4552e078bcae7134a3008d3b342011d835b007	
migwiz_1.bin	554116aabd804663c24d8b3fa41cb72c00dc5b34	64-bit version of migwiz.bin
rdpclip.bin	306498e9a9f1c6b2813dad7cdcd8433139201794	Legitimate Microsoft binary - rdpclip.exe (RDP Clipboard Monitor)
rfxvmt64.bin	81d4ad81a92177c2116c5589609a9a08a5ccd0f2	Legitimate Microsoft binary - rfxvmt.dll (Microsoft RemoteFX VM Transport)
termsvc.bin	34dd125d42fdb33d2108896ff276cbfe71154cca	
termsvc_1.bin	8ffe80190f7662422bf6c5736a01ea26880b74a2	64-bit version of termsvc.bin

Table 4. The list of decrypted files from the downloaded nsis.exe file

After the PowerShell command was executed, it decrypted the component files and dropped them in the Windows temp folder. It then terminated Remote Desktop Services and replaced rdpclip.exe and rfxvmt.dll with older versions. It also replaced the service dynamic link library (DLL) used by RDP from the registry.

```
net localgroup $adm $objUser.Value /add
reg add "HKLM\system\currentcontrolset\services\TermService\parameters" /v ServiceDLL /t REG_EXPAND_SZ /d %SystemRoot%\temp\termsvc.dll /f
reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f
#reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fSingleSessionPerUser /t REG_DWORD /d 0 /f
```

Figure 55. The threat actor replacing the DLL used by RDP from the registry

They then retrieved stored cookies, tokens, and credentials from the Chrome browser and wrote them on the following files:

- c:\windows\temp\cookies.txt
- c:\windows\temp\tokens.txt
- c:\windows\temp\logins.txt
- c:\windows\temp\logins_read.txt

They then reconnected to VNC. A few minutes later, the robotics workstation began beaconing out to afsasdfa33[.]xyz, via HTTPS (port 443), the certificate of which was generated with Let's Encrypt, a free certificate authority.[23]

At the time of writing, it was still unknown to us what exactly was being sent out by our robotics workstation. However, we were still looking into the possibility that it was for lateral movement.

Control System Attacks

As part of our conceptualization of our honeypot, we used PLCs from several different vendors to see the possible attacks their used protocols would be prone to. We also left these PLCs somewhat exposed and inadequately protected. In this subsection, we return to these PLCs and discuss the possible attacks we observed on them.

Control System 'Attacks'

In our Moloch system, as in Wireshark, it is possible to filter down. Since we wanted to see whether there were any attacks on our exposed PLCs, we tried filtering out all known scanners, a process we described earlier. As previously mentioned, doing so required taking the list of IP addresses, resolving them, and excluding those that had host names that tracked back to a known internet scanner.

Excluded Scanners

We spent quite some time building a reliable whitelist of internet scanners, which proved useful in excluding benign traffic to the exposed ports. The result is shown in Table 5.

51.15.191.81	80.82.77.139	146.88.240.6
51.254.49.101	82.221.105.7	172.105.207.40
68.169.145.238	89.248.167.131	185.142.236.34
71.6.135.131	89.248.168.51	185.142.236.35
71.6.146.130	89.248.172.16	185.173.35.0/24
71.6.146.185	89.248.174.3	185.181.102.18
71.6.146.186	92.118.160.0/24	185.216.140.6
71.6.147.254	93.174.85.106	195.154.61.206
71.6.158.166	93.174.95.106	198.20.70.114
71.6.165.200	94.102.49.190	198.20.99.130
71.6.167.142	104.251.248.86	198.108.66.0/24
71.6.199.23	139.162.65.76	208.64.252.230
80.82.77.33	139.162.83.10	212.83.146.233
	139.162.99.243	

Table 5. The IP addresses of excluded scanners

What this left us with was traffic to our PLCs that could be malicious or that could be originating from scanners that are not well known. We determined the nature of the traffic by exporting and manually verifying the PCAPs from Moloch. To do this, we filtered down by the protocol used by each PLC vendor.

Siemens S7-1200 PLC

The Siemens S7-1200 PLC sat on the honeypot network and was remapped or NAT-ed from port 102 to port 102. What we observed were hosts on the internet using valid commands to request for the CPU functions. These hosts would make the request and the PLC would respond with the basic hardware information. This could be seen as the same information that was collected by Shodan, as shown in Figure 56.

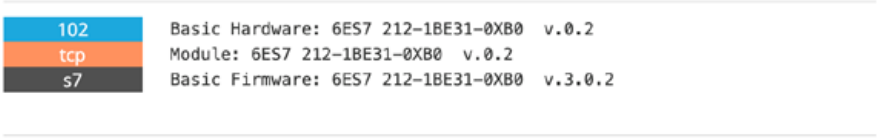


Figure 56. Shodan information on Siemens S7-1200 PLC

In 2012, the independent cybersecurity researcher group SCADA StrangeLove released a tool called PLCScan[24] (now called s7scan[25]). It was a Python script that helped pull information about Siemens S7 PLCs to aid in identifying PLCs on the network. Shodan took this script and started scanning the internet with it in 2015. Digital Bond built this into its Redpoint framework, and took PLCScan and made it into an Nmap script.

From what we saw, all traffic to our PLCs that was not scanner-related was only using PLCScan (s7scan) or using s7-info.nse (see Figure 57), which was released in 2015 into the main branch of Nmap. (The file extension “nse” stands for Nmap Scripting Engine.) No other requests or commands were sent to our Siemens S7-1200 PLC at the time of writing, both from known scanners and non-scanners.

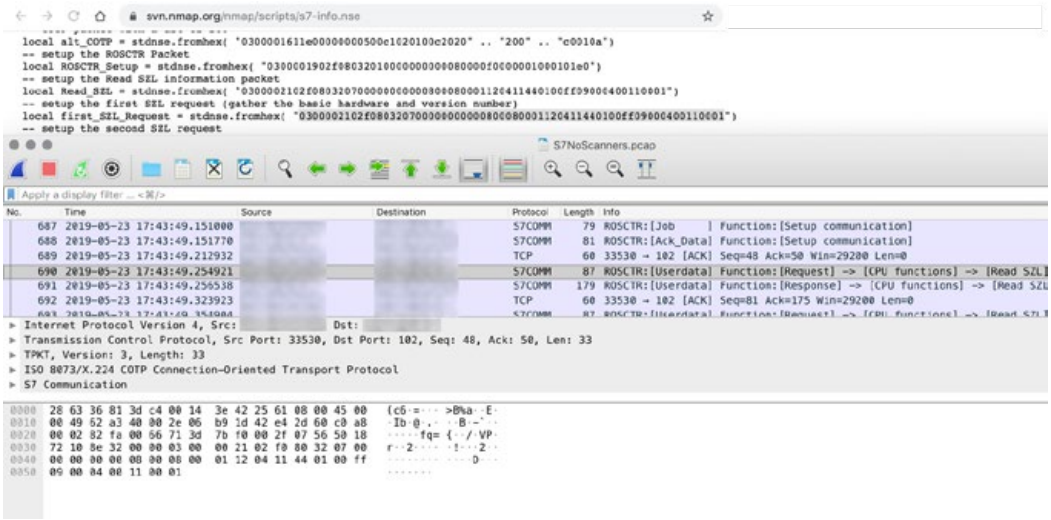


Figure 57. Comparing the Siemens S7-1200 PLC traffic with s7-info.nse

The traffic was not inherently harmful. From our perspective, the traffic was simply from unknown scanners. However, we are not discounting the possibility that this could be part of a reconnaissance activity for further attacks that were never seen.

Allen-Bradley MicroLogix 1100 PLC

One of the two Allen-Bradley MicroLogix 1100 PLCs on the honeypot network was NAT-ed through 44818 to port 44818, while the other one was not exposed. For our discussion, we consider only the one that was exposed as no attacks were seen laterally moving throughout the network to affect the other PLC.

In 2014, similar to what it did with the Siemens S7, Digital Bond added enip-info.nse to the main branch of Nmap. This script sends a command 63 (request identity) to the PLC, to which the PLC will respond with information about itself, which is the same information shown in Shodan (see Figure 58).

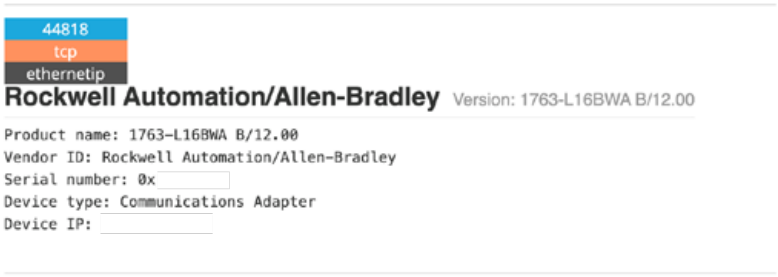


Figure 58. Information on an Allen-Bradley PLC as shown in Shodan

With some understanding of EtherNet/IP, the requesting host sets the sender context; this is changed based on the station that sends the information. However, when the Nmap script was written, we found that the sender's context was set to "0x0000c1debed1". This was the same for the majority of the scans that went against our exposed Allen-Bradley MicroLogix PLC (see Figure 59).

This led us to believe that the majority of the traffic we saw was properly formed EtherNet/IP traffic, indicating that most of the traffic was based on the Nmap script and was likely from scanners that are not well known. As with the Siemens S7-1200, these were also good recon scripts for determining whether an exposed device was a PLC, HMI, or any other type of EtherNet/IP-supported device.

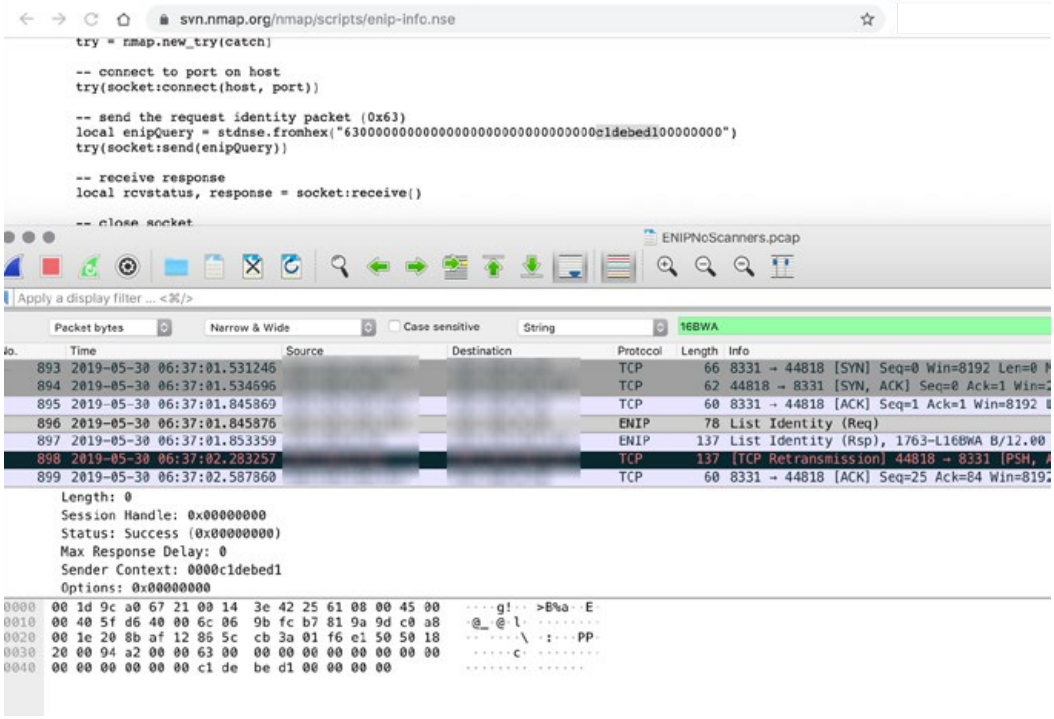


Figure 59. Comparing the Allen-Bradley MicroLogix 1100 PLC traffic to enip-info.nse, where the sender context is the same

While most of the information we saw was only “List Identity(Req)”, we did see a number of “unknown commands” (see Figure 60 for an example). However, looking further revealed that these unknowns were random information being sent to the port 44818. While in this case the PLC would respond simply by saying that it was an unknown command, sending unknown traffic to known ICS protocol ports still remains a dangerous practice that could cause older devices to crash. And while we never encountered an issue because of this, it could have eventually caused an issue, as had been shown by Cisco Talos in its released vulnerabilities for Allen-Bradley MicroLogix PLCs.[26]

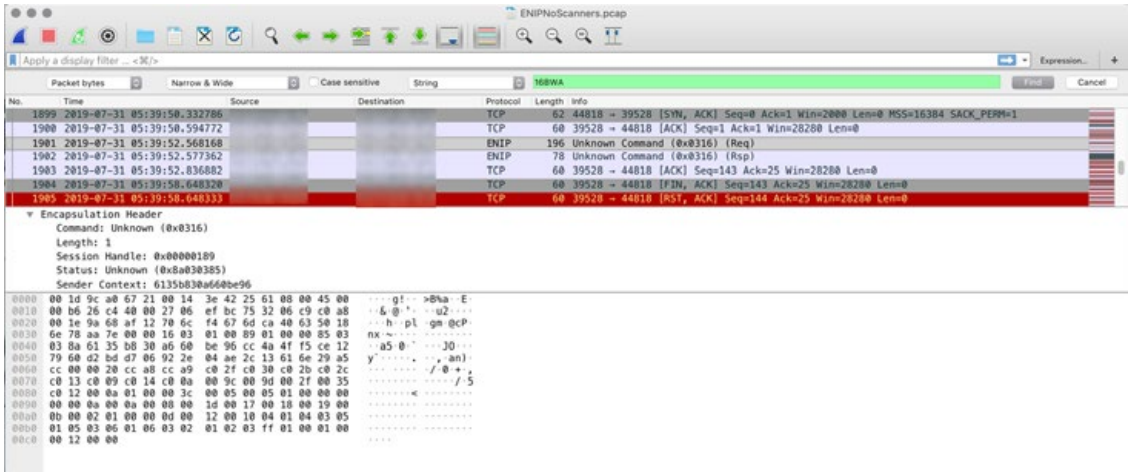


Figure 60. An unknown command sent to the Allen-Bradley MicroLogix 1100 PLC

Omron CP1L PLC

Omron communicates over a protocol called FINS, which is a UDP or Transmission Control Protocol (TCP) that operates on port 9600. In 2015, Digital Bond released an Nmap script that identifies both the UDP and TCP versions of FINS. Shortly after this, Shodan and other known scanners took this script and started scanning the internet to help identify PLCs online.

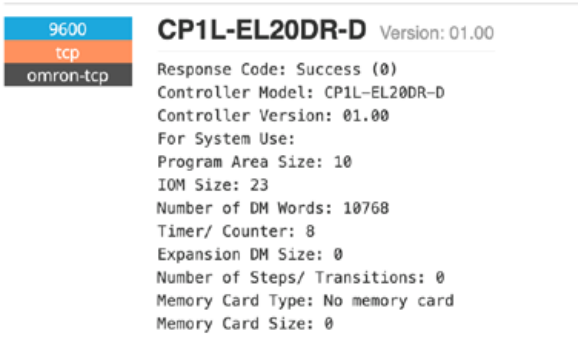


Figure 61. Information about an Omron CP1L PLC as shown on Shodan

As with the other PLCs, we took the data and filtered out all the known scanners and then manually looked at the valid FINS communications. What we saw mirrored the findings from the other PLCs. Each request matched the omron-info.nse scripts that Digital Bond released, in this case for both the TCP and UDP versions of the FINS protocol.

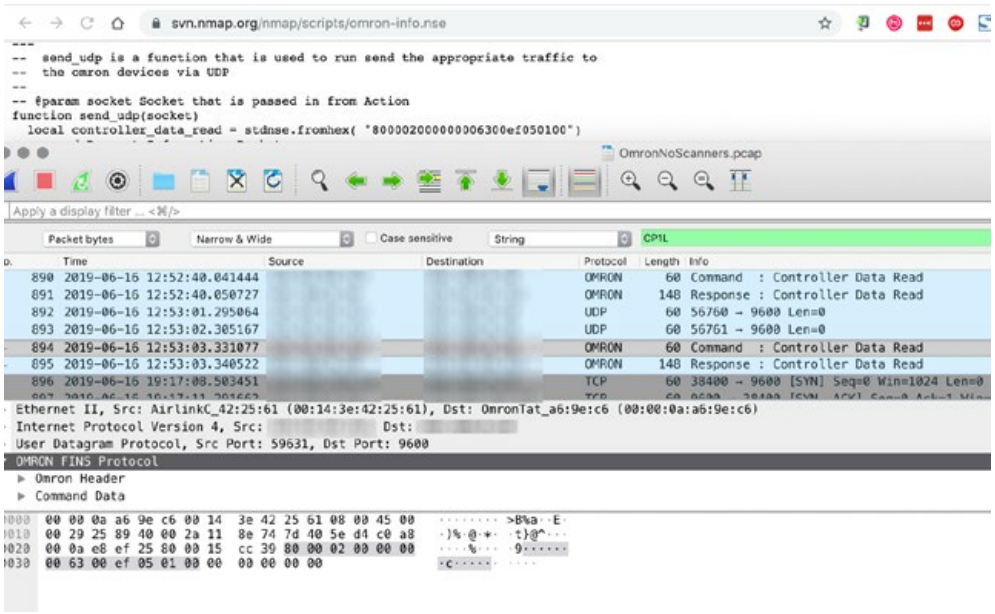


Figure 62. The UDP version of omron-info.nse

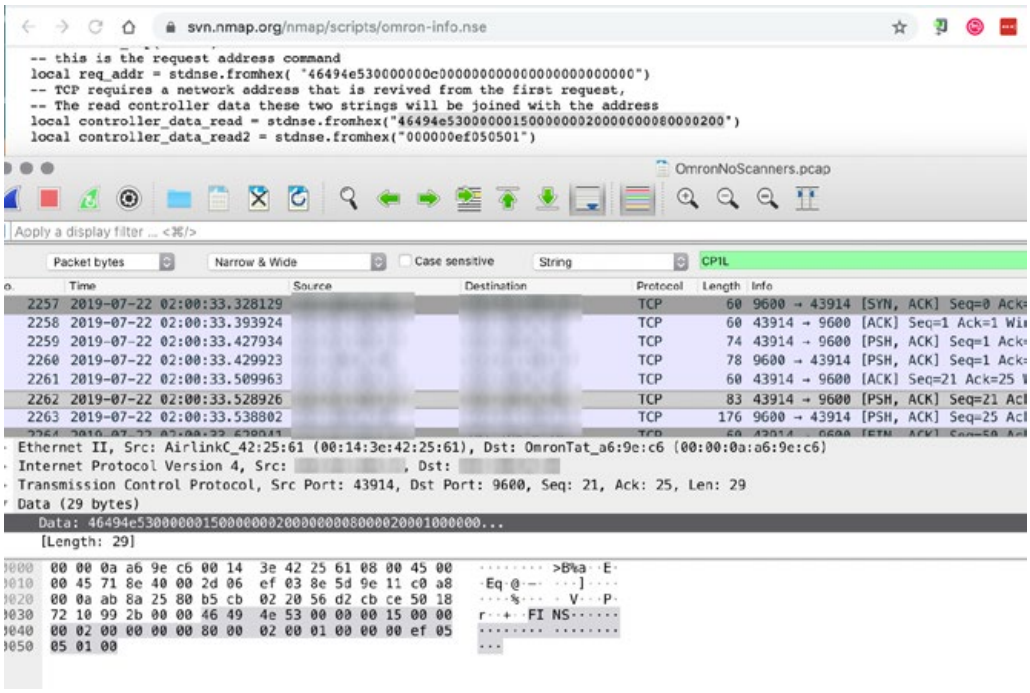


Figure 63. The TCP version of omron-info.nse

Gaining Notice From Legitimate Groups

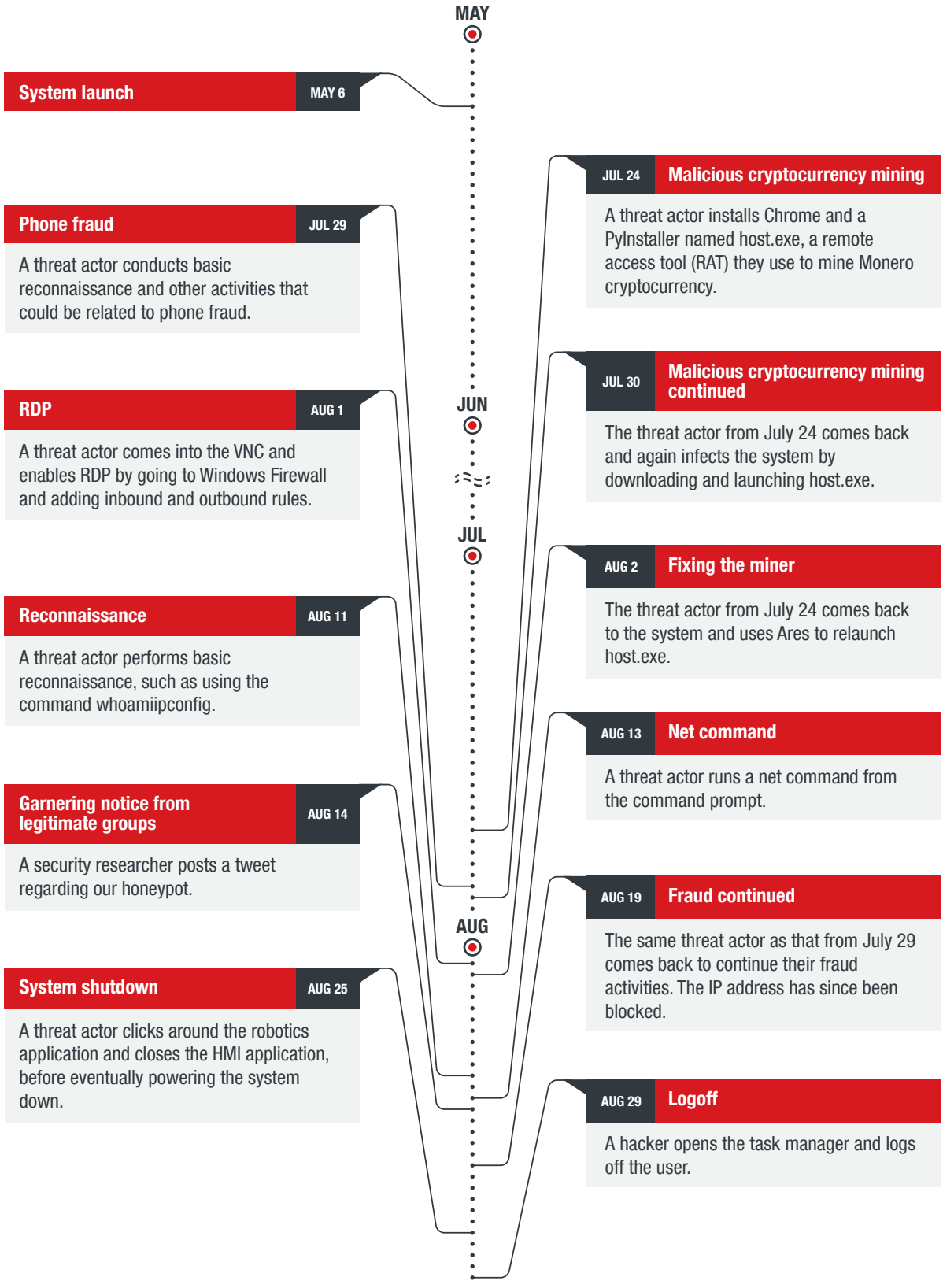
During the course of our research, a researcher named Dan Tentler (@Viss on Twitter) posted a tweet that captured our attention as it likely involved our honeypot. Tentler is a well-known researcher that has given a number of talks on finding devices on Shodan and other internet-scanning services. If he was to find our honeypot, we did not want legitimate groups to waste much time on it. This was why we had to monitor the situation and prevent further escalation.

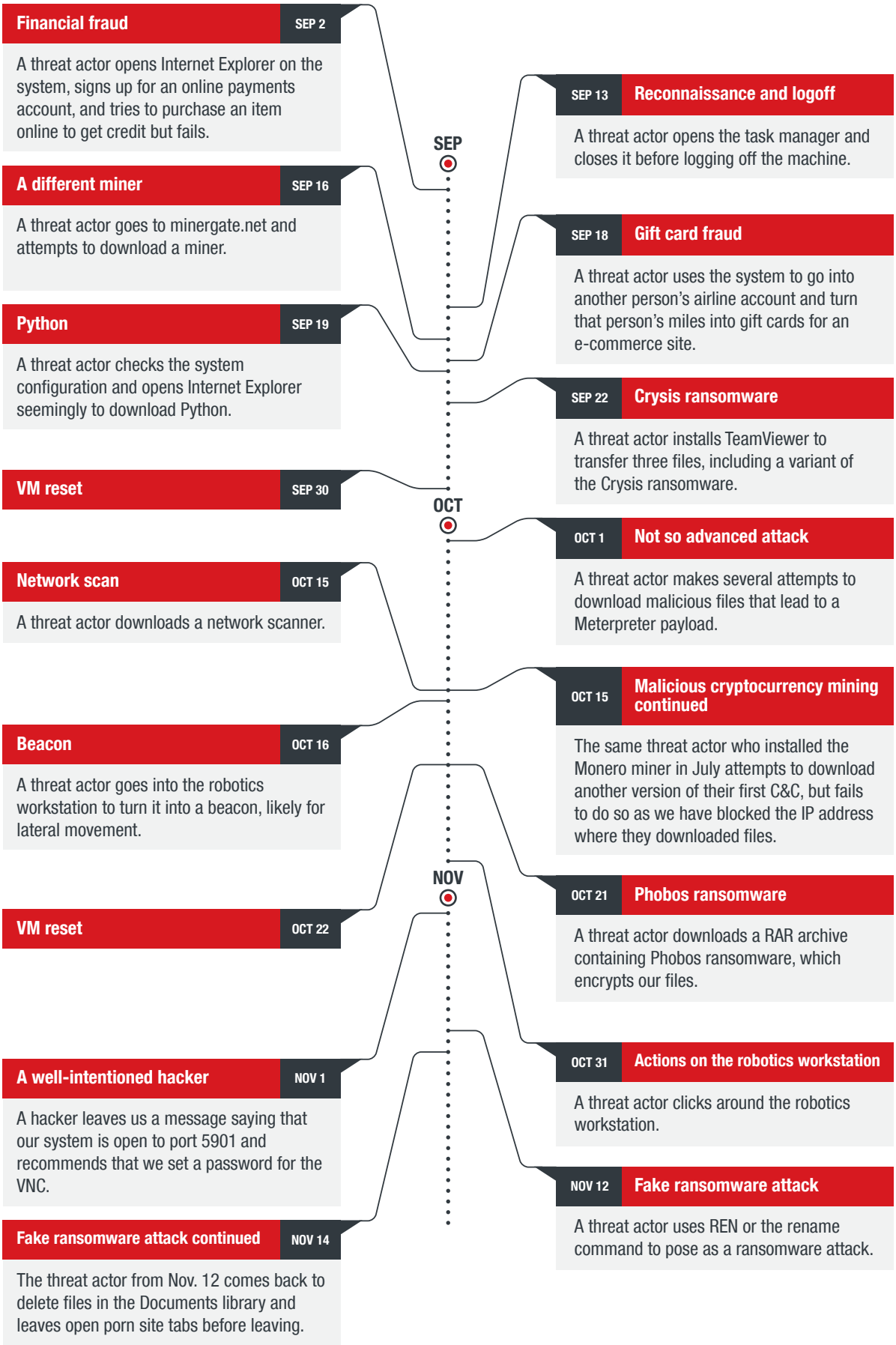
Figure 64. The tweet from a researcher about our honeypot

Within a few days, we were in contact with Tentler, who had by then escalated the issue to all of the appropriate parties. These parties included all of those who needed to be notified in the event of a control system getting exposed to the internet, meaning it also included the Industrial Control Systems Cyber Emergency Response Team (ICS-CERT). This is the right course of action should what was happening to our honeypot happen to a real system.

Honeypot Incident Timeline

We summarize the aforementioned incidents in a single timeline that spanned the period when the honeypot was online to better illustrate when attacks happened and which attacks overlapped.




Financial fraud
SEP 2
A threat actor opens Internet Explorer on the system, signs up for an online payments account, and tries to purchase an item online to get credit but fails.
A different miner
SEP 16
A threat actor goes to minergate.net and attempts to download a miner.
Python
SEP 19
A threat actor checks the system configuration and opens Internet Explorer seemingly to download Python.
VM reset
SEP 30
Network scan
OCT 15
A threat actor downloads a network scanner.
Beacon
OCT 16
A threat actor goes into the robotics workstation to turn it into a beacon, likely for lateral movement.
VM reset
OCT 22
A well-intentioned hacker
NOV 1
A hacker leaves us a message saying that our system is open to port 5901 and recommends that we set a password for the VNC.
Fake ransomware attack continued
NOV 14
The threat actor from Nov. 12 comes back to delete files in the Documents library and leaves open porn site tabs before leaving.
SEP
OCT
NOV
SEP 13
Reconnaissance and logoff
A threat actor opens the task manager and closes it before logging off the machine.
SEP 18
Gift card fraud
A threat actor uses the system to go into another person's airline account and turn that person's miles into gift cards for an e-commerce site.
SEP 22
Crysis ransomware
A threat actor installs TeamViewer to transfer three files, including a variant of the Crysis ransomware.
OCT 1
Not so advanced attack
A threat actor makes several attempts to download malicious files that lead to a Meterpreter payload.
OCT 15
Malicious cryptocurrency mining continued
The same threat actor who installed the Monero miner in July attempts to download another version of their first C&C, but fails to do so as we have blocked the IP address where they downloaded files.
OCT 21
Phobos ransomware
A threat actor downloads a RAR archive containing Phobos ransomware, which encrypts our files.
OCT 31
Actions on the robotics workstation
A threat actor clicks around the robotics workstation.
NOV 12
Fake ransomware attack
A threat actor uses REN or the rename command to pose as a ransomware attack.

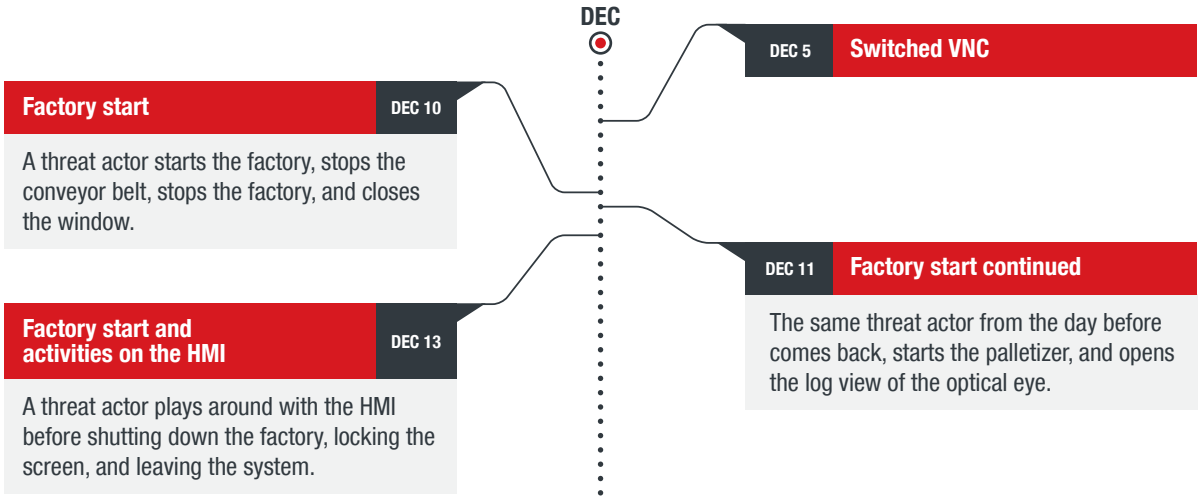


Figure 65. A timeline of incidents on our honeypot

We also summarize figures pertaining to the IP addresses that connected to our system and to communication that we detected over the period our honeypot was online, from May 6 to Dec. 31, 2019.

Unique IP addresses	
Non-scanners	8,842
Scanners	610
Total unique IP addresses	9,452

External-to-internal communication	
Packets	32,314,351
Bytes	28,229,836,479

Total communication	
Packets	565,220,996
Bytes	128,585,105,149

Scanner communication	
Packets	192,473
Bytes	222,123,953

Table 6. A summary of figures pertaining to IP addresses and communication seen on our honeypot

Conclusion

During the research period, it became apparent that there was increasing activity on the honeypot, with higher levels of interactions from day to day. For our honeypot to garner this kind of attention, we practically had to do everything wrong when it came to our faux company's general security stance. However, for many small businesses with no IT or security staff, such a situation is not uncommon.

We had VNC open and allowed no password for remote control. In the information security sector, this has long been known as a very risky configuration. Exposing any port to the internet indeed increases the risk of compromise.

In most cases, organizations should always follow the least-privilege mode. We implemented the exact opposite to lure attackers into our system. We used a common password throughout our network, but we actually saw only one potential attempt to do lateral movement. However, the longer we were exposed, the more activity we saw — and the more sophisticated attacks appeared to be compared to standard penetration-testing techniques.

While the router that we used did not support filtering to block certain items discovered during the course of our research, one feature we did not use was trust, which can be enabled on the router to allow only specific hosts to and through the device. Most industrial routers also support point-to-point virtual private networks (VPNs) to limit the exposure of remote cellular ICSs.

Our findings should serve as cautionary examples for organizations who run similar systems. We have extensively discussed the conceptualization and creation of our most realistic honeypot to date. And we have illustrated the conscious decisions and actions we took to make our system unsecure and consequently inviting for cybercriminals to target. We did all this only to a limited degree to keep our honeypot believable. This means we created openings for attacks that could realistically be found in actual smart factories.

Therefore, such attacks would not have been so successful had adequate security measures been in place to deter them in the first place. From this, organizations can take the cue to reevaluate their defenses. Organizations should ensure that their equipment and the components of their ICSs are not exposed online, as we purposely did with our various "misconfigurations." Although we did not see any attack taking advantage of how we used the same admin password for several workstations, organizations would do well to not imitate the same practice and to keep strict authentication policies to minimize the possibility of intrusions. Ultimately, weak security not only makes cyberattacks possible, but can also serve as additional invitation for attacks on industrial systems that have long stoked the interest of cybercriminals.

References

1 Danielle Veluz. (1 October 2010). *Trend Micro.* "STUXNET Malware Targets SCADA Systems." Last accessed on 6 January 2020 at https://www.trendmicro.com/vinfo/us/threat-encyclopedia/web-attack/54/stuxnet-malware-targets-scada-systems.

2 Trend Micro. (22 December 2017). *Trend Micro.* "TRITON Wielding Its Trident – New Malware Tampering with Industrial Safety Systems." Last accessed on 6 January 2020 at https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/triton-wielding-its-trident-new-malware-tampering-with-industrial-safety-systems/.

3 Andrew Krok. (21 June 2017). *CNET.* "WannaCry ransomware causes Honda plant shutdown in Japan." Last accessed on 6 January 2020 at https://www.cnet.com/roadshow/news/wannacry-ransomware-causes-honda-plant-shutdown-in-japan/.

4 Emery Dalesio. (9 August 2017). *AP News.* "Take down: Hackers looking to shut down factories for pay." Last accessed on 6 January 2020 at https://apnews.com/e316bd63f21a4fd181b3fb4a8dd7a5ba/Take-down:-Hackers-looking-to-shut-down-factories-for-pay.

5 Kyle Wilhoit. (15 March 2013). *Trend Micro Security Intelligence Blog.* "Who Is Really Attacking Your ICS Devices?" Last accessed on 6 January 2020 at https://blog.trendmicro.com/trendlabs-security-intelligence/whos-really-attacking-your-ics-devices/.

6 Kyle Wilhoit and Stephen Hilt. (5 August 2015). *Trend Micro Security News.* "The GasPot Experiment: Unexamined Perils in Using Gas-Tank-Monitoring Systems." Last accessed on 6 January 2020 at https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/the-gaspot-experiment.

7 Federico Maggi, Davide Quarta, Marcello Pogliani, Mario Polino, Andrea M. Zanchettin, and Stefano Zanero Politecnico di Milano. (3 May 2017). *Trend Micro Security News.* "Rogue Robots Testing the Limits of an Industrial Robot's Security." Last accessed on 6 January 2020 at https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/rogue-robots-testing-industrial-robot-security.

8 Moloch. (n.d.). *Moloch*. "Moloch." Last accessed on 8 January 2020 at https://molo.ch/.

9 Brendangregg. (n.d.). *GitHub*. "Chaosreader." Last accessed on 6 January 2020 at https://github.com/brendangregg/Chaosreader.

10 Jon Oberheide. (n.d.). *Jon Oberheide.* "VNC Keylogger." Last accessed on 6 January 2020 at https://jon.oberheide.org/vnclogger/.

11 Zeek. (n.d.). *Zeek.* "The Zeek Network Security Monitor." Last accessed on 6 January 2020 at https://www.zeek.org/.

12 Intel Stack. (n.d.). *Intel Stack.* "Intel Stack." Last accessed on 6 January 2020 at https://intelstack.com/.

13 Suricata. (n.d.). *Suricata*. "Suricata." Last accessed on 9 January 2020 at https://suricata-ids.org/.

14 PyInstaller. (n.d.). *PyInstaller.* "PyInstaller." Last accessed on 6 January 2020 at http://www.pyinstaller.org/.

15 Aldeid. (n.d.). *Aldeid.* "Pyinstxtractor." Last accessed on 6 January 2020 at https://www.aldeid.com/wiki/Pyinstxtractor.

16 Python Package Index. (n.d.). *Python Package Index.* "uncompyle6 3.6.2." Last accessed on 6 January 2020 at https://pypi.org/project/uncompyle6/.

17 Sweetsoftware. (8 December 2017). *GitHub.* "Ares." Last accessed on 6 January 2020 at https://github.com/sweetsoftware/Ares.

18 Kaffeine. (31 January 2018). *Proofpoint*. "Smominru Monero mining botnet making millions for operators." Last accessed on 6 January 2020 at https://www.proofpoint.com/us/threat-insight/post/smominru-monero-mining-botnet-making-millions-operators.

19 Wilbert Uy. (23 August 2019). *Trend Micro Threat Encyclopedia.* "Ransom.Win32.CRYSIS.SM." Last accessed on 8 January 2020 at https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/ransom.win32.crysis.sm.

20 Jay Garcia. (14 January 2020). *Trend Micro Threat Encyclopedia.* "HackTool.Win32.NetTool.A." Last accessed on 15 January 2020 at https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/HackTool.Win32.NetTool.A..

21 Hybrid Analysis. (n.d.). *Hybrid Analysis*. "f47e3555461472f23ab4766e4d5b6f6fd260e335a6abc31b860e569a720a5446." Last accessed on 8 January 2020 at https://www.hybrid-analysis.com/sample/f47e3555461472f23ab4766e4d5b6f6fd260e335a6abc31b860e569a720a5446?environmentId=100

22 Maureen Reyes. (12 July 2019). *Trend Micro Threat Encyclopedia.* "Ransom.Win32.PHOBOS.SM." Last accessed on 6 January 2020 at https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/ransom.win32.phobos.sm.

23 Let's Encrypt. (n.d.). *Let's Encrypt.* "Let's Encrypt." Last accessed on 6 January 2020 at https://letsencrypt.org/.

24 SCADAStrangeLove. (7 November 2012). *SCADA StrangeLove.* "PLCScan the Internet." Last accessed on 8 January 2020 at http://www.scada.sl/2012/11/plcscan.html.

25 SCADAStrangeLove. (15 October 2018). *SCADA StrangeLove.* "s7scan to replace plcscan." Last accessed on 8 January 2020 at http://www.scada.sl/2018/10/s7scan-to-replace-plcscan.html.

26 Talos Group. (28 March 2018). *Cisco.* "Vulnerability Spotlight: Multiple Vulnerabilities in Allen Bradley MicroLogix 1400 Series Devices." Last accessed on 6 January 2020 at https://blogs.cisco.com/security/talos/vulnerability-spotlight-multiple-vulnerabilities-in-allen-bradley-micrologix-1400-series-devices.

TREND MICRO™ RESEARCH

Trend Micro, a global leader in cybersecurity, helps to make the world safe for exchanging digital information.

Trend Micro Research is powered by experts who are passionate about discovering new threats, sharing key insights, and supporting efforts to stop cybercriminals. Our global team helps identify millions of threats daily, leads the industry in vulnerability disclosures, and publishes innovative research on new threat techniques. We continually work to anticipate new threats and deliver thought-provoking research.

www.trendmicro.com

TREND MICRO™ | research