515.4
Threat and Environment Manipulation

SANS
THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org

ICS515.4 ICS Active Defense and Incident Response

SANS

Threat and Environment Manipulation

The SANS ICS515 – Industrial Control System Active Defense and Incident Response course was developed by Robert M. Lee with input from a collection of experts, whose diverse work experiences, knowledge, and skills truly blended together to cover the specific content areas for this course. The author would specifically like to thank Michael Assante and Tim Conway for their continued support and contributions.

Robert M. Lee
Robert M. Lee is the CEO and Founder of the critical infrastructure cyber security company Dragos, Inc. where he and his team develop ICS cyber security products and perform remote ICS threat hunting in their Threat Operations Center. He is a SANS Certified Instructor and the course author of SANS ICS515 - "Active Defense and Incident Response" and the co-author of SANS FOR578 - "Cyber Threat Intelligence." Robert is also a nonresident National Cyber Security Fellow at New America focusing on policy issues relating to the cyber security of critical infrastructure and a PhD candidate at Kings College London. For his research and focus areas, he was named one of Passcode's Influencers, awarded EnergySec's 2015 Cyber Security Professional of the Year, and inducted into Forbe's 30 Under 30 in 2016 as one of the "brightest entrepreneurs and change agents" in technology.

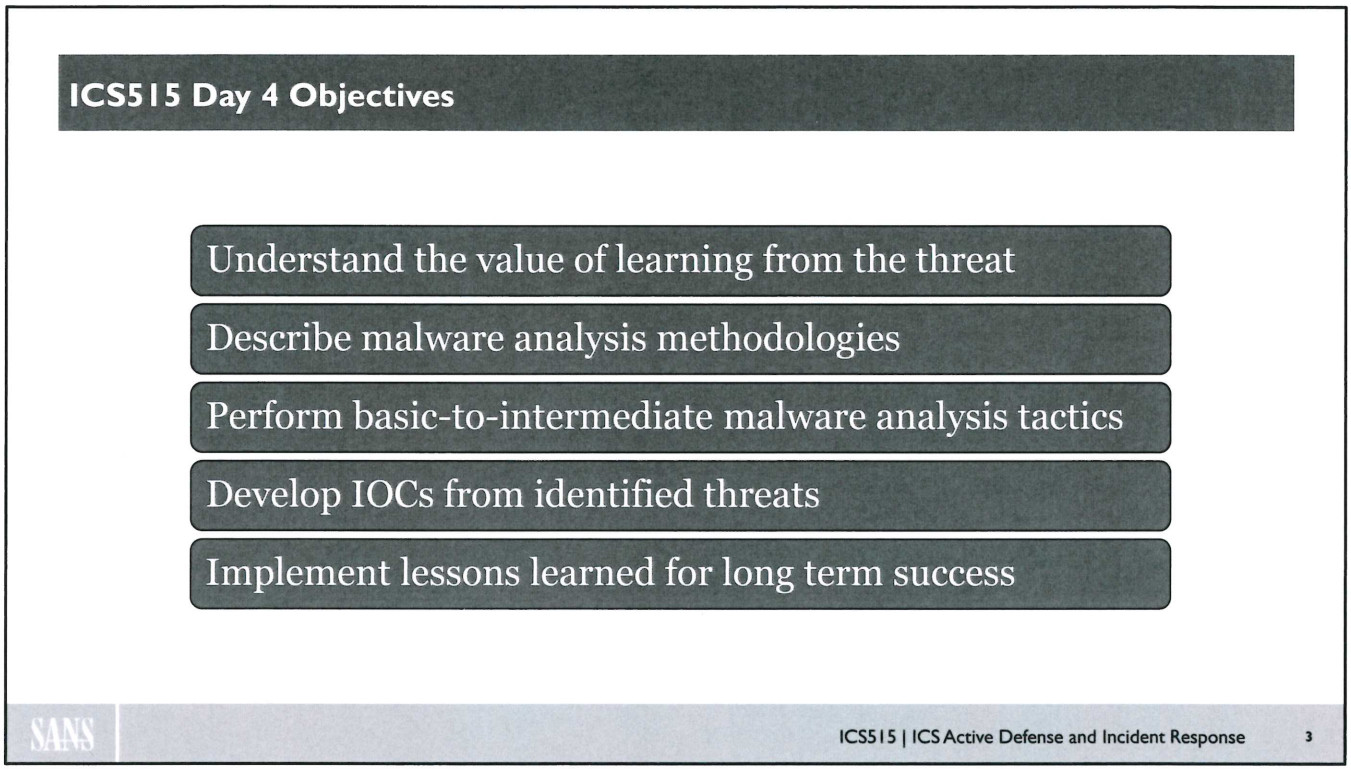
Robert obtained his start in cyber security in the U.S. Air Force where he served as a Cyber Warfare Operations Officer in the U.S. Intelligence Community. He has performed defense, intelligence, and attack missions in various government organizations including the establishment of a first-of-its-kind ICS/SCADA cyber threat intelligence and intrusion analysis mission. Robert routinely writes articles in publications such as *Control Engineering* and the Christian Science Monitor's *Passcode* and speaks at conferences around the world. Lastly, Robert, is author of the book *SCADA and Me* and the weekly web-comic http://www.LittleBobbyComic.com

Robert may be found on Twitter @RobertMLee or contacted via email at: RLee@Dragos.com

Contributions By:

Michael Assante
Michael Assante is currently the SANS project lead for Industrial Control System (ICS) and Supervisory Control and Data Acquisition (SCADA) security. He served as vice president and chief security officer of the North American Electric Reliability Corporation (NERC), where he oversaw industry-wide implementation of cyber security standards across the continent. Prior to joining NERC, Michael held a number of high-level positions at Idaho National Labs and he served as vice president and chief security officer for American Electric Power. Michael's work in ICS security has been widely recognized. He was selected by his peers as the winner of *Information Security Magazine*'s security leadership award for his efforts as a strategic thinker. The RSA 2005 Conference awarded him its outstanding achievement award in the practice of security within an organization. He has testified before the U.S. Senate and House and was an initial member of the Commission on Cyber Security for the 44th Presidency. Prior to his career in security, he served in various naval intelligence and information warfare roles. He developed and gave presentations on the latest technology and security threats to the chairman of the Joint Chiefs of Staff, director of the National Security Agency, and other leading government officials. In 1997, he was honored as a Naval Intelligence Officer of the Year.

Tim Conway
Tim Conway is currently the technical director of ICS and SCADA programs at SANS. He was responsible for developing, reviewing, and implementing technical components of the SANS ICS and SCADA product offerings. He was formerly the director of CIP Compliance and Operations Technology at Northern Indiana Public Service Company (NIPSCO). He was responsible for Operations Technology, NERC CIP Compliance, and the NERC training environments for the operations departments within NIPSCO Electric. Throughout his career Tim was previously an EMS computer systems engineer at NIPSCO for eight years, with responsibility over the control system servers and the supporting network infrastructure. Tim previously served as the chair of the RFC CIPC, is the current chair of the NERC CIP Interpretation Drafting Team, a current member of the NESCO advisory board, the current chair of the NERC CIPC GridEx 2013 Working Group, and current chair of the NBISE Smart Grid Cyber Security panel.



ICS515 Day 4 Objectives
Here are the five up-front objectives for today's class:

1. Understanding the value and importance of learning from the threat by safely interacting with it is the cornerstone of making an adversary's best capability your best source of defense information.
2. The ability to describe malware analysis methodologies can help you at a high level determine the type of malware analysis that should be performed in various situations.
3. The ability to perform some of those analysis methodologies is going to make you a strong contributor to the team; it is the goal to impart basic-to-intermediate tactics, as this is understandably just a one-day section.
4. From these skills, though, you can develop IOCs from identified threats so that you can implement the lessons learned toward actionable recommendations. These recommendations can help guide defense personnel and help influence architecture changes to neutralize or delay a threat.
5. You should leave today understanding the importance of interacting with threats and what that can bring. It is not expected that you will become a malware analyst. However, what is expected is that you'll know where some of the IOCs come from that you'll undoubtedly use and why they are of value (as well as how to spot bad ones and ask the right questions). There's a lot of threats out there and being a malware analyst as a full-time job or having deep experience on it is too much for one class, let alone one day. The intent is not to make you an expert in assembly and disassembly; the intent is to help you quickly identify information useful to feeding the ACDC cycle actionable threat data that can be used internally or externally for threat intelligence.

Today, we focus on some of the common threats observed in ICS environments.

Introduce the methodologies that go into interacting with threats
The operating assumption:

- Security personnel use IOCs and thus need to know where they come from
- Use automated malware analysis through sandboxes to get quick and actionable information useful to feed the ACDC process, especially during an incident
- To perform meaningful automated malware analysis, you need to understand the "behind the scenes"

Therefore:

- For malware analysis, we introduce and highlight methodologies
- We get hands on with some basic-to-intermediate techniques
- We employ these techniques or establish an automated malware analysis capability

Takeaways for Malware Analysis

It is a bit foreboding anytime a class basically has a warning page; however, this page is meant to help guide the discussions instead of serving as a warning. This subject matter is fairly technical, and there will be a lot of things to ask questions about that are not in the slides. For example, threads and processes are not covered fully in depth because you need to know only that they exist for this class. But if you have questions, ask as it'll contribute to everyone's learning. This class day is likely the most interactive because of how technical the material is.

Most ICS organizations are not going to have dedicated Threat and Environment Manipulation (TEM) personnel performing malware analysis. And we'll talk about some available automated malware analysis capabilities to help supplement that fact. However, understanding the science behind these capabilities and performing some of them yourself, especially when other capabilities fail, can carry you far.

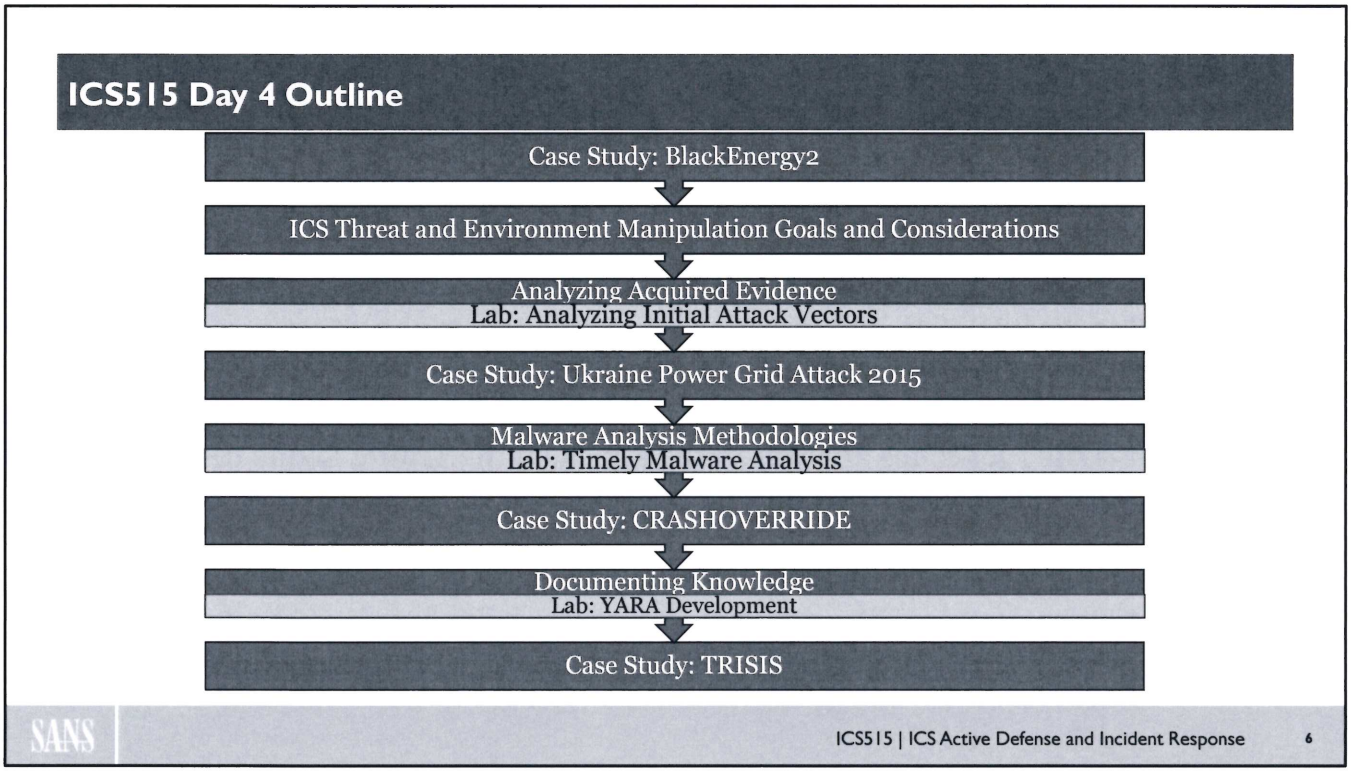
Therefore, remember that the focus is on understanding malware analysis, getting some hands-on experience, and using this information in your organization directly or indirectly.

Course Scenario Goals

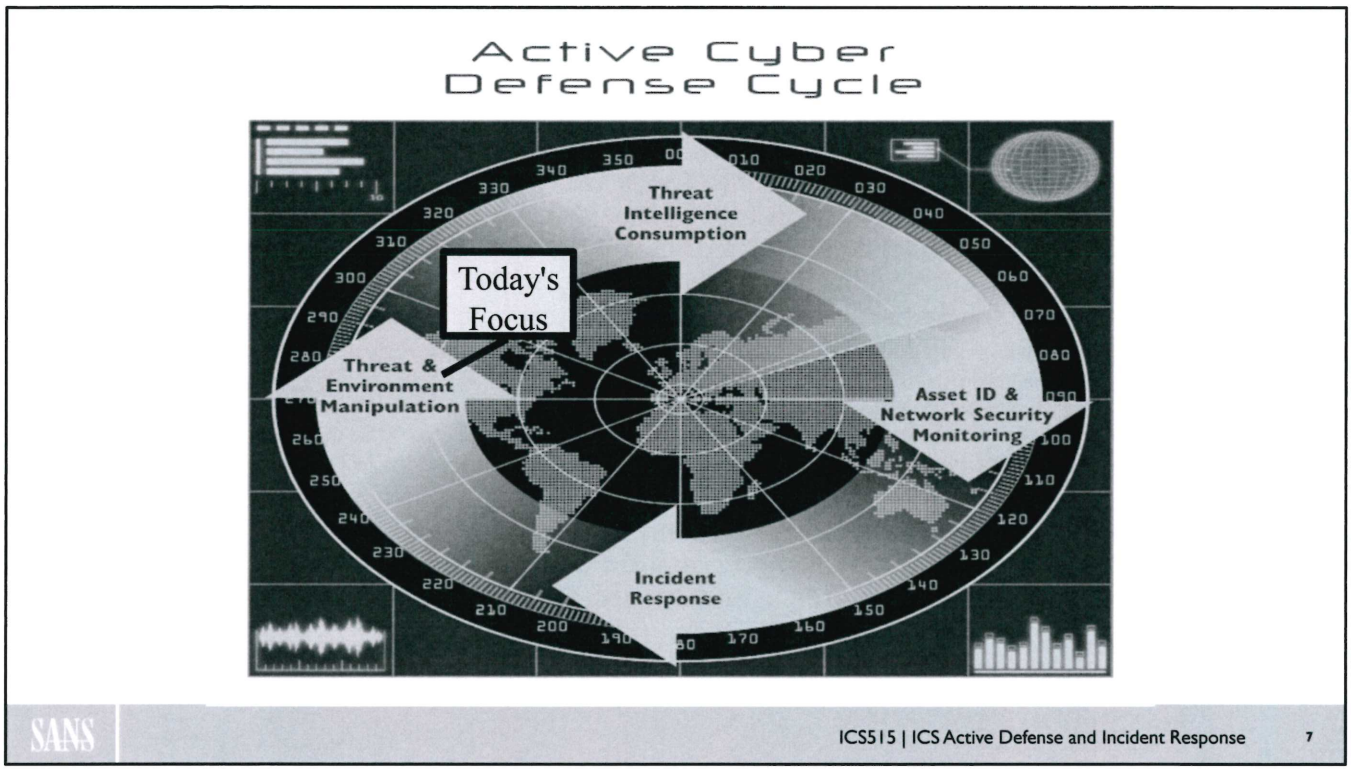
- Water Plant:
  - ~~Make a topology map of the water utility's plant network.~~
  - ~~What is the root cause analysis of the attack on the water plant?~~
  - What is the intrusion and infection vector that led to the attack?
  - ~~Are there any other intrusions and~~ are they related to the attack?
- Traffic Lights:
  - What is the root cause analysis of the attack on the Traffic Lights network?
  - Are there any other intrusions and are they related to the attack?
- General:
  - ~~What activity groups could reasonably target the ICS represented in the city?~~
  - ~~Would it be useful to share indicators from one attack to the other?~~
  - What lessons learned would be useful to share between the organizations?

Course Scenario Goals

You have specific goals for the scenario. You will make a network map of the water plant and you will investigate any incidents and intrusions. You will get walked through the traffic light scenario and will have goals related to labs you will work in the future, such as the root cause analysis of the Traffic Lights network attack. In general, you'll also try to determine if the incidents between the Traffic Lights network and CWU are related.



ICS515 Day 4 Outline
This page intentionally left blank.



Active Cyber Defense Cycle: Day 4

Today's focus is on threat and environment manipulation. This phase of the Active Cyber Defense Cycle enables analysts to safely manipulate a threat. The word manipulation is chosen instead of reverse-engineering malware or simply malware analysis for two main reasons. First, malware is an adversary capability that is not meant to be run in an analyst's VM for analysis and threat data collection; the adversary doesn't like that. You are manipulating the threat and it's an offensive-minded kind of defense approach. You are trying to get at the information the malware has in any way you can that is legal and internal to your environment. Second, threats are not always malware-based. Our first case study actually shows physical damage to control systems that was not malware-focused. ICS specifically understands that threats can be a guy opening up sewage lines with unauthorized access to the facility outside in the parking lot. But he's still a threat. Through manipulating threats to learn from them, you can also determine architectural changes (logical or physical) that would neutralize the threats or otherwise delay them. For example, a piece of malware that tries to connect to a specific command and control server can be impacted by environment changes that tell the systems that any requests to Malicious Command and Control Server goes to an analysis machine for the TEM personnel to look at. It's now neutralized and information is gathered. This phase of ACDC puts the focus on interacting directly with the threat inside of your own network to counter its intentions.

General Malware Analysis in ICS Environments

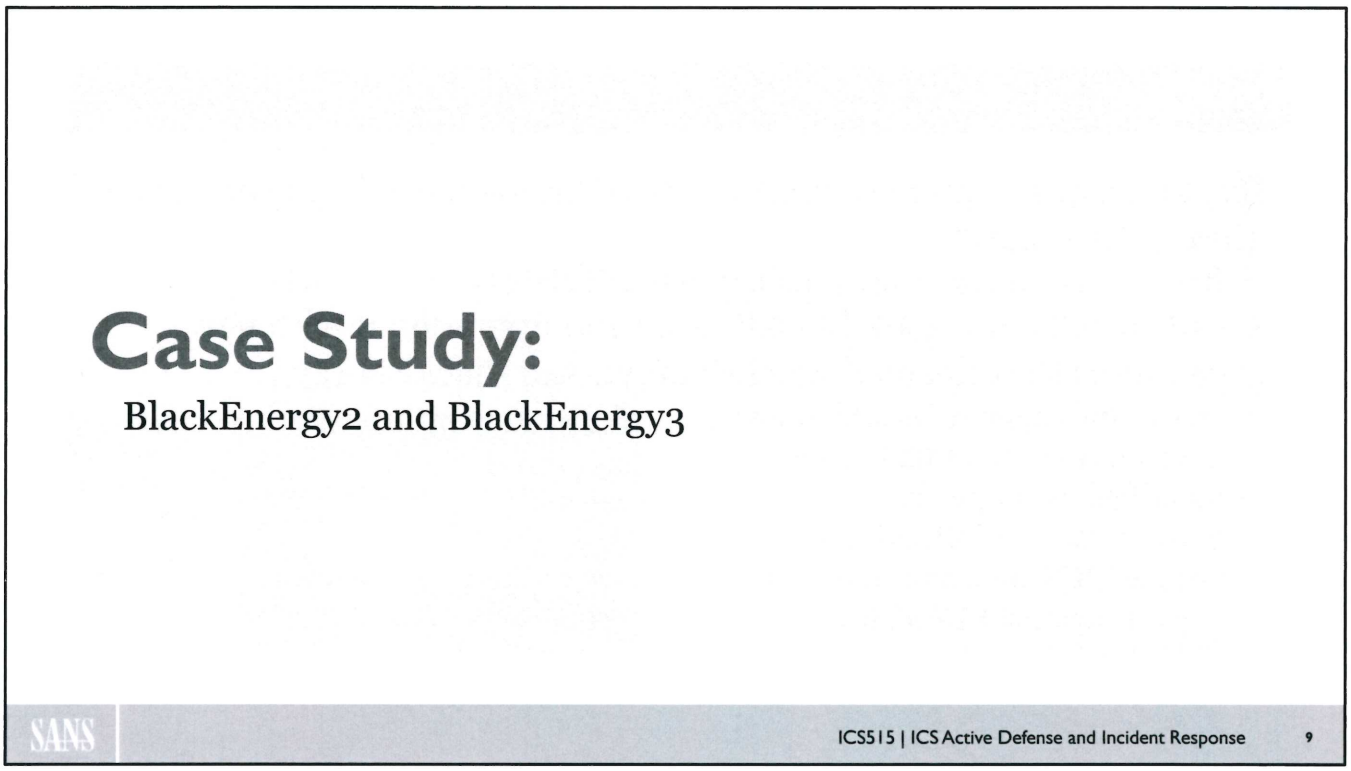
1. Establish a safe working environment (for example, use VMs)
2. Document received evidence files (such as phishing email, pcap, memory files, and so on)
3. Perform initial analysis to identify useful information (for example, pre-existing YARA rules)
4. Identify and investigate interesting files, executables, and such
5. Extract suspect file(s) from the evidence and document them, including hashes
6. Run copies of the suspect files through in-house antivirus systems to identify matches:
   a. If the suspect files have known AV signatures, it is known malware, and online research can take place, such as looking up digital hashes, antivirus company threat reports, and more
   b. If the suspect file does not have a known AV signature, continue on (or, if at least some do not)
7. Begin processing the suspect file(s) through an in-house automated malware analysis sandbox if possible
8. Begin extracting information using behavioral malware analysis techniques
9. Use the analysis to create IOCs and provide context while documenting findings
10. Pass IOCs, documentation, and so on to those managing the internal threat intelligence products

General Malware Analysis in ICS Environments

Most threats involve malware, and today's focus is heavily centered around learning from malware. Even threats that don't rely on malware to impact the ICS may rely on malware to gain a foothold in the ICS or corporate network. The same thought processes and type of analysis that goes into malware analysis is useful to understanding nonmalware-based threats.

To help guide the day, the 10-step process is used throughout the day. This is a general methodology and may not represent the abilities and processes of every ICS security team, but it covers the vast majority of cases.

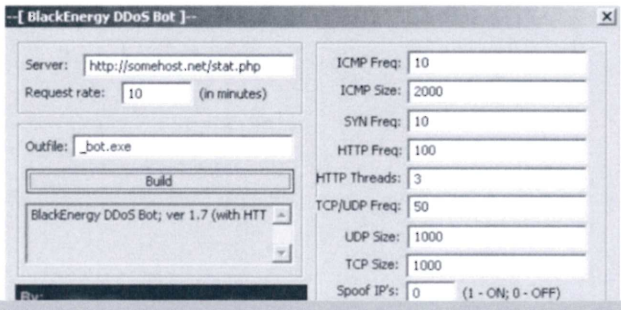
Throughout the 10-step process, analysts should think about what can be done to the architecture to deny or delay the adversary and its capabilities.



This page intentionally left blank.

BlackEnergy

- BlackEnergy is a piece of malware that has been used by cybercrime groups since 2007
  - BlackEnergy was also used in distributed denial of service attacks
- An advanced actor took BlackEnergy and upgraded it with new capabilities thus deemed BlackEnergy2 and BlackEnergy3
  - One of the capabilities added was a zero-day exploit that became identified as sandworm
  - Some versions of BlackEnergy2 targeted ICS environments with exploits against SIMATIC, CIMPLICITY, and Advantech



BlackEnergy

BlackEnergy2 (BE2) and BlackEnergy3 (BE3) represent an interesting case study. It was a piece of traditional cybercrime type malware even used in distributed denial of service (DDoS) attacks by those groups. However, it was wrapped up into a toolset that became identified as BlackEnergy2 by an advanced persistent threat (APT). There it was made more complex and was leveraged in espionage campaigns targeting various groups, including NATO and defense contractors.

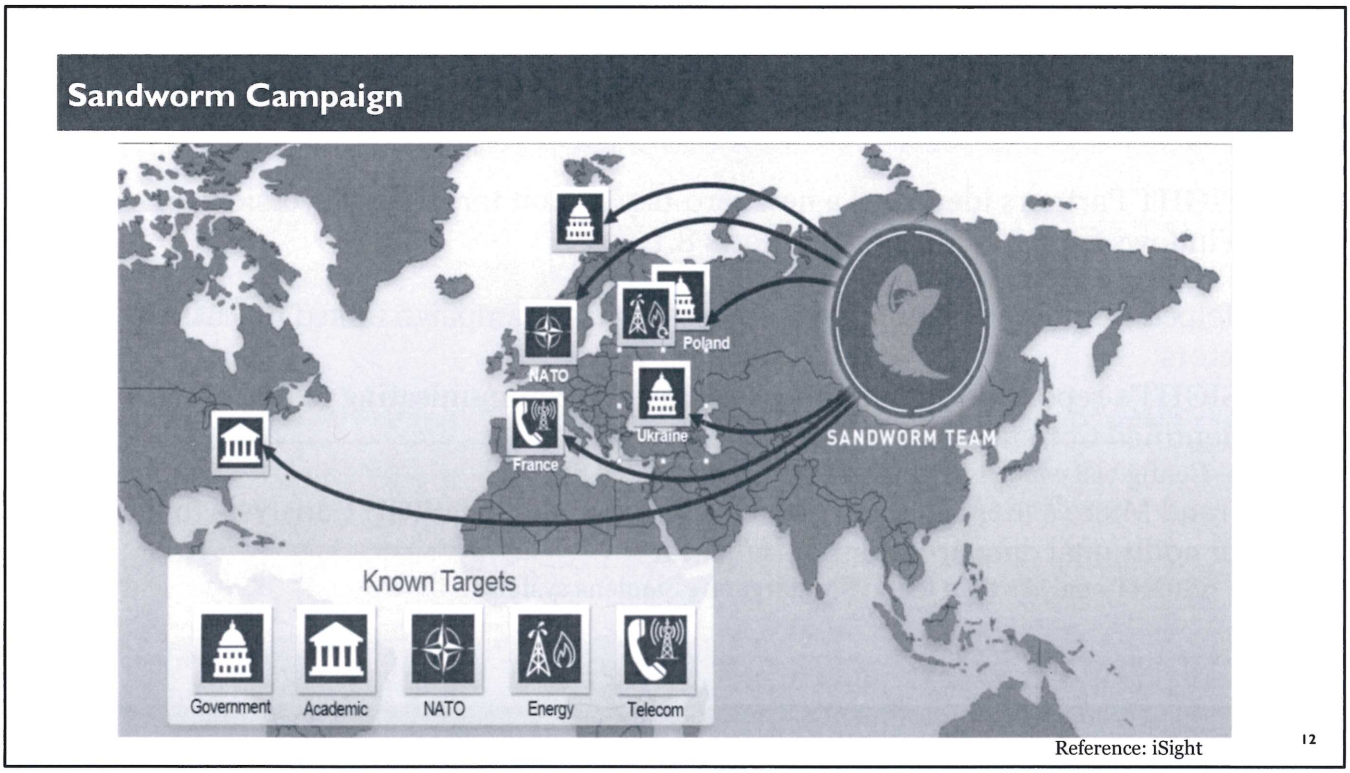
Sandworm Exploit

- iSIGHT Partners identified a new zero-day exploit targeting all versions of Windows from VistaSP2 to Windows 8.1
  - The zero-day took advantage of OLE
- Helped the company identify a cyber-espionage campaign linked to Russian actors
- iSIGHT's report led Trend Micro to find a file communicating to one of the identified Command and Control (C2) servers
  - Config.bak which is a CIMPLICITY Server related file
- Trend Micro's identification of the ICS related file led iSIGHT analysts to look for additional data around that indicator
  - iSIGHT analysts then found files targeting Siemens systems

Sandworm Exploit

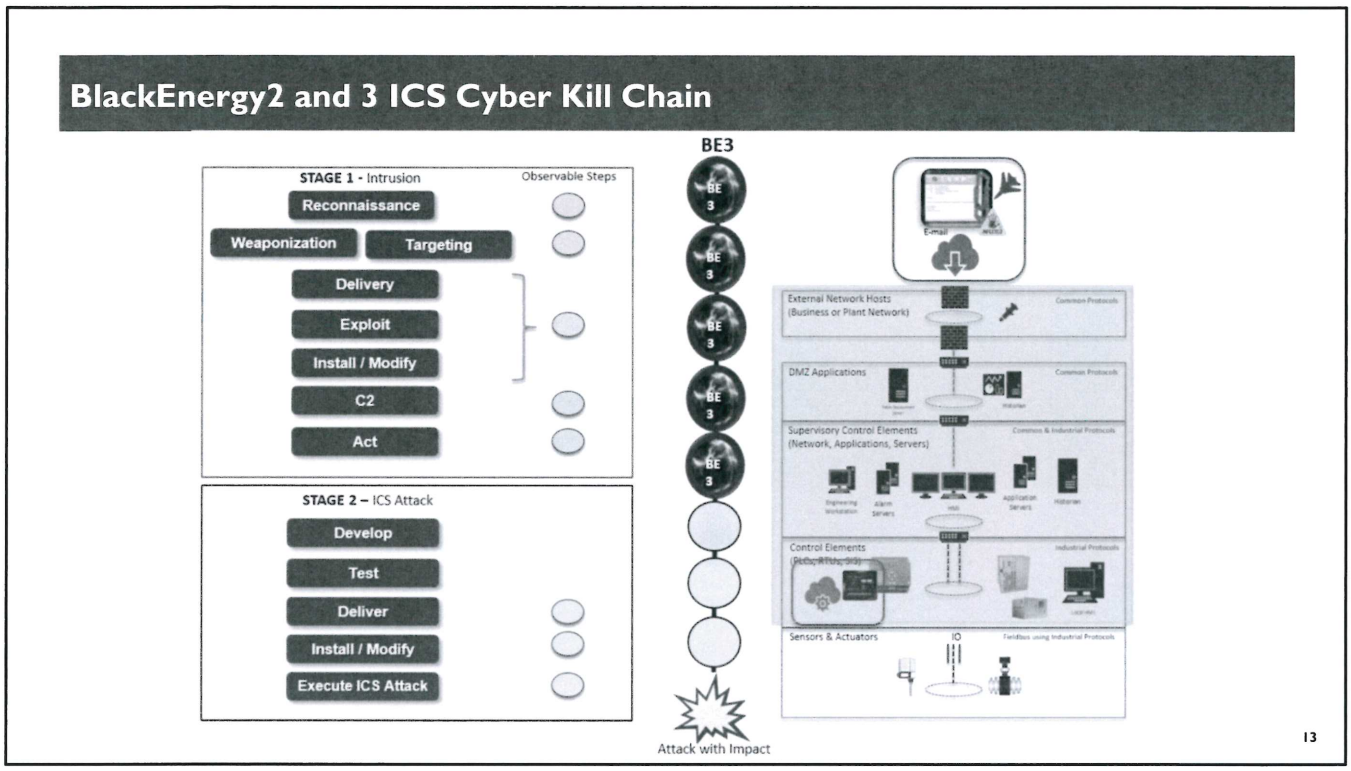
The Sandworm Exploit report by iSIGHT Partners (https://www.isightpartners.com/2014/10/cve-2014-4114/) led to the identification of multiple C2 servers. Trend Micro analysts pivoted off of the C2 servers to find a file that had been seen communicating to one of the IP addresses. The file was identified as config.bak, which is a type of file that goes with GE's CIMPLICITY SCADA software. From this the analysts were able to identify other files related to the config.bak file, including files that were downloaded from the C2 server. These other files were additional type of CIMPLICITY-related malicious files as well as non-CIMPLICITY related malware. The other files were useful in identifying email addresses, follow-on files, C2 servers, and targets.

The work by the Trend Micro analysts off of the iSIGHT report allowed the iSIGHT analysts to revisit the data in a follow-on report (http://www.isightpartners.com/2014/10/sandworm-team-targeting-scada-systems/) and identify additional files such as CCProjectMgr.exe that were being used—which are related to SIMATIC and Siemens WinCC software. This helped developed new indicators useful for analysts to identify infected systems.



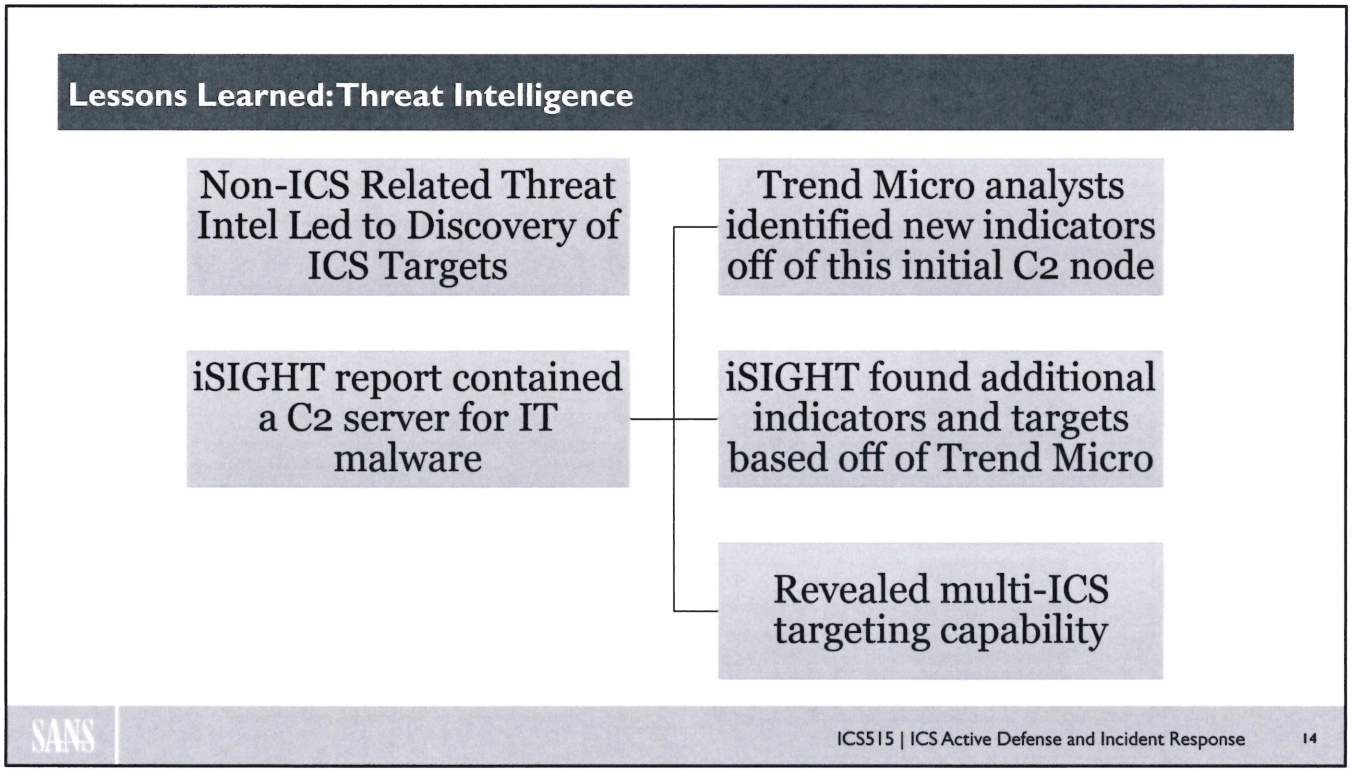
Sandworm Campaign

The team known as Sandworm had a coordinated campaign across multiple years targeting Russian-based interests such as Ukraine, Poland, NATO, and organizations linked to analysis and preparation of military forces. This victim analysis shows that the targets were not of interest for cyber-crime-type-based groups but more aligned with national interests.



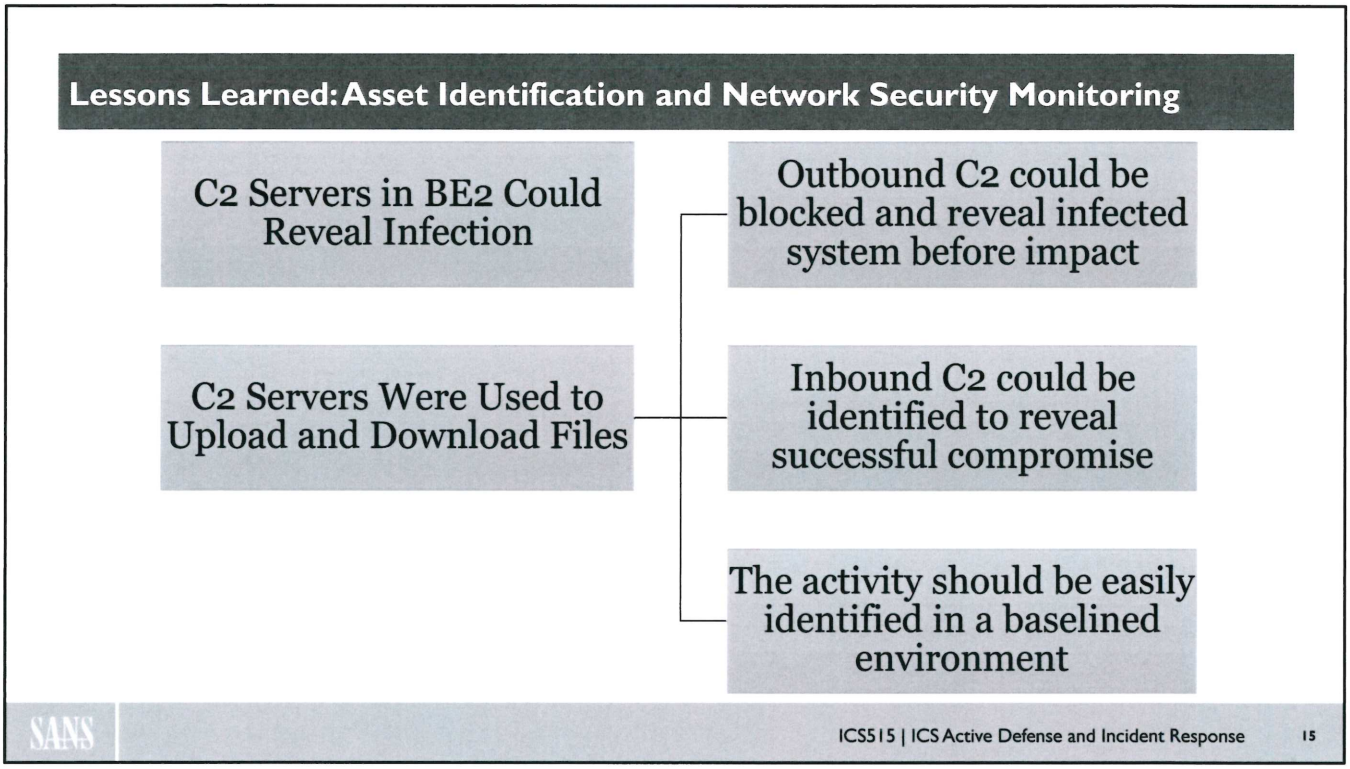
BlackEnergy2 and 3 ICS Cyber Kill Chain

Here we see that in both BE2 and BE3, there was never an attack capability. It was simply an espionage toolkit. This led to a lot of early confusion around the Ukraine Cyber Attack. News reports indicated that "BlackEnergy caused the power outage" which is not true. It enabled the adversaries to get access to the environment but the malware did not cause the power to go out. BE2 and BE3 kill chain analysis prior to the incident could have helped alleviate the confusion. However, this was used as the Stage 1 of the 2-stage attack.

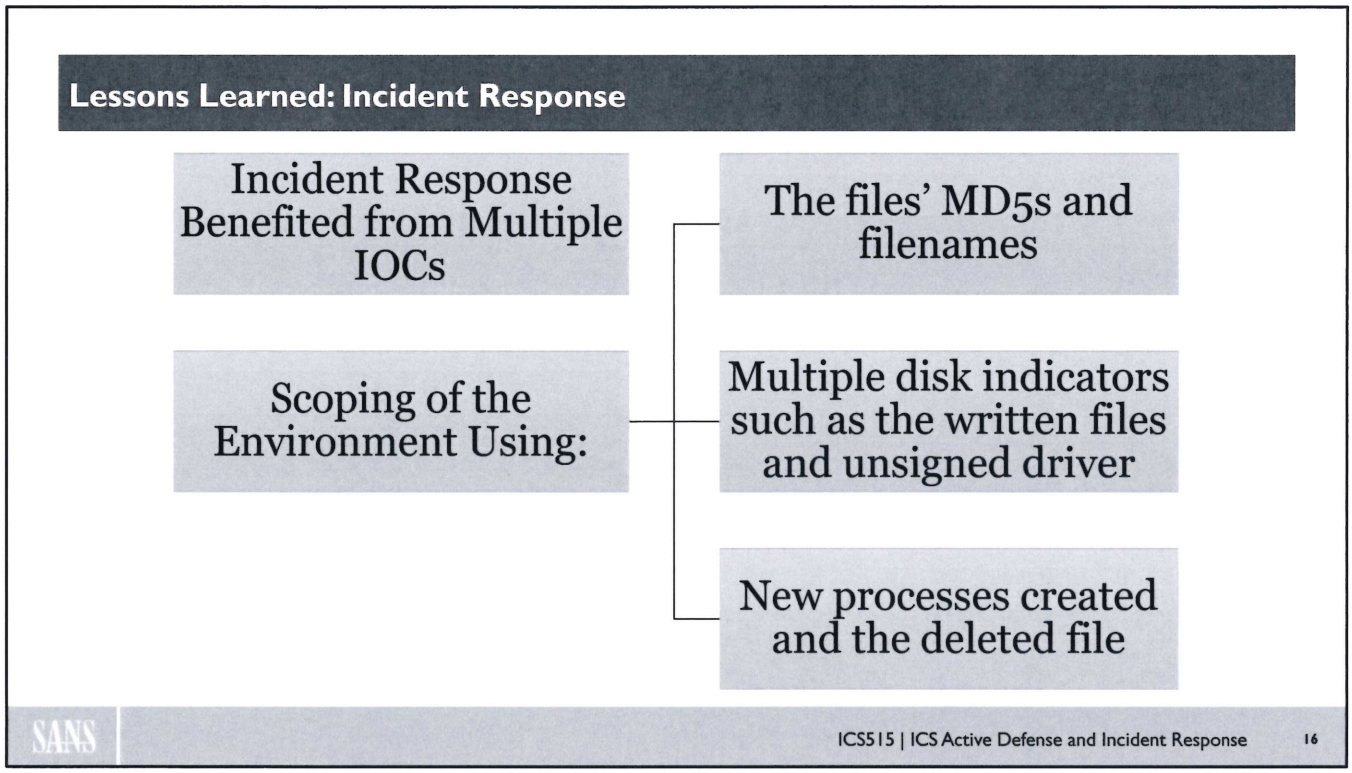


Lessons Learned: Threat Intelligence

BlackEnergy2 was originally targeted only toward IT networks for cyber crime purposes. The iSIGHT identification of the zero-day exploit in BlackEnergy2 led to the identification by them of multiple Command and Control (C2) servers. This threat intelligence helped the Trend Micro analysts uncover files related to ICS that were communicating with that C2 server. This then led iSIGHT to identify a different type of ICS being targeted and additional indicators. These indicators would be exceptionally useful for incident response and network security monitoring analysts.

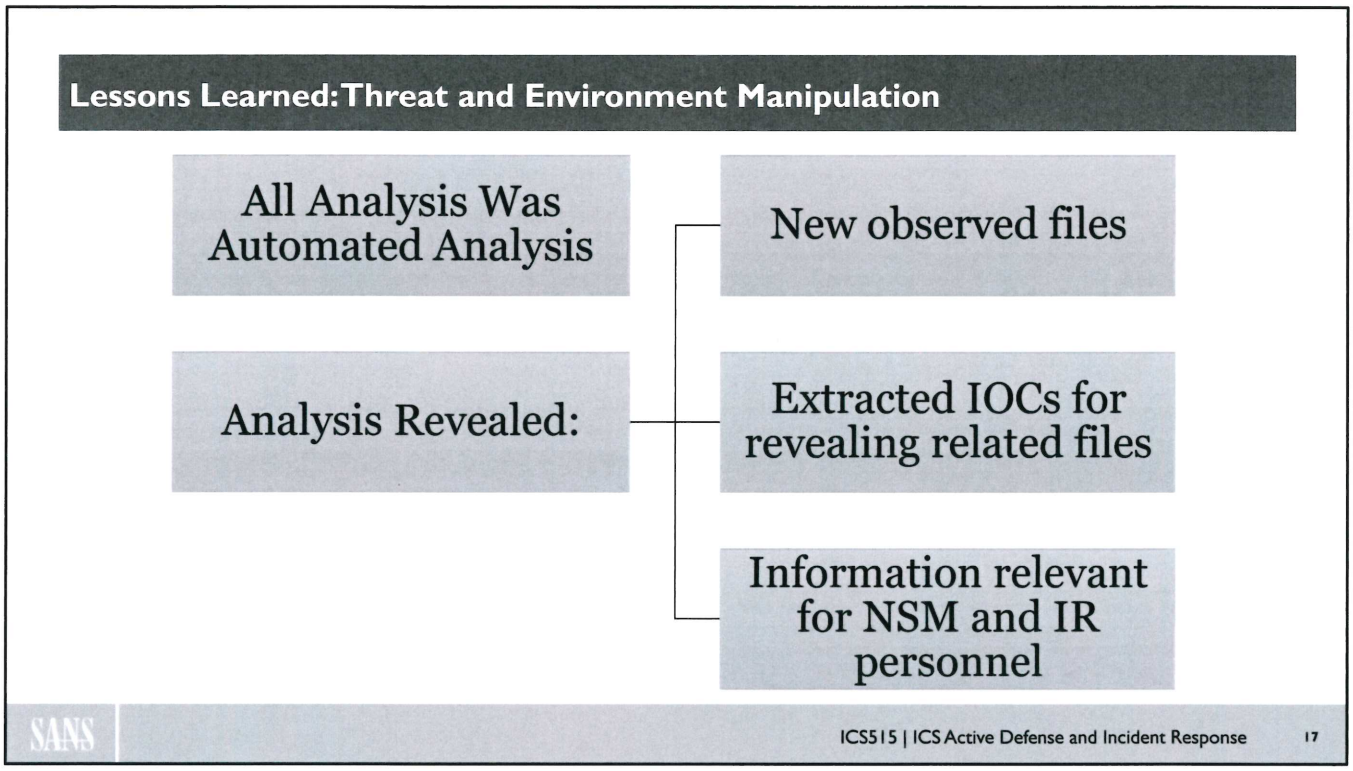


Lessons Learned: Asset Identification and Network Security Monitoring
The C2 servers that BE2 used were leveraged by the adversary to upload and download new files. This was the ability of the malware to steal data from the environment as well as push new modules and capabilities to the malware. The C2 servers would look very foreign to an ICS network where the SCADA servers were connected. Additionally, this activity should have shown defenders that Internet-connected SCADA servers would be compromised.



Lessons Learned: Incident Response

From an incident response perspective, the IOCs could have quickly led to the identification of infected systems as well as the collection of new variants of the malware. The files' MD5s, the unsigned driver, the artifact of the deleted file, and the created processes are all indicators that would make high-confidence IOCs.



Lessons Learned: Threat and Environment Manipulation

The iSIGHT analysts' analysis led to the identification of the zero-day exploit and C2 servers. This led the Trend Micro analysts to identify the ICS-related malware, which led the iSIGHT analysts to find an additional and different ICS software being targeted. This all created new indicators that could be used for pivoting to identify new C2 servers and malware variants as was done in this case study. This level of activity is something that ICS security personnel should be able to do even without in-depth malware analysis skills. Malware analysis skills help, but automated analysis is a timely and achievable tactic. The TEM would have led to better NSM and IR at any given ICS organization that was impacted with BlackEnergy2.

Identified ICS Threat Tactics: Nonmalware

- Radio frequency manipulation to impact control systems:
  - For example, Polish teen manipulated tram system in the city of Lodz with a modified TV remote control
- Direct interaction with the control systems after initial access:
  - For example, German steelworks case study
  - For example, Maroochy Water Services compromise of the sewage management system
- Takeaways:
  - Manipulation of the control systems through unauthorized access is possible using the systems and technologies in place and can be as impactful if not more so than malware
  - Identifying these types of threats require network monitoring, and interacting with this threat in a timely manner is unlikely

Identified ICS Threat Tactics: Nonmalware
Here's a quick overview of two case studies where malware wasn't responsible for damaging the process or physically damaging the systems.

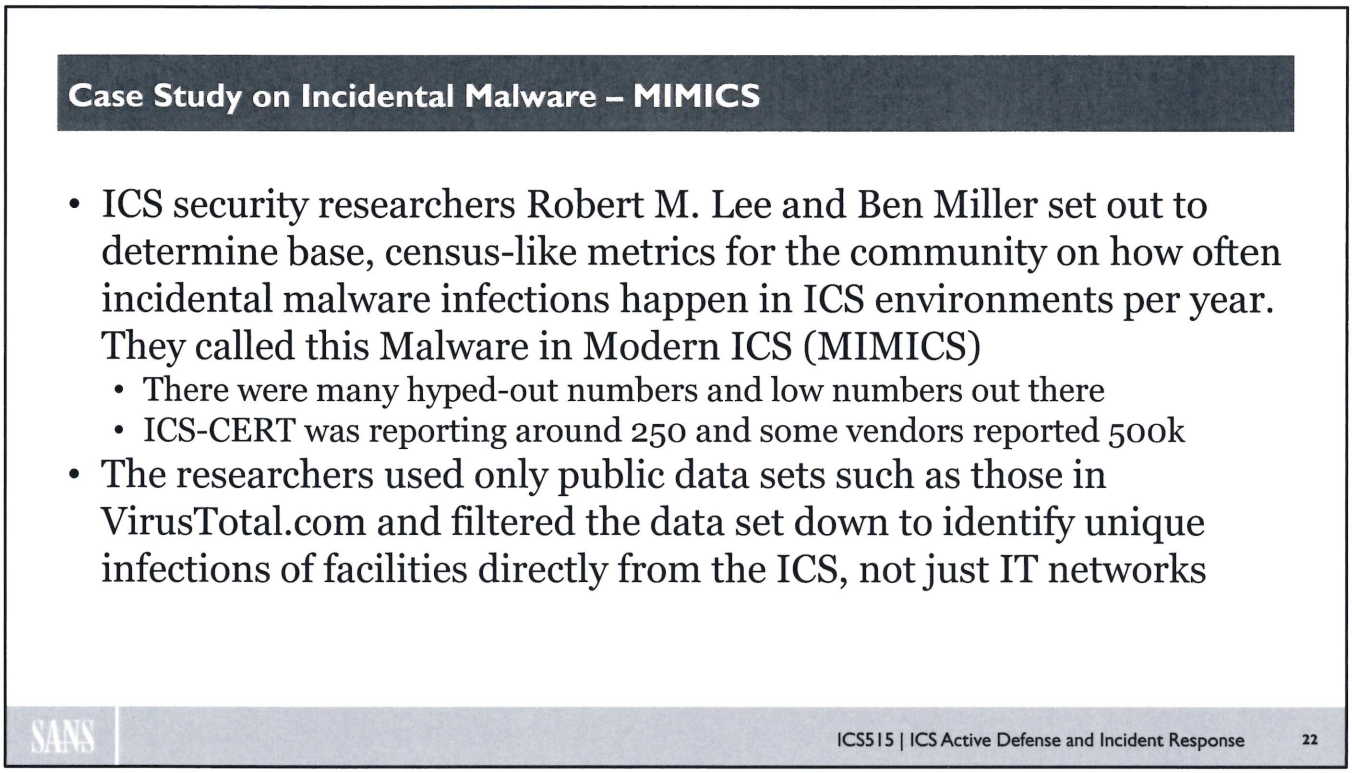
References:
http://www.theregister.co.uk/2008/01/11/tram_hack/
http://www.computerworld.com/article/2561484/security0/utility-hack-led-to-security-overhaul.html

Identified ICS Threat Tactics: Traditional IT Malware

- Traditional IT malware used as a first stage access to ICS business connected networks:
  - For example, malware such as zero-access botnet has been previously observed as a first-stage access to a network
- Traditional IT malware impacting control system environments:
  - For example, Slammer worm crashed Davis-Besse Nuclear Power Station HMI and bogged down SCADA network
  - For example, Conficker accidentally brought into control system network via USB halted steel facility network in Brazil

Identified ICS Threat Tactics: Traditional IT Malware
In these case studies, malware was used to affect the process or system. But none of the pieces of malware were targeted toward the ICS. They may have been intentionally placed and abused (like using zero-access as a first-stage access into the network by an adversary) but none reveal specific ICS intent. Other nontargeted malware can be incredibly impactful, such as Conficker.

Reference: http://www.networkworld.com/article/2217684/data-center/attacks-on-power-systems--hackers--malware.html



Case Study on Incidental Malware – MIMICS

In 2016, researchers Robert M. Lee and Ben Miller saw a number of low and hyped-out numbers related to ICS incidents. ICS-CERT was reporting around 200 incidents of incidental malware infections and Dell SecureWorks was reporting 500,000 a year. The two generated four hypotheses and set out to hunt for some base metrics that the community could use.

The hypotheses were:

1. Lots of infected ICS software exists
2. Public reports contribute to discoveries internal to environments
3. ICS-themed malware is not uncommon
4. Untrained IT security teams submit sensitive ICS files to the Internet

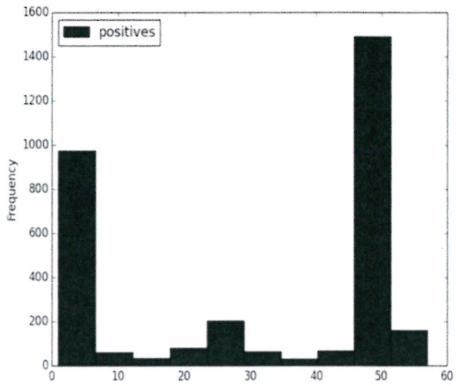
To test these hypotheses, the researchers only leveraged public data sets such as VirusTotal.com and Malwr.com to see submissions for ICS related files. They filtered through to ensure the ICS software was installed correctly on systems infected, helping to ensure it was legitimate and not just honeypots or randomly infected files. They also weeded through the data to ensure that a single infection wasn't seen as one incident because a single site might have hundreds of infections.

Prior to the research, the community did not have a base set of metrics of the realistic numbers of incidental infections and, prior to the research, there were less than five cases of ICS-themed malware (not ICS-tailored) cases.

The testing also only included known viruses to antivirus engines.

Case Study on Incidental Malware – MIMICS Findings

- The research revealed that in a 90-day period there were around 3,157 unique infections around the world
- When filtered for the organizations (organizations may have numerous infected files) it led to a discovery of about 6,000 industrial sites around the world infected a year
- Other findings included highly sensitive files such as project files being directly uploaded to these databases by untrained IT staff and non-ICS Security MSSPs who used VirusTotal like a free sandbox; some highly sensitive findings were redacted from the overall report and SANS talk



count	3157.000000	25%	2.000000
mean	30.013937	50%	46.000000
std	21.142954	75%	48.000000
Min	1.000000	max	57.000000

Case Study on Incidental Malware – MIMICS Findings

The MIMICS findings showed definitively that around 6,000 industrial sites were infected in 2016—numbers that are reasonable for an annual average as well. Thus, 200'ish would be extremely low and 500k would be extremely high. However, these are not industrial attacks nor are they targeted, so not all of them would require incident response. It's simply infections.

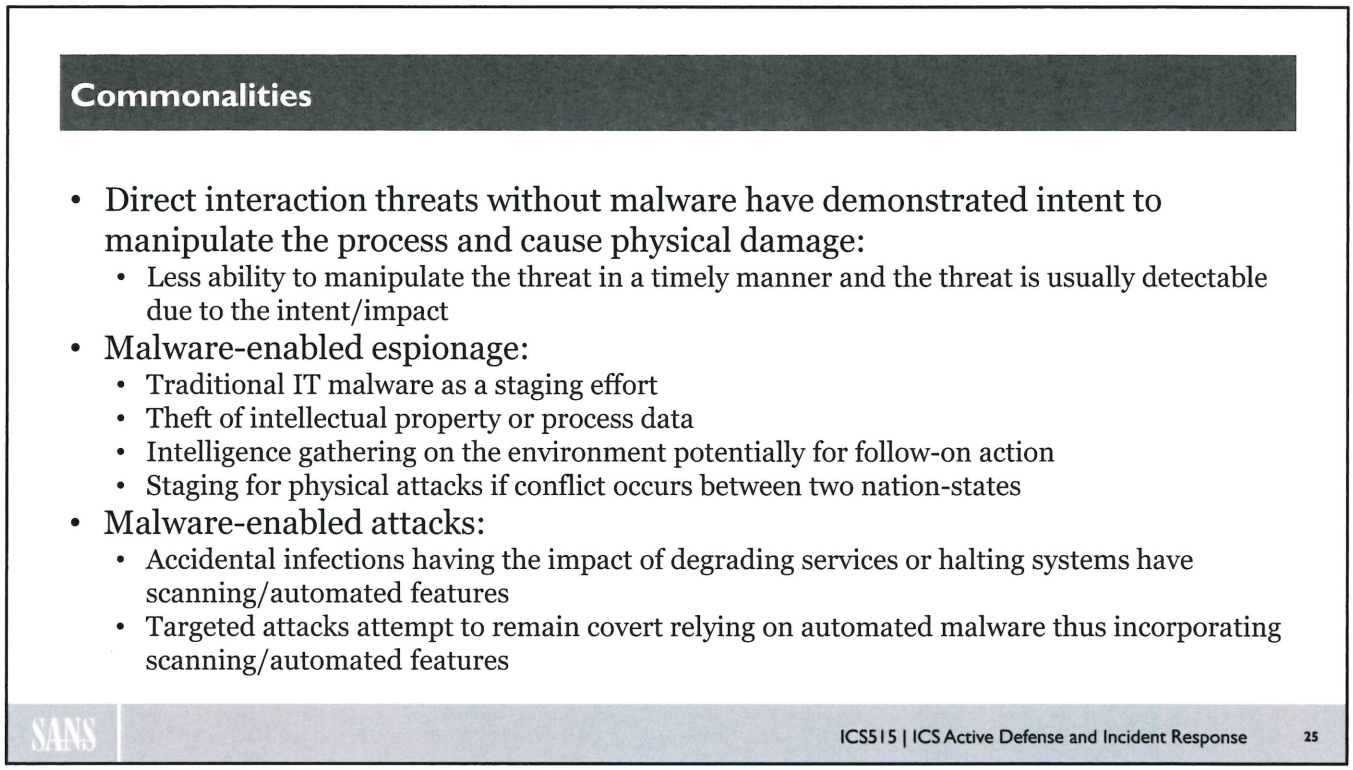
Link to the SANS talk, slides, and blog:
https://www.sans.org/summit-archives/file/summit-archive-1492176391.pdf
https://www.youtube.com/watch?v=Kq4Gm0NjPLo
https://dragos.com/blog/mimics/

Identified ICS Threat Tactics: Targeted Malware

- Destructive and untargeted malware:
  - For example, Shamoon targeted Saudi Aramco, failed to reach the ICS, but rendered 30k+ computers unusable
- Targeted ICS espionage malware:
  - For example, Havex was used to scan OPC environments and gather intelligence
  - For example, BlackEnergy was used to infiltrate ICS networks and gather intelligence
- Targeted ICS disruptive malware:
  - CRASHOVERRIDE had operations knowledge to disconnect a transmission substation
  - TRISIS had Safety Instrumented System (SIS) knowledge to disrupt operations
- Targeted ICS destructive malware:
  - For example, Stuxnet was tailored for a specific ICS network and caused massive physical damage

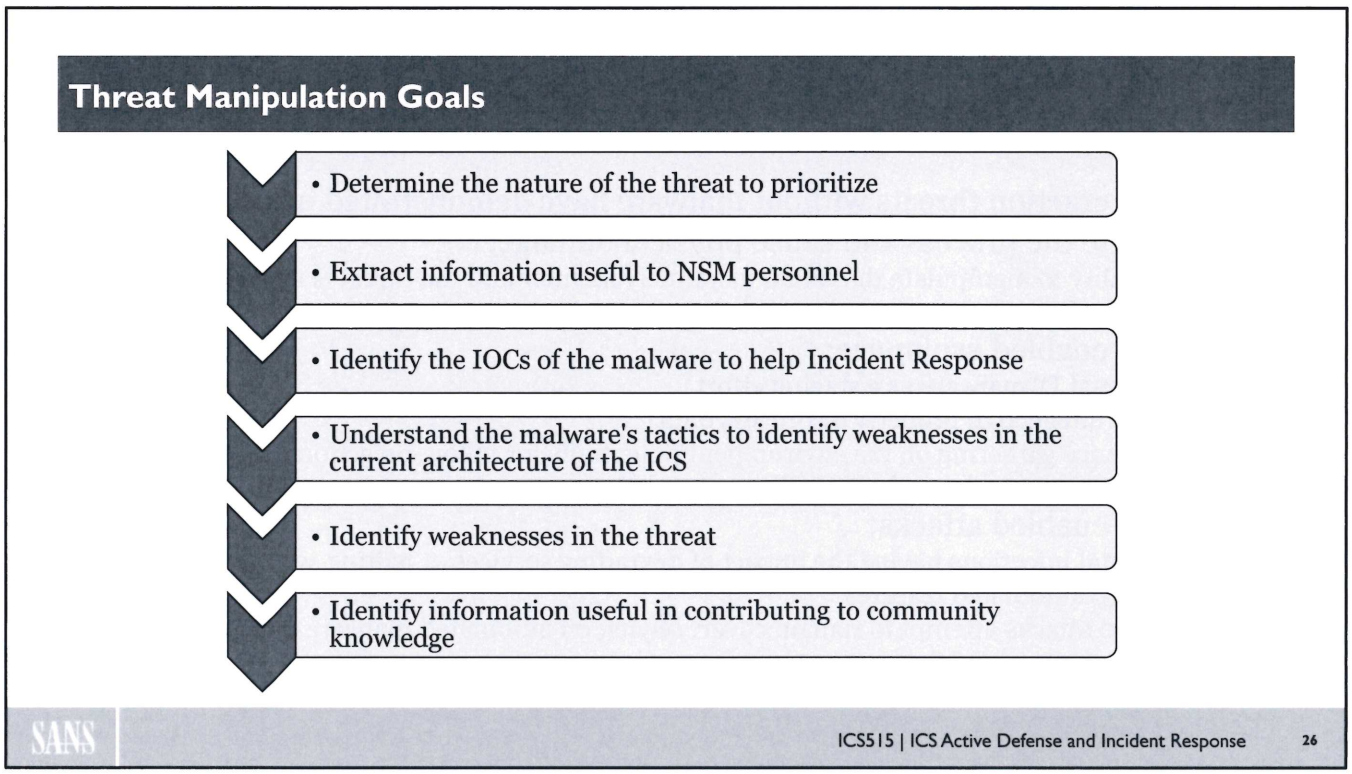
Identified ICS Threat Tactics: Targeted Malware

Note that Shamoon didn't impact the ICS, but it attempted to, and it did cause a lot of financial damage to Saudi Aramco. Traditional IT malware can be just as dangerous as targeted malware in the right conditions, though. In addition, traditional IT malware can serve as a first stage access point to the environment for the adversary to enter and further compromise the network. There are only five pieces of malware that specifically targeted the ICS from a code perspective (BlackEnergy, Havex, Stuxnet, CRASHOVERRIDE, and TRISIS) but remember malware is just a capability for the adversary and can be used in a variety of ways.



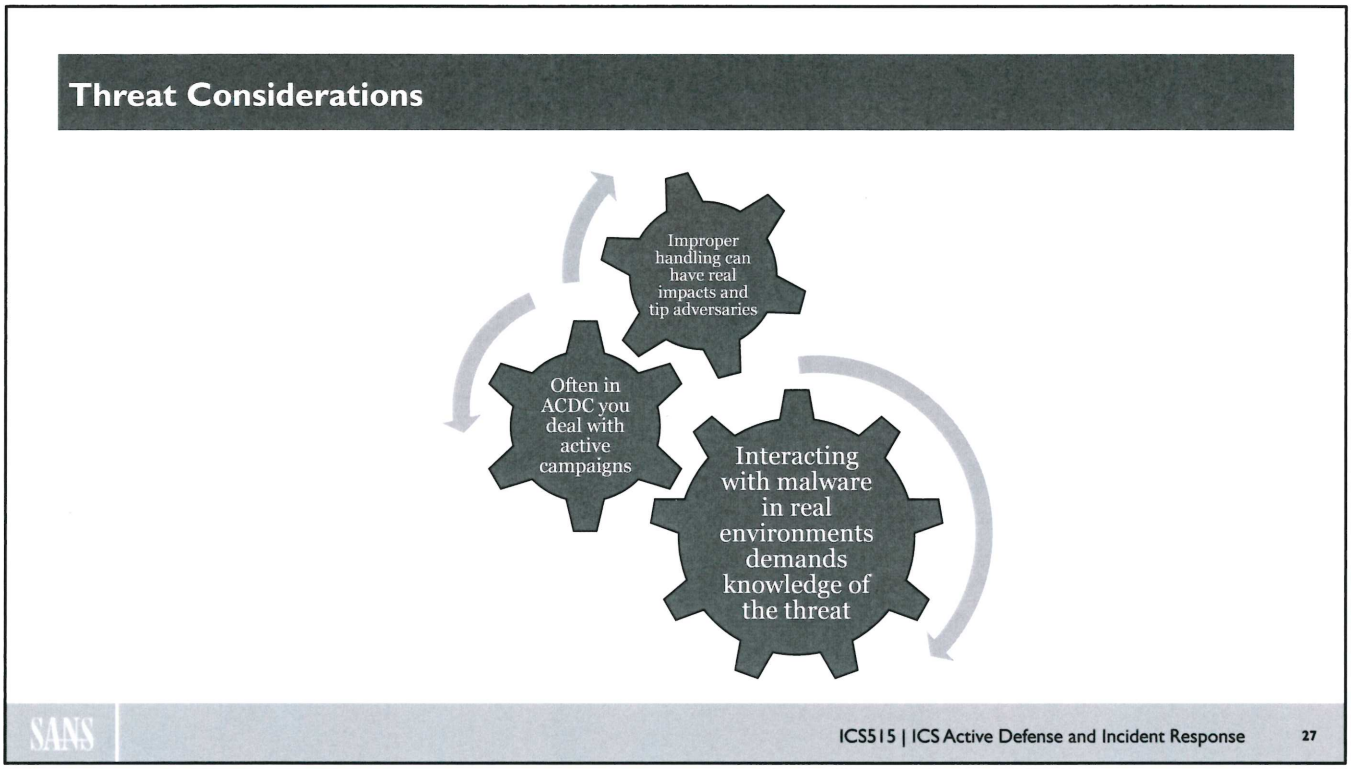
Commonalities

Extracting useful data from case studies can help encourage best practices for security professionals; we desperately need more case studies in the community. However, from the ones we have, we can glean a few lessons learned such as identifying the difference between malware-enabled espionage (like Havex) and malware-enabled attacks (like Stuxnet).



Threat Manipulation Goals
Manipulating a threat leads to gaining information for NSM and IR. In addition, the information can be shared with others in the form of threat data or threat intelligence. Malware doesn't want to be analyzed or run in VM environments and "tricked" into revealing its secrets…manipulation is not the offensive sounding word it can be; it's about being able to choose the battlefield and force threats/malware into situations it would rather not be in.

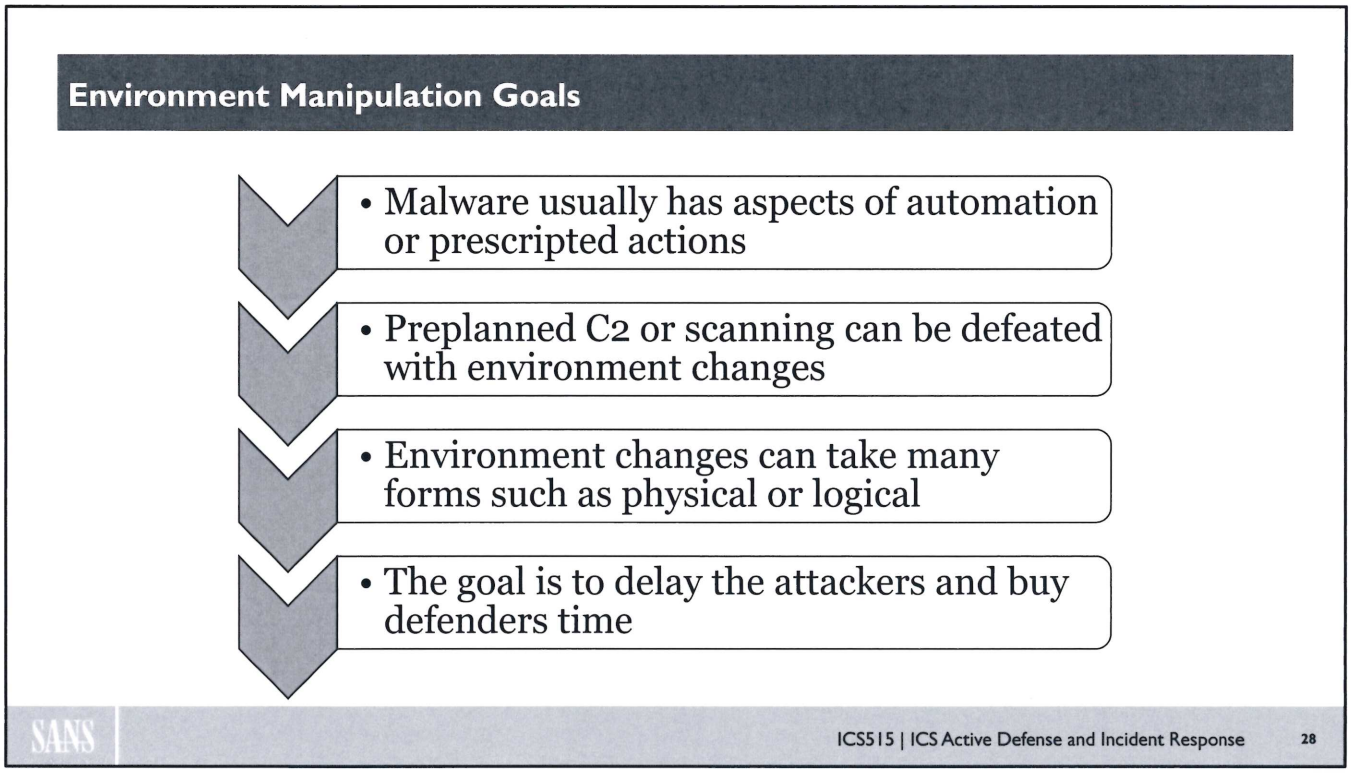
Also, attempt to identify weaknesses in the threat such as automated scanning or hardcoded IP addresses/passwords, which can be manipulated to neutralize the threat. Destructive malware observed in ICS has had automated features to limit the ability to directly attribute the attack and automated features such as scanning and exploitation can be manipulated internal to the victim network. Also, identify information useful in contributing to intrusion analysis and campaign analysis. Attribution should not be the goal, but the information gathered may be used in attribution attempts outside of the ACDC, especially when nation-state level malware is involved. Attribution is not useful for network defense but can be important for political impact and national-level responses, as well as board-of-director-level understanding and direction.



Threat Considerations

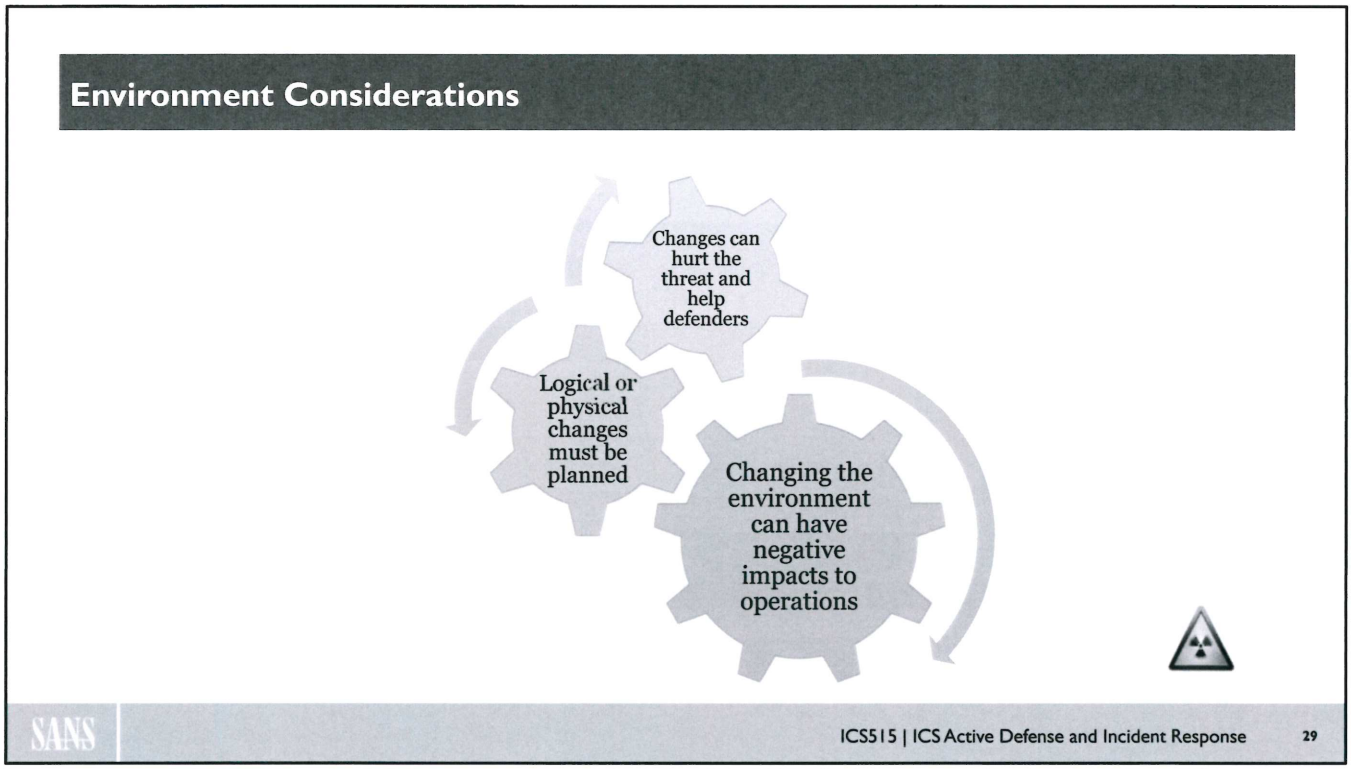
Because you are dealing with a real threat, there are some considerations to be made. First, a meaningful analysis of threats for ACDC means that at some point you are going to be running malware in an environment. A safe working environment (discussed in a few slides) is incredibly important. In addition, if a threat is already on the production network, you should not try to manipulate it on the production environment. Even if you identify a magic value that can be sent to the malware to neutralize, you need to ensure that it works 100% before doing it on the production environment and only when authorized, when it is scheduled, and as a last option.

Malware can have safeguards to activate unknown routines such as self-destruction or data-destruction. Lastly, improperly handling malware can tip adversaries that you're on to them, and they will change tactics. Do not let malware "call home" to an active Internet connection.



Environment Manipulation Goals

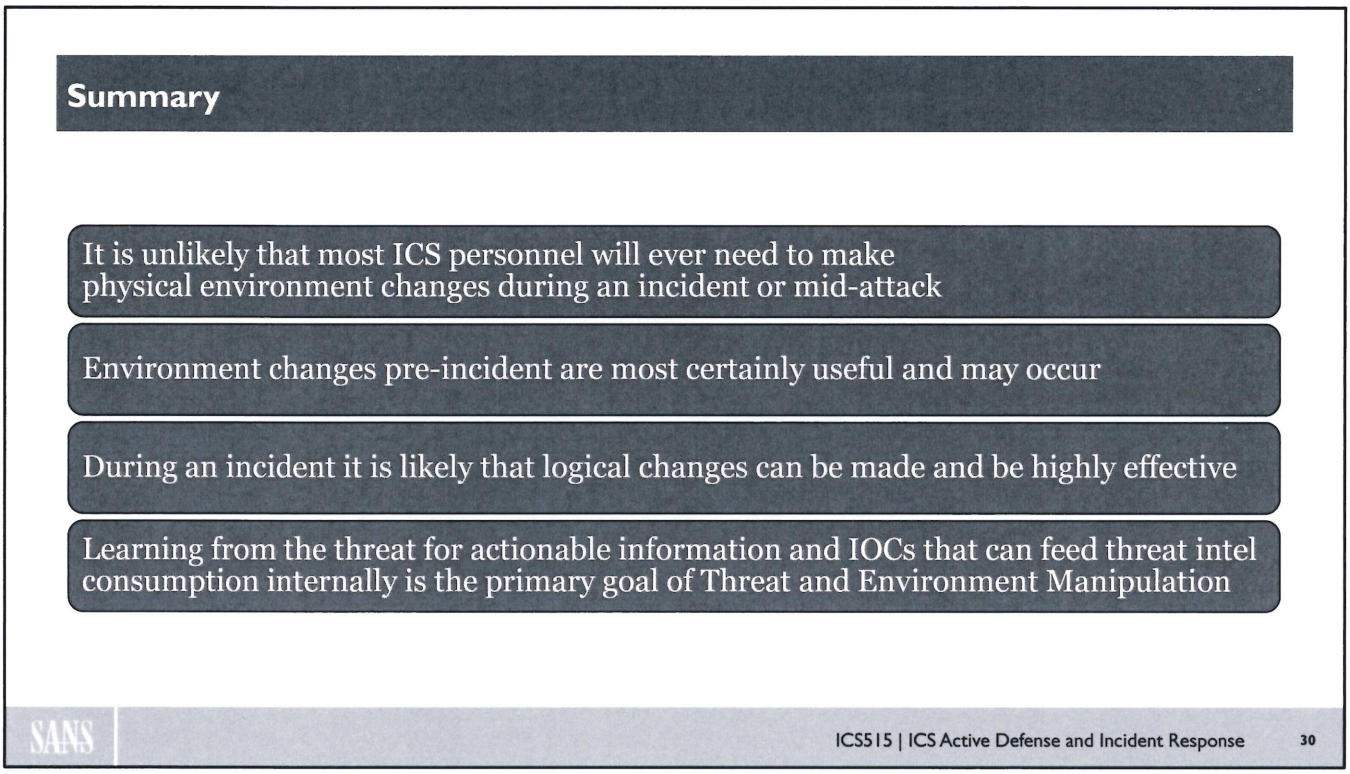
Automated malware has a lot of appeals, but it is especially enticing in ICS networks. Because the malware is mostly automated, it can be understood and manipulated so that routines, deactivation commands, or weaknesses can be identified. Sometimes, these weaknesses lead to an understanding that the automated malware is operating on previously gathered intelligence about the environment. Changing the environment then significantly delays or destroys the malware's capability to do any harm. However, realize you aren't "defeating" the adversary; defense is always an ongoing process. Environment manipulation can be logical or physical changes to the network: you can place false indicators on the network such as setting up honeypots, or you can change passwords on systems that might be hardcoded into the malware to work. We'll explore these topics a bit more in depth toward the end of the day.



Environment Considerations

In short, do not do anything without permission. ACDC personnel should not be actively changing the environment, even TEM-focused folks. Understanding the threat and environment to make recommendations is the role of ACDC TEM personnel. Changing the environment can have negative impacts and should be planned appropriately. Logical and physical changes before, during, or after an incident can confuse engineers and operators or impact the system process if it does not include input from all parties involved and high levels of testing/validation. Changing the network layout can introduce issues with vendor warranties and support as well and must be considered. Lastly, changing the network can also be done as a purely enabling strategy during an incident, such as the inclusion of network taps, collection points, and so on.

During an incident response scenario, the incident response personnel must be involved in the planning of environment changes so that it does not impact their operations and collection efforts.



Summary

It is unlikely that most ICS personnel will ever need to make physical environment changes during an incident or mid-attack:

- Knowing that the option is available and preparing scenarios of how it would occur is useful
- Just learning how to go about this process is useful to connecting people across teams and encouraging security talk

Environment changes pre-incident are most certainly useful and may occur:

- TEM analysts build on NSM personnel's knowledge of the architecture and identify weak spots that threats they've trained against or encountered can take advantage of
- Physical changes and segmentation strategies, as well as useful logical changes pre-incident, are helpful against threats and can be recommended to those in charge of the networked environment and ICS operations
- Always be mindful and careful of the #1 mission: safety and reliability

During an incident, it is likely that logical changes can be made and be highly effective.

- Most malware rely on C2 servers for commands or to upload exfil; sinkholing those requests provides a great opportunity for analysis while mostly neutralizing the malware

Learning from the threat for actionable information and IOCs that can feed threat intel consumption internally is the primary goal of Threat and Environment Manipulation. The ability to manipulate the threat and disable some of its functionality is a major win for defenders and should be explored whenever possible.

Manipulating the environment to make logical or physical changes useful in delaying the adversary or enabling defenders is often a more challenging and advanced tactic, but it is useful to understand the option exists:

- In the event of a particularly serious and targeted attack environment, changes can mean the difference between system failure and fighting through the attack

Analyzing Acquired Evidence

Evidence Handling
Applying Pre-Existing Information
Initial Attack Vectors
PDF Analysis
Extracting Files from Acquired Evidence

This page intentionally left blank.

General Malware Analysis in ICS Environments

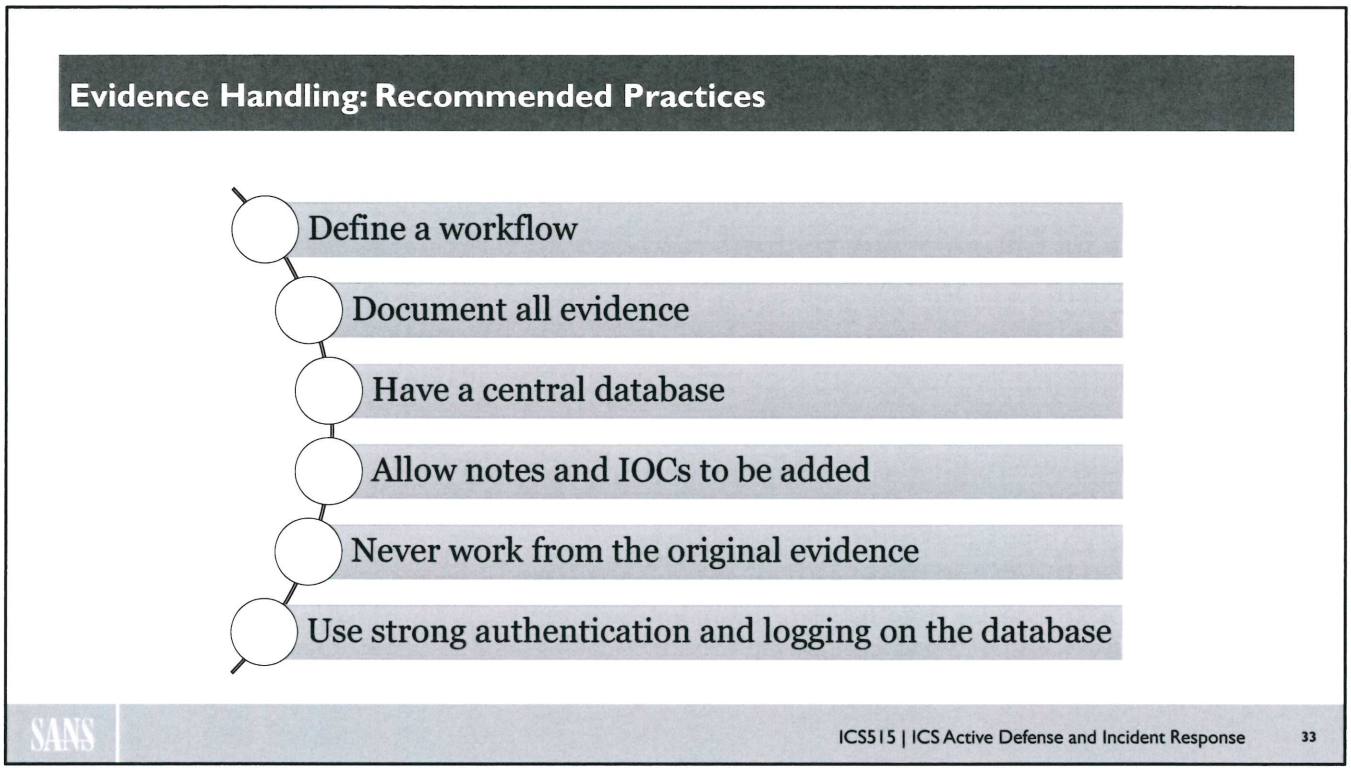
1. Establish a safe working environment (for example, use VMs)
2. **Document received evidence files (such as phishing email, pcap, memory files, and so on)**
3. **Perform initial analysis to identify useful information (for example, pre-existing YARA rules)**
4. **Identify and investigate interesting files, executables, and such**
5. **Extract suspect file(s) from the evidence and document them including hashes**
6. Run copies of the suspect files through in-house antivirus systems to identify matches:
   a. If the suspect files have known AV signatures, it is known malware, and online research can take place, such as looking up digital hashes, antivirus company threat reports, and more
   b. If the suspect file does not have a known AV signature, continue on (or, if at least some do not)
7. Begin processing the suspect file(s) through an in-house automated malware analysis sandbox if possible
8. Begin extracting information using behavioral malware analysis techniques
9. Use the analysis to create IOCs and provide context while documenting findings
10. Pass IOCs, documentation, and so on to those managing the internal threat intelligence products

General Malware Analysis in ICS Environments

Most threats involve malware, and today's focus is heavily centered around learning from malware. Even threats that don't rely on malware to impact the ICS may rely on malware to gain a foothold in the ICS or corporate network. The same thought processes and type of analysis that goes into malware analysis is useful to understanding nonmalware-based threats.

To help guide the day, the 10-step process is used throughout the day. This is a general methodology and may not represent the abilities and processes of every ICS security team, but it covers the vast majority of cases.

Throughout the 10-step process, analysts should think about what can be done to the architecture to deny or delay the adversary and its capabilities.



Evidence Handling: Recommended Practices

Evidence Handling

Everything needs to be documented. Who accessed what evidence, when did it take place, and what information was gathered? Documentation is king. Without proper documentation, teams will duplicate efforts and not learn from analysis. Analysis performed in one investigation or one training environment can be useful in others. Training with live malware and documenting findings can find commonalities between actual threats (maybe there is a reuse of malware) that are useful to have available.

Recommended practices:

- Define a workflow to allow evidence of different types to be looked at by those with relative skill sets
- Document all evidence with a digital hash and a short description
- Have a central database to "check out" evidence to ensure that multiple people do not work on the same piece
- Have the central database also have a capability to upload notes and detection signatures/IOCs to go with the evidence
- Never work from the original evidence file; always use a copy
- Have a plan and database to securely store previous analysis

When analyzing a current threat to the ICS, it is important to apply previous analysis and IOCs:

- No reason to reinvent the wheel if pre-existing information is present

Malware often changes some of its components, resulting in an overall different digital hash:

- However, individual components such as files, Registry keys, and so on that are manipulated or left behind when the malware runs on a system can match previous malware campaigns and provide context and insight on the threat
- Network connections such as C2 servers often provide correlating information between threat campaigns

YARA rules that already exist from the team or from outside the organization such as those found in ICS-CERT alerts, threat intelligence reports, and so on should be applied to the evidence before examining it to give analysts a starting place

Initial Attack Vectors

- The starting location for analyzing and understanding a threat is generally its initial attack vector
- Initial attack vectors include spear-phishing emails, exploitation of external websites, direct Internet connections to control systems, and abused trusted networks and connections between other networks, such as vendor or corporate networks:
  - Analyzing how a threat initially infected the ICS can drive remediation recommendations and can help identify and close the vulnerability the organization faced to reduce the chances of reinfection
- One of the most common initial attack vectors is phishing emails:
  - The next few slides demonstrate some analysis methods for phishing emails that may get passed to TEM personnel

Initial Attack Vectors

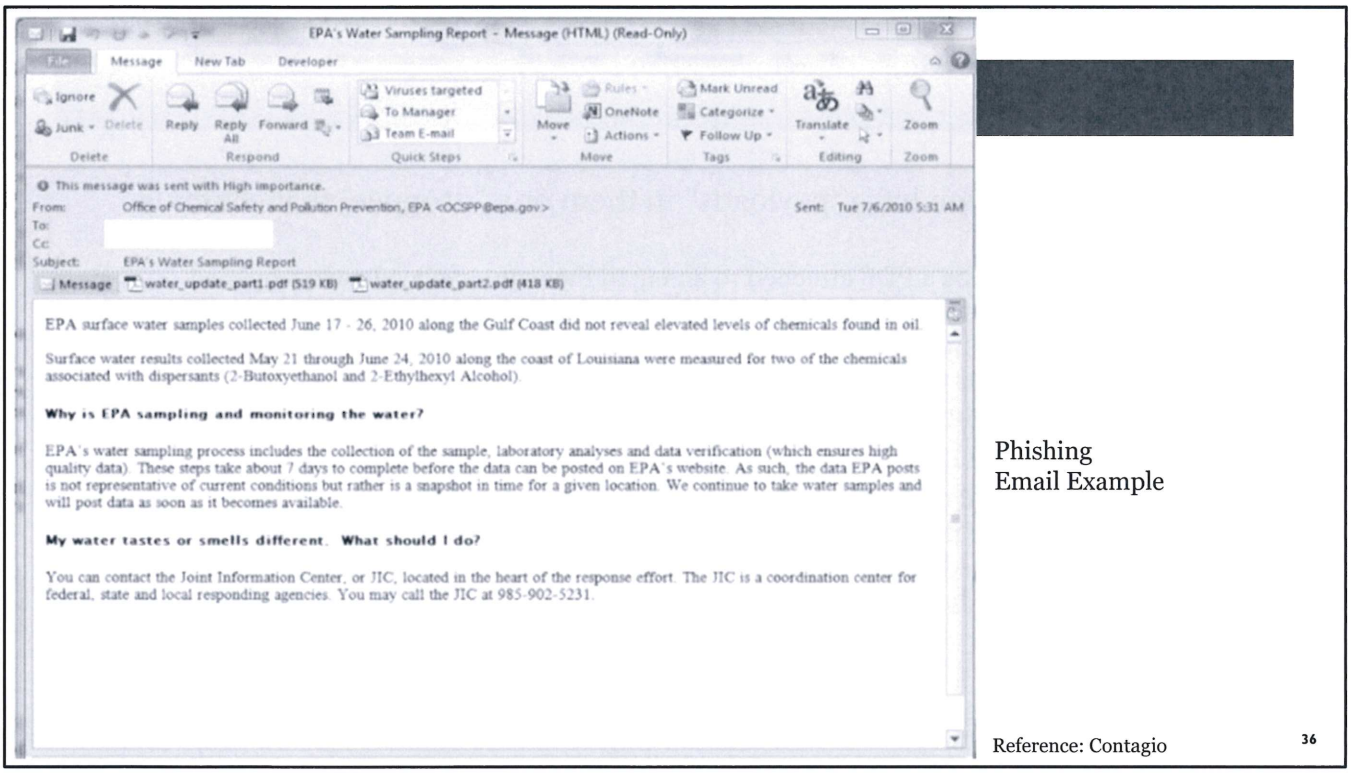
Initial attack vectors can vary including a variety of client-side attacks, but social engineering and phishing emails are extremely common because they are effective. Internet-connected PLCs are common and vulnerable, but phishing emails often give access to systems that the adversary needs to fully understand and pivot throughout a network. For this reason, we'll look at phishing emails a bit and some of the ways to analyze them for indications of being malicious.

Phishing Emails

- Phishing emails often have "payloads" in them or a "dropper" that contains malicious code:
  - An example would be a PDF attached to an email that runs shellcode when opened
  - Shellcode is a specific type of code that is used as the payload of an exploitation attempt; it usually starts a command shell (thus the name shellcode) to run and execute processes, commands, and sets up a connection point
- Because phishing emails have to take advantage of some vulnerability to introduce malware into the environment or open up a connection for a remote adversary, malware can be examined by TEM personnel
  - Given the nature of the malware as the initial attack vector for the adversary, the information gathered from it is extremely important, often representing the C2 servers of the adversary or indications into how advanced the threat is

Phishing Emails

The email portion of a phishing email is usually not the capability that the adversary uses. Most often, there is a file attached that must be run for the "payload" or "dropper" (the malicious code) to be executed on the system. Often this includes shellcode that allows adversaries access to the system. Phishing emails often have malicious code in them, even if contained in something like a PDF, and must be handled with care.

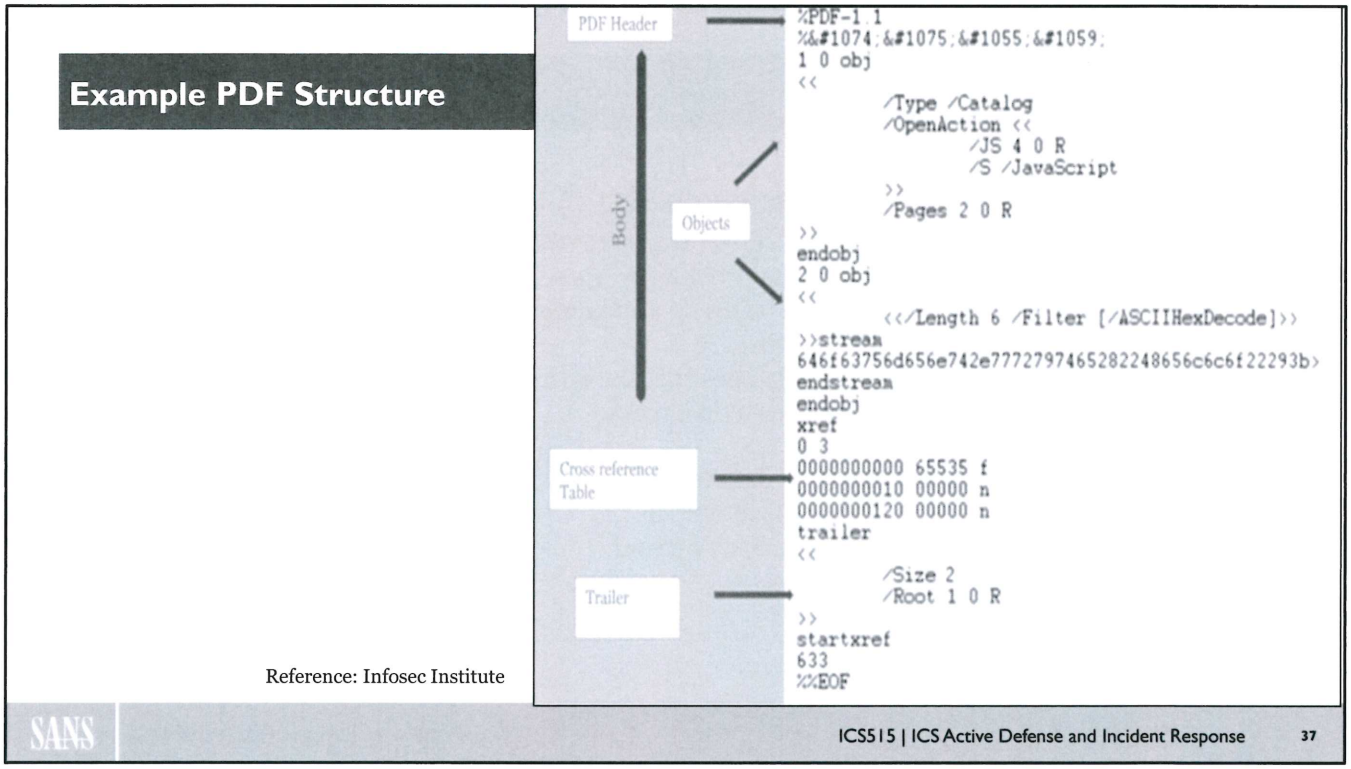


Phishing Email Example

Here's a phishing email that was found as part of an APT campaign. Contagio is a website that has a wide source of phishing emails, malware, and information useful for TEM personnel to get samples to train on. In this case, a well-tailored phishing email includes PDFs containing an exploit that gives an attacker access to the system that runs it. Here, we see water targeted from an EPA.gov email address. (The email address was spoofed.)

All around this is a believable email and only one person in the organization has to click it. But having access to the email and the PDFs can reveal information vital to identifying the threat after it has breached the organization. Until you lose something of value or the adversary gains what they were after, there is no "defeat" in this threat getting in. But responding is vital.

As an aside, a common misconception is that phishing emails have misspelled words like the Nigerian prince emails because the adversaries aren't that good. The truth is that research has shown that the Nigerian Prince emails often contain bad grammar and obvious indicators of malicious activity because those running the campaign want to get the gullible individuals. The type of people who respond to those emails is the type who might give over their credit card information. That is, the Nigerian Princes often just really know their target audience. Adversaries to ICS are the same way.



Example PDF Structure
So why are PDFs so common? PDFs are incredibly popular and people are used to opening them. Adobe also is a relatively easy application to find vulnerabilities in. But, as you can see in this example, there's a lot of space to do things in a PDF. You can embed JavaScript or other commands as "objects" and embed malicious code that can be called and executed. There are known indications of malicious behavior because people have analyzed enough PDFs to realize what common areas are malicious.

A PDF file is composed of header, objects, cross-reference table (to locate objects), and a trailer.

Reference:
http://resources.infosecinstitute.com/analyzing-malicious-pdf/

PDF Aspects to Focus On

- SANS Instructor Lenny Zeltser recommends (on his website) to focus on specific PDF file formats because they have been observed in various malicious instances:
  - /OpenAction and /AA (Additional Action) specify the script or action to run automatically
  - /Names, /AcroForm, /Action can also specify and launch scripts or actions
  - /JavaScript specifies JavaScript to run
  - /GoTo* changes the view to a specified destination within the PDF or in another PDF file
  - /Launch launches a program or opens a document
  - /URI accesses a resource by its URL
  - /SubmitForm and /GoToR can send data to a URL
  - /RichMedia can be used to embed Flash in PDF
  - /ObjStm can hide objects inside an Object Stream

PDF Aspects to Focus On
Lenny Zeltser is going to come up a few times in this course; he is the guru on malware analysis and teaches at SANS. He put together a list on his website of some specific areas of concern to look for in PDF files. You could create an IOC out of this, but there are already tools available that do this.

None of these should be seen as obvious indicators of malicious activity and some can be quite common, but they are good commands to highlight and look for—specifically, learn what's normal for your environment and look for commands that aren't normal. Additionally, a high number of each of these can indicate malicious activity.

Reference:
http://zeltser.com/reverse-malware/analyzing-malicious-documents.html

PDFiD

- A simple tool to scan for strings in PDFs that may be malicious
- Created by Didier Stevens in 2009
- Outputs the number of times a PDF document has the set strings in it
- Useful for quick analysis of a single document given an understanding of what PDF objects and strings may be malicious:
  - For example, combine your knowledge or a list (such as the Lenny Zeltser one) with this tool to perform a quick strings search

```
# pdfid.py notepad.exe
PDFiD 0.0.2 notepad.exe
 Not a PDF document


# pdfid.py test.pdf
PDFiD 0.0.2 test.pdf
 PDF Header: %PDF-1.1
 obj                    7
 endobj                 7
 stream                 1
 endstream              1
 xref                   1
 trailer                1
 startxref              1
 /Page                  1
 /Encrypt               0
 /JS                    1
 /JavaScript            1
 /AA                    0
 /OpenAction            1
 /JBIG2Decode           0
```


PDFiD
This is a simple but powerful tool created by Didier Stevens that quickly scans a PDF for keywords and key objects that are likely malicious. It returns a count of the different keywords it finds.

References:
http://blog.didierstevens.com/2009/03/31/pdfid/
The image is taken from the above reference

AnalyzePDF

- AnalyzePDF is a Python script that uses PDFiD, PDFinfo, and YARA rules to analyze PDFs:
  - One of the newer PDF analysis tools (2013) to be created with the intent of watching for well-known "bad" signs
- Developed by Mandiant analyst Glenn P. Edwards, Jr. as an attempt to build upon the available tools and make something smarter that could use YARA rules:
  - He analyzed 13k+ PDFs (clean and malicious) to determine common patterns that may indicate maliciousness
- Great resource for incorporating YARA rules with traditional PDF scanning tools (PDFiD and PDFinfo) to get a better idea on if a PDF is malicious

AnalyzePDF
AnalyzePDF is a tool created by Mandiant analyst Glenn P. Edwards, Jr. in an attempt to combine the functionality of PDFiD, another tool known as PDFinfo, and YARA rules. In this way, it allows the tool to use tactical threat intelligence and identify malicious documents. He analyzed thousands of known good and known bad documents to tailor the tool to focus on common malicious patterns.

Reference:
http://hiddenillusion.blogspot.com/2013/12/analyzepdf-bringing-dirt-up-to-surface.html

AnalyzePDF Example

```
remnux@remnux:~/Desktop$ AnalyzePDF.py -y pdf_rules.yara Mandiant_cve

==================================
[+] Analyzing: Mandiant_cve
----------------------------------
[-] Sha256: d0375fb2448e91b47b97f3fb132a6eafd04974da5496c55adb2bdb310e9f5ea3
[-] JavaScript count.......: 1
[-] Open Action............: 1
[-] AcroForm...............: 1
[-] Total Entropy..........: 4.328586
[-] Entropy inside streams : 2.187668
[-] Entropy outside streams: 4.502080
       [*] Entropy of outside stream is questionable:
       [-] Outside (4.502080) +2 (6.502080) > Total (4.328586)
[-] (1) page PDF
[-] YARA hit(s): [suspicious_js, invalid_trailer_structure, date_value_error]
----------------------------------
[-] Total YARA score.......: 5
[-] Total severity score...: 16
[-] Overall score..........: 21
==================================
[!] HIGH probability of being malicious
```

Reference: Glenn P. Edwards Jr.

41

AnalyzePDF Example
Notice that this tool gives the SHA256 hash, counts some of those interesting commands noted earlier, displays entropy, and allows the use of YARA directly against the PDFs from this tool. This is a great tool to be able to use for those reasons.

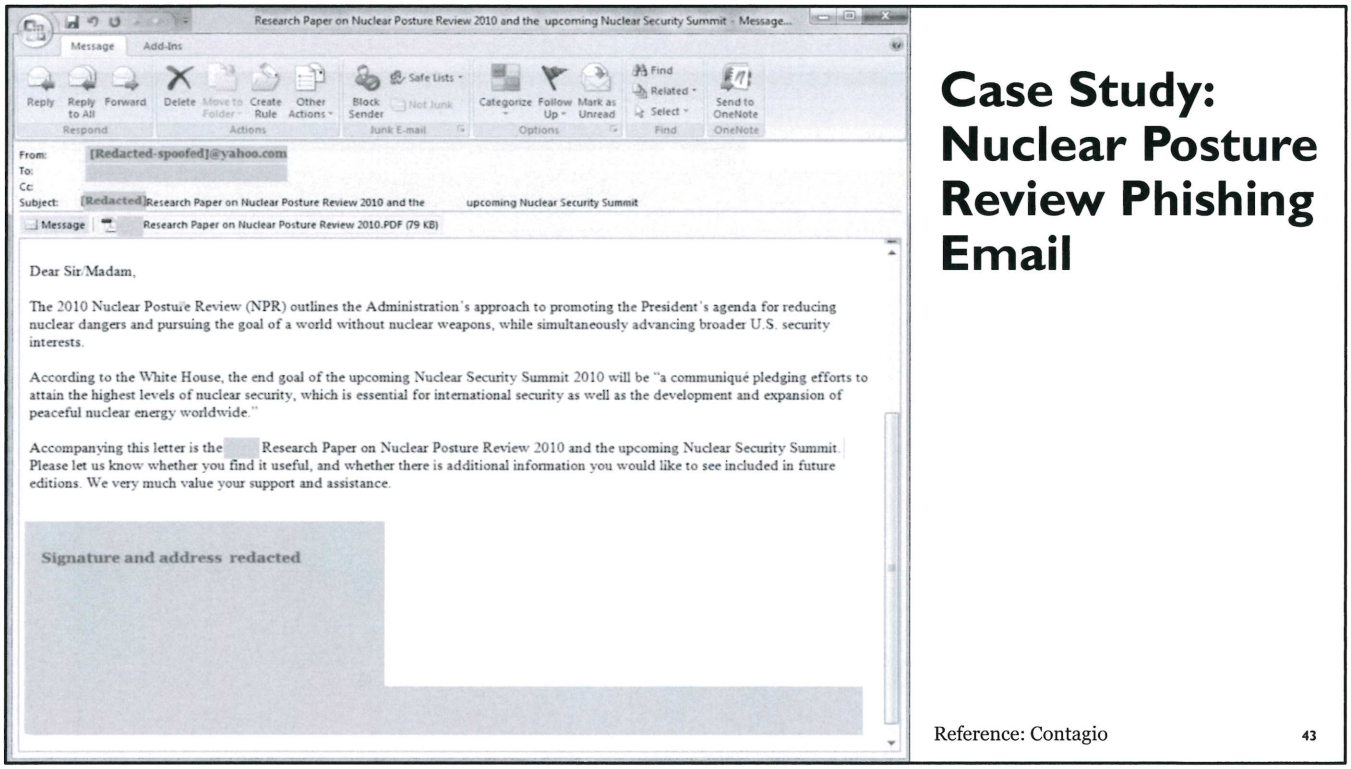
Reference:
http://hiddenillusion.blogspot.com/2013/12/analyzepdf-bringing-dirt-up-to-surface.html

Automated PDF Analysis

- Automated malware analysis will be discussed later today; however, be aware that it is possible to put a PDF into an automated malware analysis platform and let it execute its malicious code:
  - This lets an analyst determine not just if the PDF is malicious, but also what kind of activity the PDF performs
- It is a good practice to use the manual PDF tools to analyze PDFs that may be malicious and then let the potentially malicious PDFs run in an automated analyzer instead of wasting time
- The next few slides demonstrate a case study of automated malware analysis on a phishing email that was targeting nuclear security community members

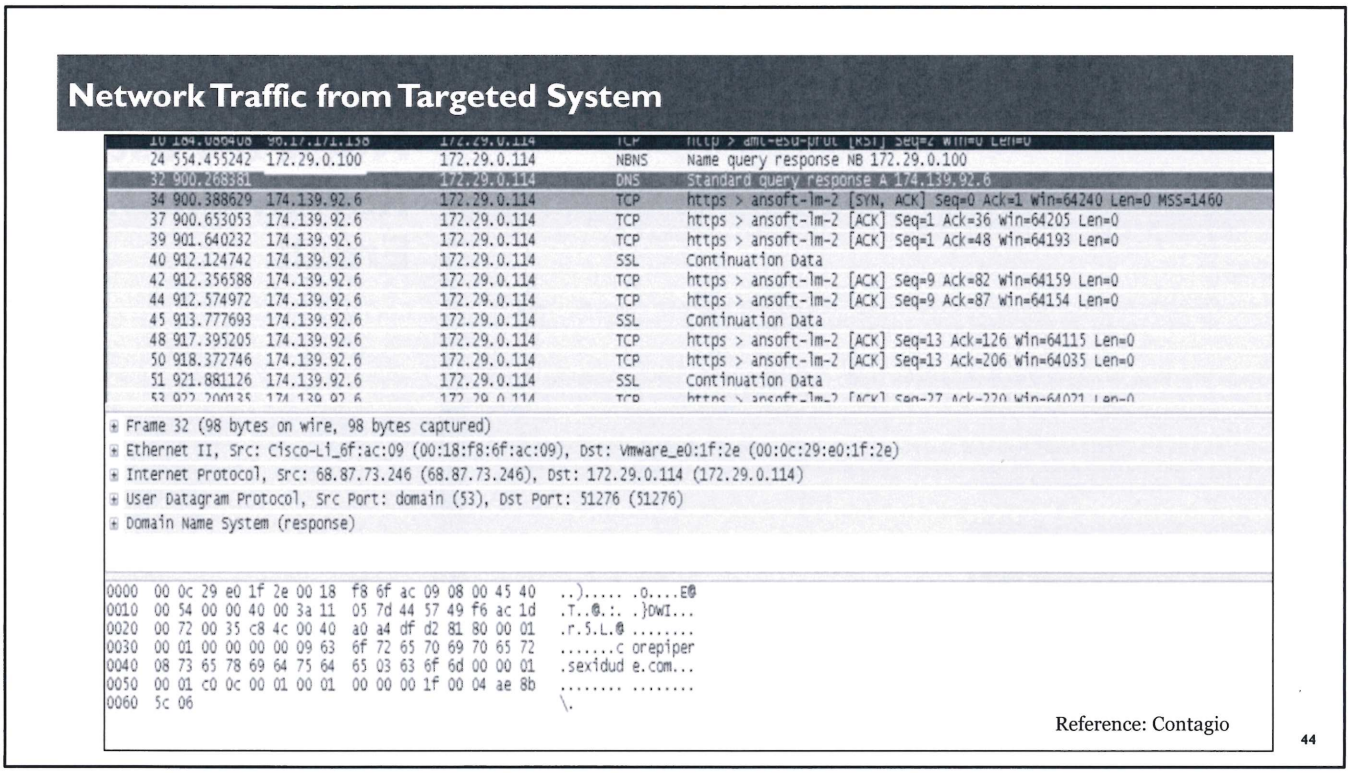
Automated PDF Analysis

Automated malware analysis is a great resource for analysts, and it's one of the types of malware analysis methodologies. It will be explored later in depth, but realize at this point that it's possible to just put a malicious PDF into an automated malware analysis platform and it will extract useful information.



Case Study: Nuclear Posture Review Phishing Email

Over the next few slides, you'll see some analysis that was done on an APT campaign where an adversary targeted members of the nuclear community and specifically a summit that was taking place in DC in 2010. After the PDF was opened, it executed on the system and started sending communication back to the adversary's command and control (C2) server.



Network Traffic from Targeted System

Here, we see the communication that took place on the infected system. The Contagio analyst opened the PDF and had Wireshark running to capture the network communication. We see that the C2 server is located at the website sexidude.com. (Real website, actually malicious, don't go there.) A large amount of analysis could take place from there and what information is gathered from this network capture (such as looking up whois information for the website, seeing what other malware campaigns it's been associated with, and looking for links in other malware).

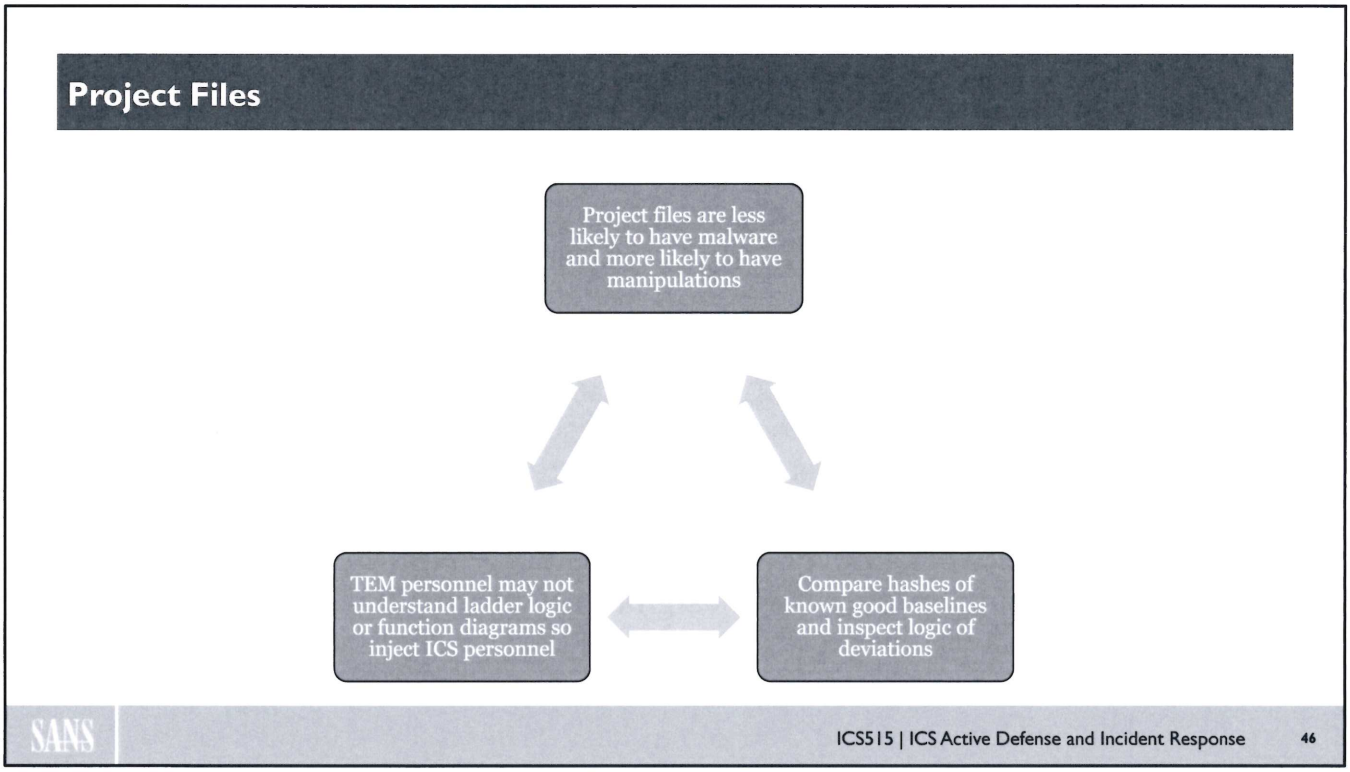
Analysis of Nuclear Phishing Email

- Automated analysis by the Contagio researchers in the Anubis sandbox showed that the PDF, when opened, created the file %Temp%\AcrRd32.EXE with the hash MD5 5a67c2a64e17a2e3e5efd0ae94db715c
- The executable then created and opened:
  - %Temp%\11111111.pdf MD5 6b4162954594a6c6e4287773fced7e5f
  - %Temp%\wuweb.exe MD5 8ae20aabfb207f5bb4e3918b043d37fa
- Through automated analysis alone, a TEM team member could have written a YARA rule off of the digital hashes and filepaths/executables as well as given network information (the IP address for the C2 server) to NSM personnel to create network signatures in an IDS:
  - This would have also quickened any Incident Response efforts to locate the infected systems

Analysis of Nuclear Phishing Email
Analysis performed by the Contagio researchers showed that the PDF created an initial file and then a few others. This is the type of information TEM personnel could gather and pass to incident responders looking for infected systems. The network data could be passed to NSM personnel to look for network based indicators.

Additional information:
Took advantage of the CVE-2010-0188 vulnerability in Adobe Reader.
The summit was in DC in 2010.
At the time, less than 13 percent of AV caught the threat.

Reference:
http://contagiodump.blogspot.com/2010/04/apr-10-cve-2010-0188-pdf-research-paper.html



Project Files

Project files and logic on controllers don't often come to mind when thinking about malware and threats. However, any adversary actually wanting to manipulate a system's process or cause damage is going to have to at some point manipulate the logic or project files. Stuxnet was known to have manipulated the project files of the Siemens controllers. There's no quick-and-easy way to analyze for malicious routines or indicators of compromise. The concept seems like it would not be feasible. (Small changes can mean big damage, and there's nothing that looks or is inherently malicious.) However, gathering hashes of these files and having Incident Response personnel hash the project files during an incident will allow TEM personnel to identify if there's any manipulations that have occurred. Then, engineers can be involved to determine what the threat is trying to do and how to stop it. Also, consider having discussions with your lawyers to ensure record retention and eDiscovery issues do not conflict with project file needs or information protection program requirements in your organization.

Course Scenario

This page intentionally left blank.

Traffic Lights – Phishing Emails

The Traffic Lights network personnel have identified a person on staff that likely self-infected the HMI with the malware in an attempt to get more attention and budget for security.

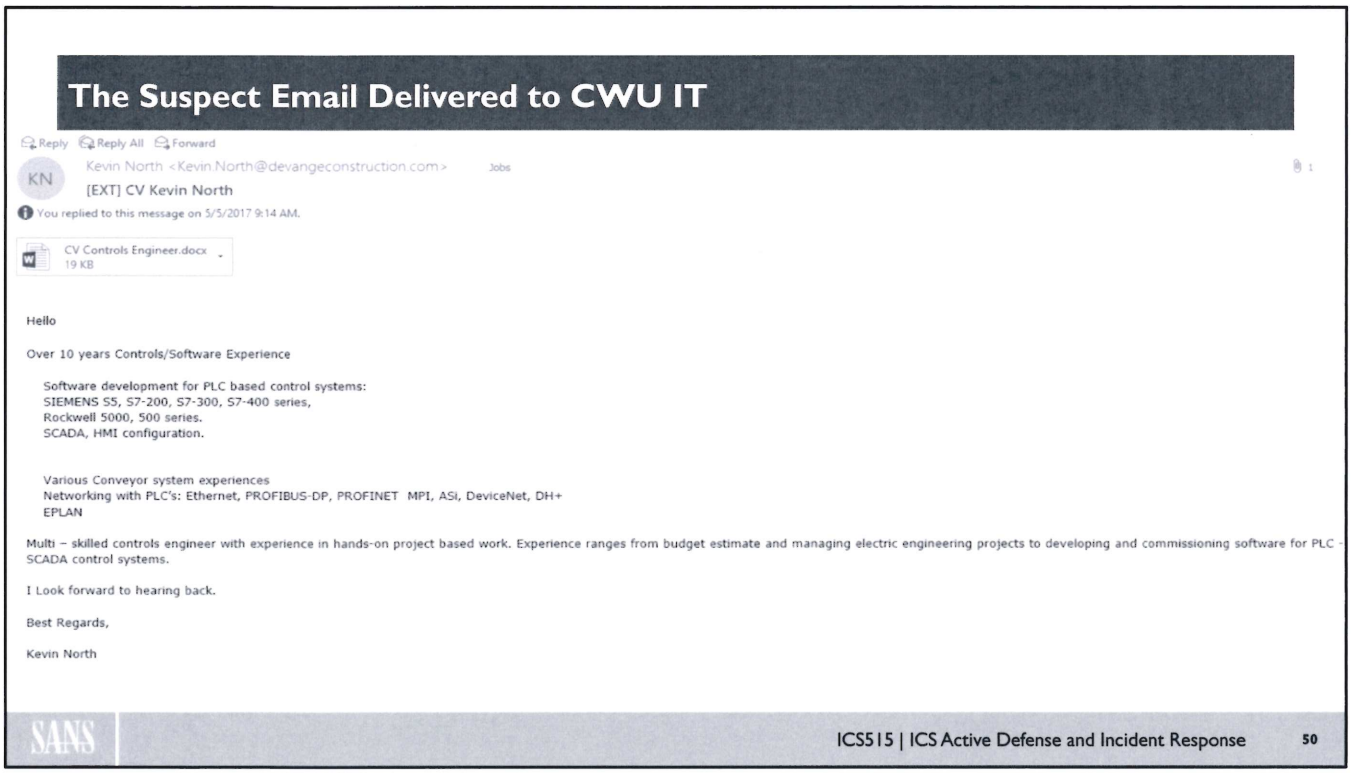
However, the city CSO wants to ensure there aren't things they are missing before they make this determination. They want you to analyze phishing emails received by the organization and determine if they are malicious. It is understood by the security personnel that this was likely not the infection vector but, to appease the CSO, the analysis is needed.

This page intentionally left blank.

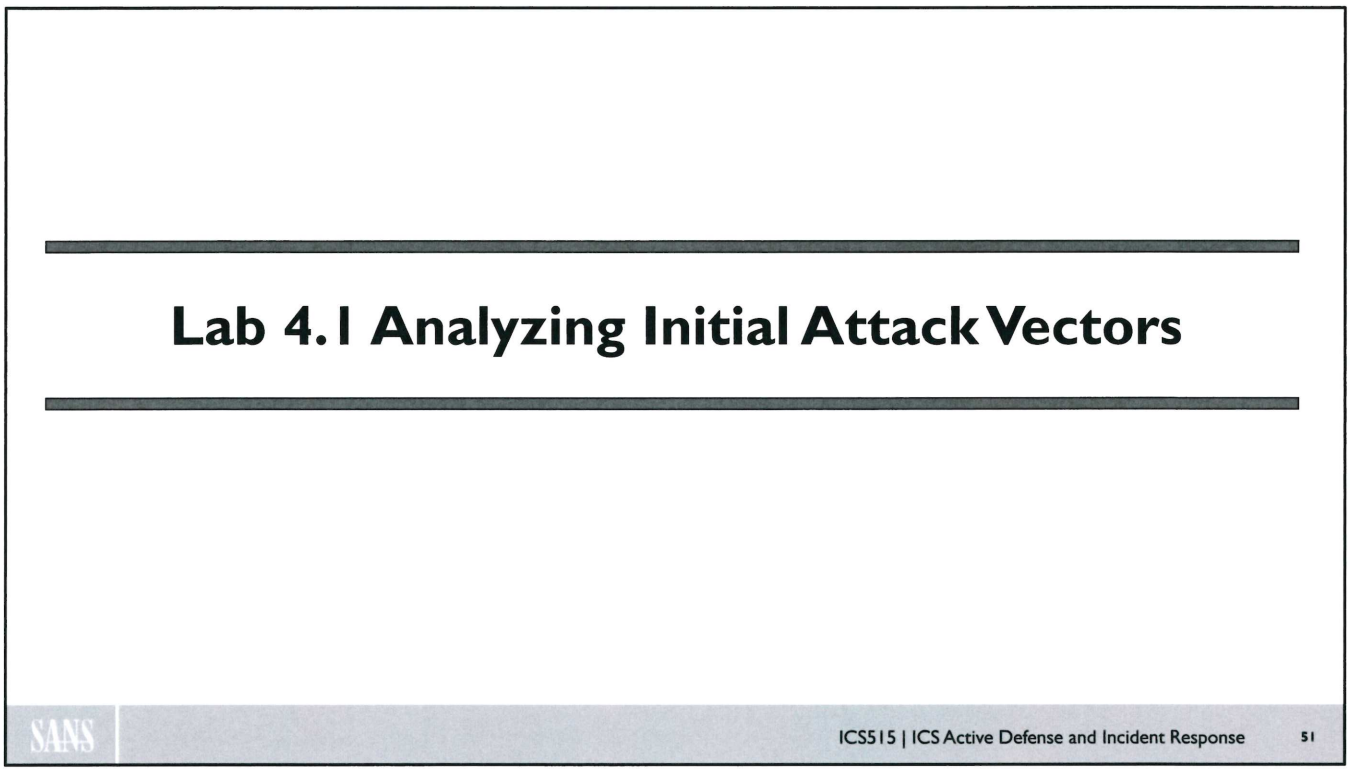
CWU – Phishing Email

- CWU did receive a likely phishing email to its IT networks and there was odd activity on the business network historian, which does have connectivity to the water plant
  - This is the system that was leveraged for the SCADA Hijack attack
- The phishing email does not appear to have any malicious scripts or compiled code in it, though, so it will be run in a sandbox in Lab 4.2

This page intentionally left blank.



This page intentionally left blank.



Lab 4.1: Analyzing Initial Attack Vectors
Reference the Lab Workbook.

Traffic Lights – Recap

- The phishing emails were actually malicious, but they would not have been found to be the infection vector into the Traffic Lights network
- The employee ended up coming forward and admitting to having infected the system but claims to have had nothing to do with the traffic lights attack that took place

This page intentionally left blank.

Case Study:

The Ukraine Cyber Attack

This page intentionally left blank.

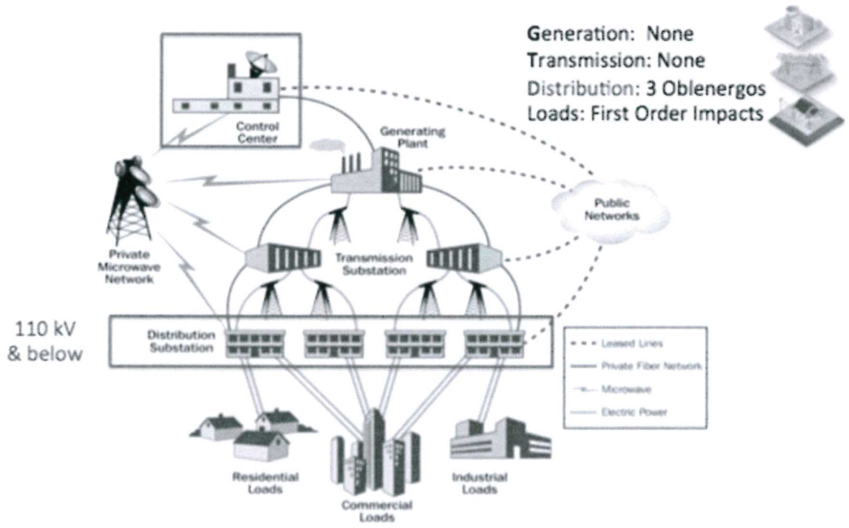
The Incident

- On Dec 23, 2015, a Ukrainian Oblenergo reported outages
- Skepticism was the initial response
- Analysis of the ICS and what could cause the issue as well as an uncovered malware sample helped confirm the attack
- The attack was blamed on the Russian services by the Ukrainians
- The Sandworm team was linked in the attack
- Overall, 3 Oblenergos, ~62 substations were disconnected, leaving ~225,000 customers without power for 3-6 hours
- Ukraine was in manual operation for months

The Incident
On December 23, 2015, attackers leveraged access that they had for the previous six months to knock three regions of Ukraine off the grid. Overall, around 62 substations were disconnected and the outages lasted between 3-6 hours, depending on the region. The power was turned back on relatively quickly because of the Ukrainian's ability to go back to manual operations, but they had to persist in manual operations for multiple months.

From the official report:
On December 23, 2015, the Ukrainian Kyivoblenergo, a regional electricity distribution company, reported service outages to customers. The outages were due to a third party's illegal entry into the company's computer and SCADA systems: Starting at approximately 3:35 p.m. local time, seven 110 kV and 23 35 kV substations were disconnected for three hours. Later statements indicated that the cyber attack impacted additional portions of the distribution grid and forced operators to switch to manual mode. The event was elaborated on by the Ukrainian news media, which conducted interviews and determined that a foreign attacker remotely controlled the SCADA distribution management system. The outages were originally thought to have affected approximately 80,000 customers, based on the Kyivoblenergo's update to customers. However, later it was revealed that three different distribution oblenergos (a term used to describe an energy company) were attacked, resulting in several outages that caused approximately 225,000 customers to lose power across various areas. Shortly after the attack, Ukrainian government officials claimed the outages were caused by a cyber attack, and that Russian security services were responsible for the incidents. Following these claims, investigators in Ukraine, as well as private companies and the U.S. government, performed analysis and offered assistance to determine the root cause of the outage.

What Are Power Grids Like?

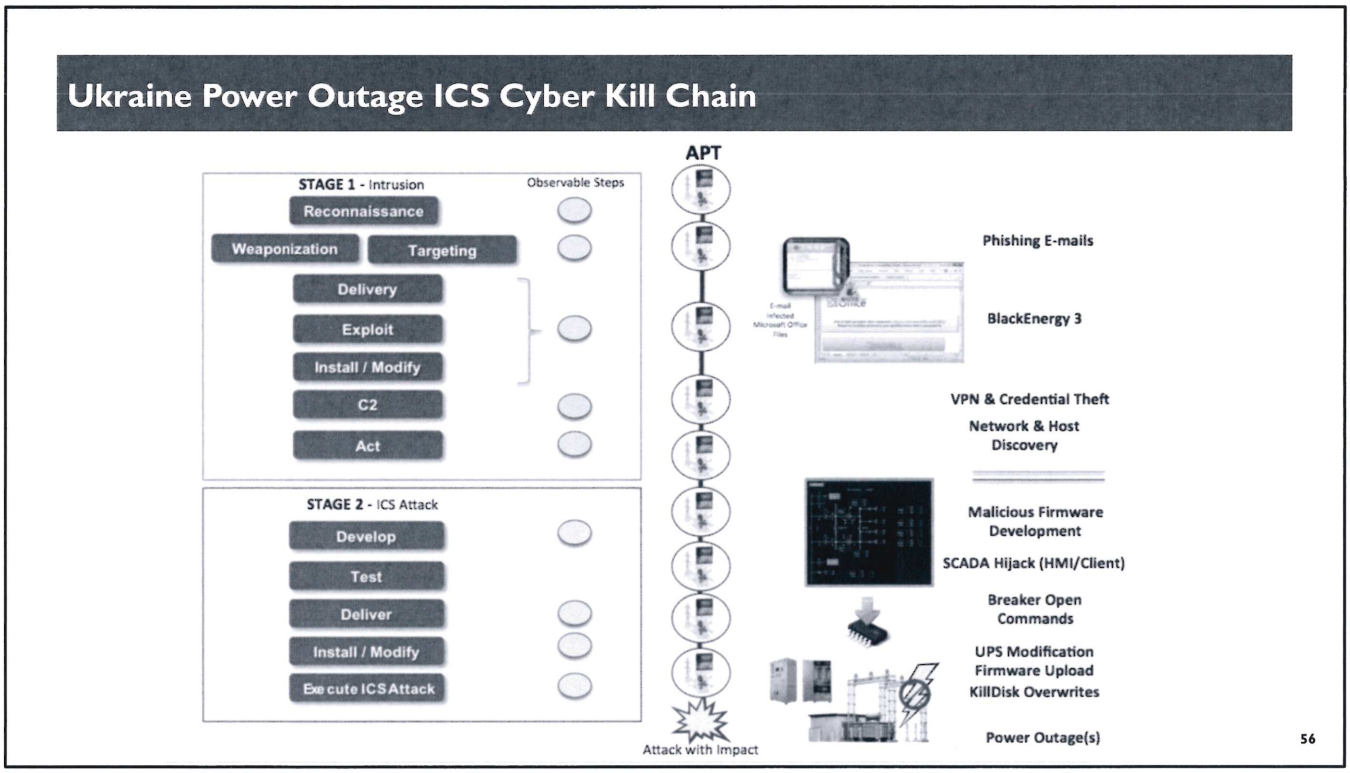


Source: Modification to the DHS Energy Sector-Specific Plan 2010

55

What Are Power Grids Like?

In Ukraine, three Oblenergos were impacted. Oblenergos are the Ukrainian version of energy companies. In a power grid there are generally three main components: Generation, Transmission, and Distribution. Historically, we have prioritized Generation and Transmission security which was well placed. Unfortunately, Distribution is the soft underbelly; many companies attempt to secure these sites anyway but many companies do not nor are there any regulations or requirements to do so (not that that is the best mechanism anyway). The Ukrainian attackers targeted Distribution.

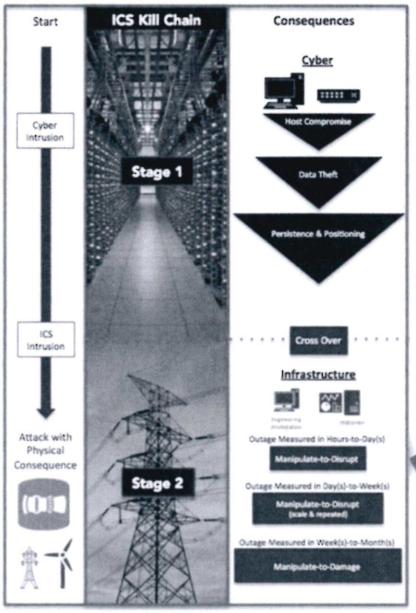


Ukraine Power Outage ICS Cyber Kill Chain

The first step in Stage 1 is Reconnaissance. There were no reports of observed reconnaissance having taken place prior to targeting the energy companies. However, an analysis of the three impacted organizations shows they were particularly interesting targets due to the levels of automation in their distribution system; enabling the remote opening of breakers in a number of substations. Additionally, the targeting and final attack plan for the electricity companies in general were highly coordinated, which indicates that reconnaissance took place at some point. This was very unlikely to have been an opportunistic attack. The second step is Weaponization and/or Targeting. Targeting would normally take place when no weaponization is needed; such as directly accessing Internet-connected devices. In this attack, it does not appear that targeting of specific infrastructure was necessary to gain access. Instead, the adversaries weaponized Microsoft Office documents (Excel and Word) by embedding BlackEnergy 3 within the documents. Samples of Excel and other office documents have been recovered from the broader access campaign that targeted a multitude of organizations in Ukraine; including Office documents used in the specific attack against the three electricity companies. During the cyber intrusion stage of Delivery, Exploit, and Install, the malicious Office documents were delivered via email to individuals in the administrative or IT network of the electricity companies. When these documents were opened, a popup was displayed to users to encourage them to enable the macros in the document. Enabling the macros allowed the malware to Exploit Office macro functionality to install BlackEnergy 3 on the victim system and was not an exploit of a vulnerability through exploit code. There was no observed exploit code in this incident. The theme of using available functionality in the system was present throughout the adversary's kill chain. Upon the Install step, the BlackEnergy 3 malware connected to command and control (C2) IP addresses to enable communication by the adversary with the malware and the infected systems. These pathways allowed the adversary to gather information from the environment and enable access. The attackers appear to have gained access more than six months prior to December 23, 2015, when the power outage occurred. One of their first actions happened when the network was to harvest credentials, escalate privileges, and move laterally throughout the environment (e.g., target directory service infrastructure to directly manipulate and control the authentication and authorization system). At this point, the adversary completed all actions to establish persistent access to the targets. While the initial footholds were used to harvest legitimate credentials for pivoting and systematic takeover of IT systems and remote connections, it is likely that the attackers moved quickly away from their initial footholds and vulnerable C2s in an effort to blend in to the target's systems as authorized users. With this information, the attackers would be able to identify VPN connections and avenues from the business network into the ICS network. Using native connections and commands allows the attackers to discover the remainder of the systems and extract data necessary to formulate a plan for Stage 2.

During the ICS Attack Stage, the adversaries used native software to Deliver themselves into the environment for direct interaction with the ICS components. They achieved this using existing remote administration tools on the operator workstations. The threat actors also continued to use the VPN access into the IT environment. In final preparation for the attack, the adversaries completed the Install/Modify stage by installing malicious software identified as a modified or customized KillDisk across the environment. While it is likely the attackers then ensured their modifications to the UPS were ready for the attack, there was not sufficient forensic evidence available to prove this. The last act of modification was for the adversaries to take control of the operator workstations and thereby lock the operators out of their systems. Finally, to complete the ICS Cyber Kill Chain and to Execute the ICS Attack, the adversaries used the HMIs in the SCADA environment to open the breakers. At least 60 substations (the total number is probably higher) were taken offline across the three energy companies, impacting roughly 225,000 customers. Simultaneously, the attackers uploaded the malicious firmware to the serial-to-ethernet gateway devices. This ensured that even if the operator workstations were recovered, remote commands could not be issued to bring the substations back online (We have characterized the firmware attacks against field communication devices as "blowing the bridges"). During this same period, the attackers also leveraged a remote telephonic denial of service on the energy company's call center with thousands of calls to ensure that impacted customers could not report outages. Initially, it seemed that this attack was to keep customers from informing the operators of how extensive the outages were; however, in review of the entirety of the evidence, it is more likely that the denial of service was executed to frustrate the customers since they could not contact customer support or gain clarity regarding the outage.

Ukraine Power Outage



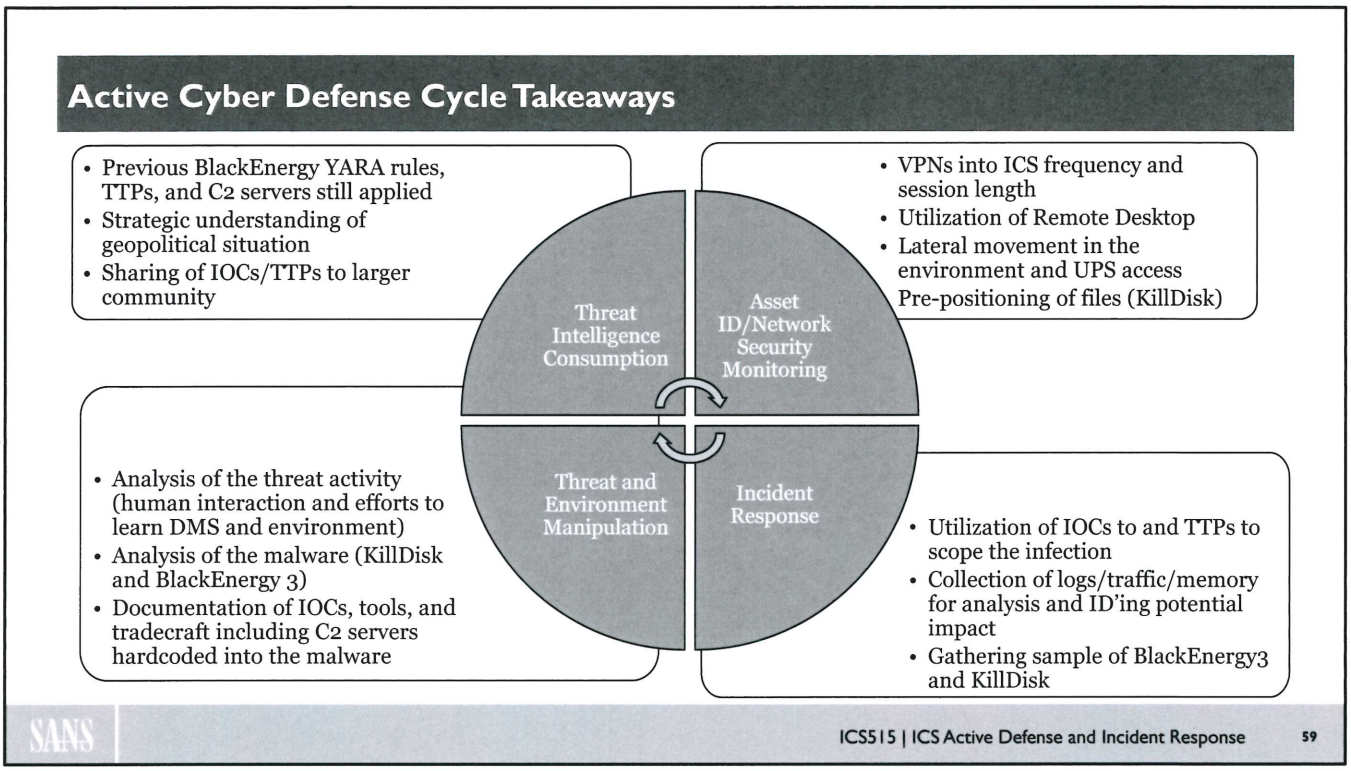
- Integrated Attack
- Highly Coordinated
- Logistic Sophistication
- Used Tools to Enable
- 6+ Months in the Environment

Full Ukraine Report:
http://ics.sans.org/duc5

You are here — Ukraine Incident

58

Ukraine Power Outage
Full Ukraine Report:
http://ics.sans.org/duc5



Active Cyber Defense Cycle Takeaways

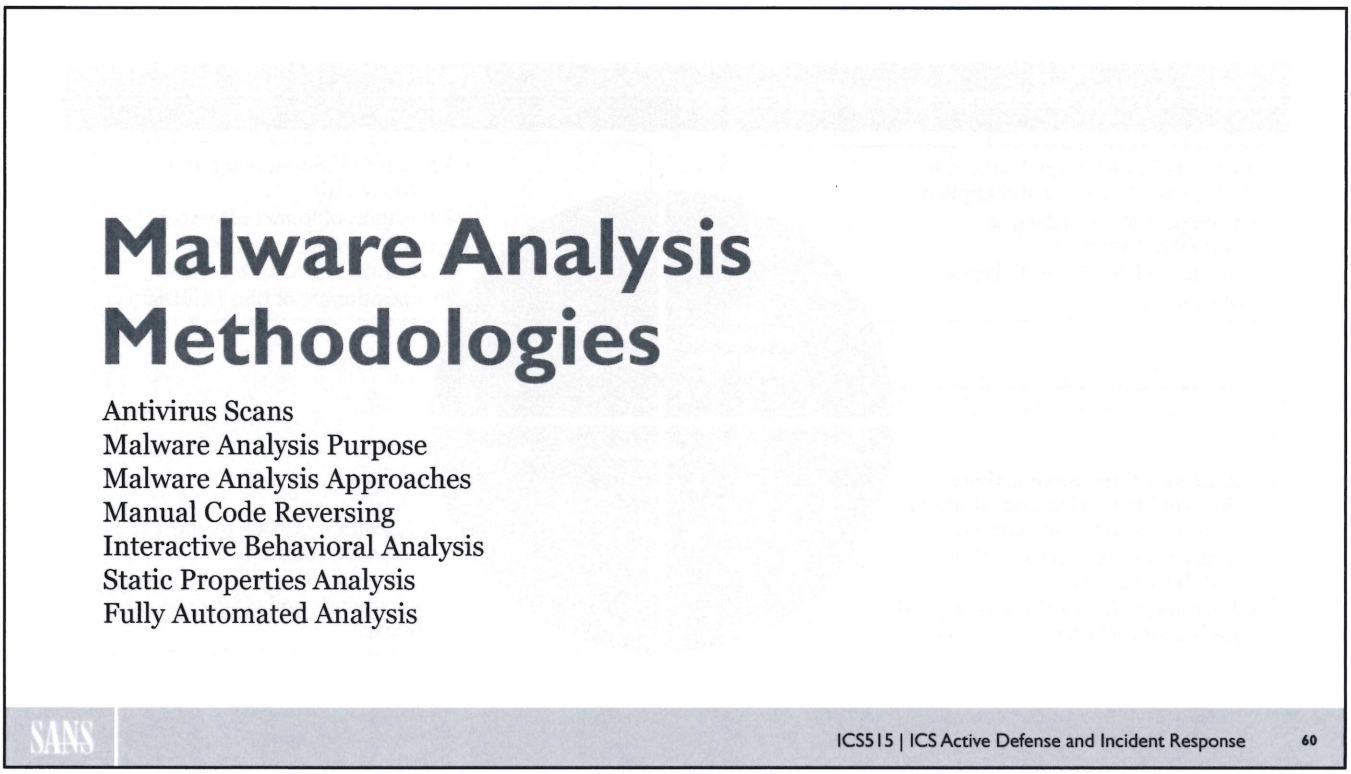
There were numerous takeaways from the Ukraine scenario along the Sliding Scale of Cyber Security—Everything from Architecture decisions such as the need for limiting remote desktop access and enabling two-form authentication on VPNs to Passive Defenses in the IT side of the house to detect abnormal activity and traversal into the ICS. The Active Defense takeaways are the most prominent, though, as with this style of targeted threat, the adversary would have worked to defeat the architecture and passive defenses in place. This is the perfect case study for showing what active defenders should have been doing against an active adversary.

The Threat Intelligence Consumption personnel could have taken the larger geopolitical situation into consideration to determine that there was a heightened security posture needed, especially in relation to physical conflict and sabotage against electrical equipment and stations in other parts of Ukraine and Crimea. They also could have identified what threat actors have shown interest in the energy supply before, such as the Sandworm team, and gathered the indicators, tradecraft, and YARA rules related to them. These were all still useful and relevant in the Ukraine facility.

This information could have been passed to the NSM analysts monitoring the environment. Monitoring the ICS particularly would have been useful as frequency and session length changes in VPN connections as well as odd utilization of remote desktop and the UPS could have helped identify abnormal activity. Additionally, log modifications on systems such as new software being deployed across the ICS would have been particularly worth seeing.

The incident responders could have been guided to the impacted systems quickly to gather samples and forensic evidence while quickly passing information back to the Threat and Environment Manipulation analysts to analyze out and extract knowledge about the adversary, the potential impact of the actions, and helping give recommendations for personnel to clean up and secure the environment. Even something as simple as disabling remote desktop assistance temporarily would have been an awesome recommendation.

All this information could be passed back to the Threat Intelligence Consumption personnel to drive the process numerous times in the face of the adversary and to share relevant information to the larger community, especially in Ukraine to help everyone's security posture get better.



This page intentionally left blank.

General Malware Analysis in ICS Environments

1. Establish a safe working environment (for example, use VMs)
2. Document received evidence files (such as phishing email, pcap, memory files, and so on)
3. Perform initial analysis to identify useful information (for example, pre-existing YARA rules)
4. Identify and investigate interesting files, executables, and such
5. Extract suspect file(s) from the evidence and document them including hashes
6. ***Run copies of the suspect files through in-house antivirus systems to identify matches:***
   a. ***If the suspect files have known AV signatures, it is known malware, and online research can take place, such as looking up digital hashes, antivirus company threat reports, and more***
   b. ***If the suspect file does not have a known AV signature, continue on (or, if at least some do not)***
7. ***Begin processing the suspect file(s) through an in-house automated malware analysis sandbox if possible***
8. ***Begin extracting information using behavioral malware analysis techniques***
9. Use the analysis to create IOCs and provide context while documenting findings
10. Pass IOCs, documentation, and so on to those managing the internal threat intelligence products

General Malware Analysis in ICS Environments

Most threats involve malware, and today's focus is heavily centered around learning from malware. Even threats that don't rely on malware to impact the ICS may rely on malware to gain a foothold in the ICS or corporate network. The same thought processes and type of analysis that goes into malware analysis is useful to understanding nonmalware-based threats.

To help guide the day, the 10-step process is used throughout the day. This is a general methodology and may not represent the abilities and processes of every ICS security team, but it covers the vast majority of cases.

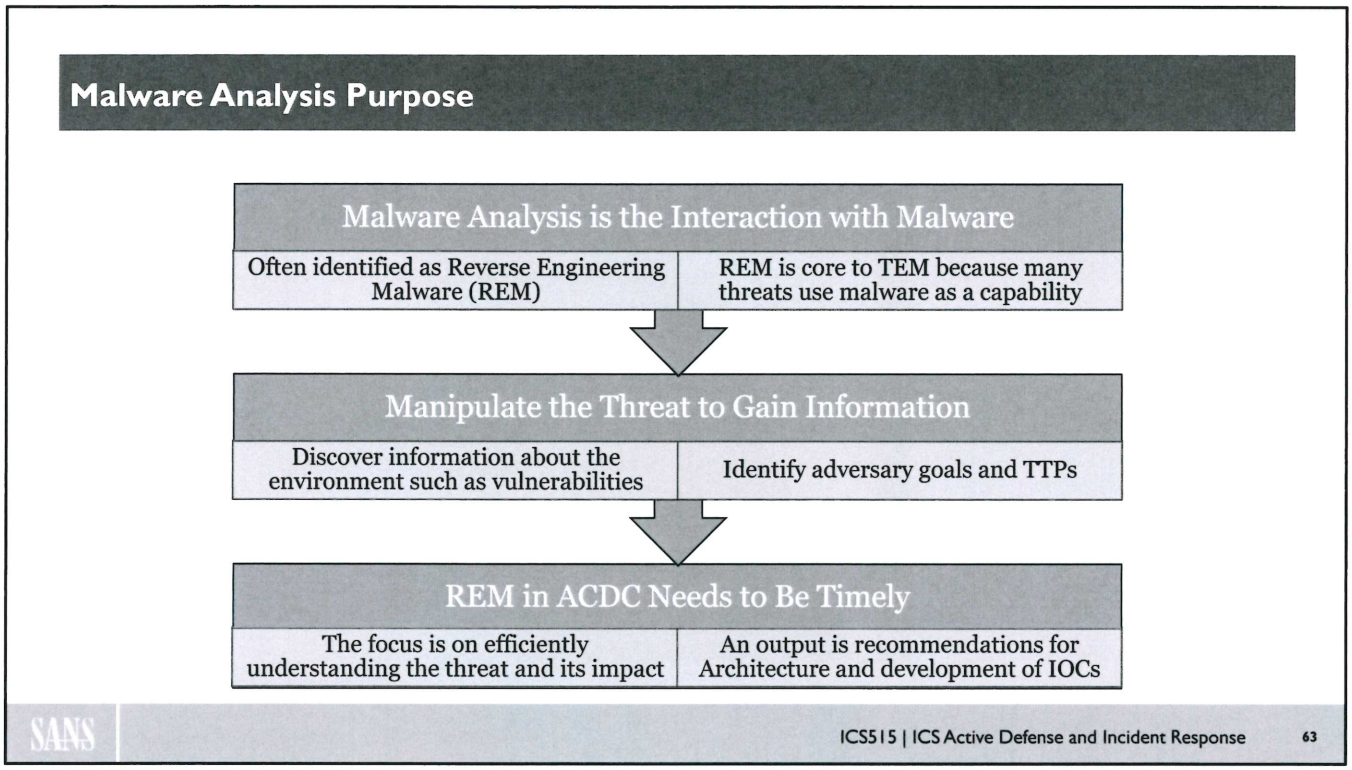
Throughout the 10-step process, analysts should think about what can be done to the architecture to deny or delay the adversary and its capabilities.

Antivirus Scans

- After suspect files have been identified and extracted, it is useful to run copies of the files through in-house antivirus software
- Analysts often have 2+ VMs that each have a single-version antivirus on it:
  - For example, one VM with Symantec, one VM with Kaspersky, one VM with F-Secure
  - This tactic is known as having an antivirus farm and is what websites such as VirusTotal do
- The importance of doing this specifically in ICS is the need to keep files/evidence secret:
  - ICS attacks are more likely to be new, advanced, and impactful; submitting online alerts the adversary
  - Running copies ensures that the evidence isn't destroyed but can return valuable information
- If the antivirus report comes back positive with a known piece of malware, research it:
  - Look up online for the digital hash or malware name of the identified threat, but do not use your identified files
  - In addition, some files or modifications to the malware could be unique; adversaries have been known to build on widely available malware because it reduces chances for attribution and decreases barrier to entry

Antivirus Scans
Once files or processes have been identified as potentially malicious (such as the .dmp files from the Stuxnet memory sample), it is good to run them through antivirus software to see if the malware is already known and if there's any information about it. Sometimes, AV comes back positive only because of heuristics, though, and not signatures. Be very careful to note the difference by understanding the outputs of the tool and the reason for the alerts.

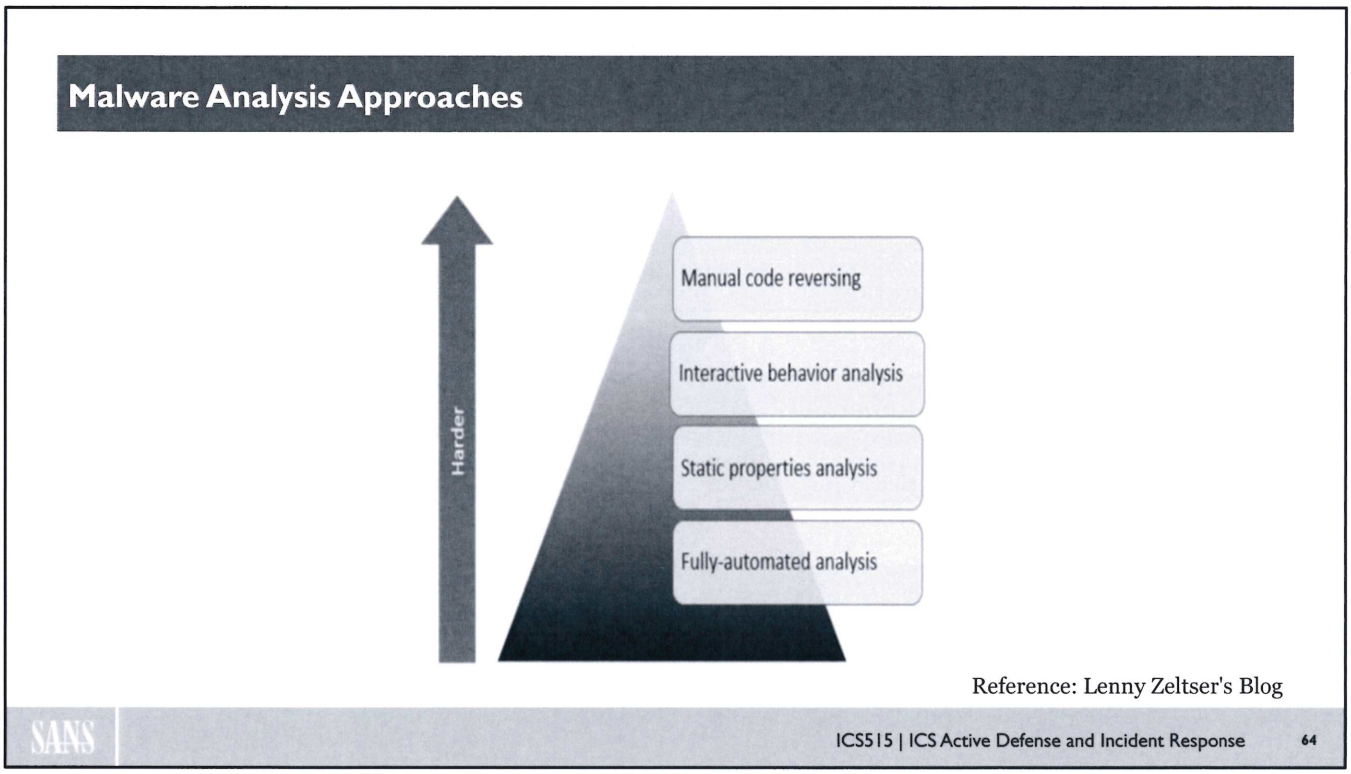
Antivirus Scans Outputs
Submitting the malware to antivirus that is connected to the Internet may send information back to the AV companies and potentially alert adversaries because AV companies will then update their signatures. Be sure you have considered confidentiality issues, and risk of alerting adversaries before you utilize malware upload sites. Thus, an AV farm is a good choice for ICS environments.



Malware Analysis Purpose
The purpose of malware analysis is to manipulate the threat to gain as much information as possible about it that is useful for ACDC in a timely manner.

Malware analysis allows the deep understanding of malware and generally the overall threat. It helps to identify information useful to protecting systems such as the discovery of vulnerabilities that were previously unknown (or not prioritized), and it allows analysts to understand the tactics, techniques, and procedures of the adversary as well as the sophistication. It also helps identify adversary goals and understand what systems/information are likely to be targeted or impacted.

Malware analysis is simply interacting with and understanding malware and is often identified as Reverse Engineering Malware (REM). There are a few general approaches and methodologies that we will go over in the next slide. REM and deep malware analysis are great to know how to do but aren't necessarily part of ACDC; ultimately, that depends on your goals and requirements, but initial triage and analysis are perfect to get through the ACDC quickly and repeat the process until the threat is expelled. In ACDC, the ability to perform malware analysis drives the development of threat information, such as IOCs and context key to internal threat intelligence and actionable information for network security monitoring and incident response.



Malware Analysis Approaches
As stated earlier, Lenny is a malware analysis guru, and these are his four approaches to malware analysis. Generally, ACDC analysts will focus on fully-automated analysis, static properties analysis, and potentially interactive behavior analysis. Manual code reversing is often too time consuming and cumbersome for the ACDC process, although it can be of value when done.

Reference:
http://blog.zeltser.com/post/79453081001/mastering-4-stages-of-malware-analysis

Manual Code Reversing

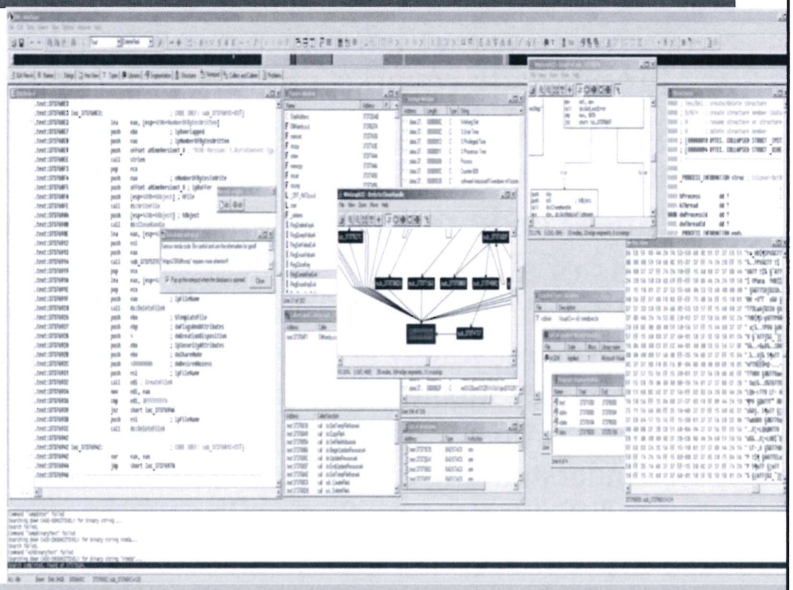
- Manual code reverse-engineering occurs when analysts manually examine, decode, and deconstruct malicious code to gather information
- Manual code reversing can gather information not feasible in other forms of analysis:
  - For example, behavioral analysis performed by running the malware in a safe environment may not reveal all the capabilities of the malware itself. For example, Stuxnet had multiple exploits but used only specific ones on specific operating systems
  - Understanding cryptographic routines, algorithms, and obfuscation practices is difficult outside of manual reversing

Manual Code Reversing

Not a lot of ICS teams will have the dedicated personnel to perform manual code reversing. This is generally what people think about when they hear about malware analysis. Manual code reversing is the manual examination of the code through decoding it, deconstructing it, and analyzing the functions and capabilities of the malware. Lots of useful information can be gained this way (this is the only way to understand all the capabilities of a piece of malware consistently), but it is time consuming. Even the Stuxnet malware was initially evaluated with interactive analysis and not manual code analysis.

Disassemblers and Debuggers

- To perform manual code reversing, most analysts use a Debugger and Disassembler
- Disassembler:
  - Takes binary files and translates the code into human readable assembly code
- Debugger:
  - Displays the human readable assembly code and steps through the code showing how it is executed
  - Reveals function calls, variables executed when and why, and allows analysts to set break points to interact with code



SANS ICS515 | ICS Active Defense and Incident Response 66

Dissemblers and Debuggers
Manual code reversing often requires a debugger and a disassembler. The Disassembler takes the binary files and translates them into "human readable" (although not friendly) assembly code. The Debugger helps step through that disassembled code to understand what is going on.

It is not recommended for you to do the manual analysis during incident response. We need much more timely information than this.

IDA: Disassembler and Debugger
When people get worried about malware analysis, it is usually because they see screens like this one. Needless to say, we will not be performing this in class, but it's not as hard as it looks; you just have to learn a bit of assembly and practice a lot. But there are easier and faster ways to go about malware analysis. It is worth noting that to truly understand a threat and all its capabilities, this is the way to go.

Reference:
https://www.hex-rays.com/products/ida/

Interactive Behavior Analysis (Dynamic)

- Interactive behavior analysis (also sometimes referred to as dynamic analysis) is the process of executing malicious files and observing their interactions on the system including processes spawned, files modified/created/accessed, Registry keys manipulated, network traffic, and so on
- This is the area most ACDC personnel spend their time for malware analysis to understand threats that are relevant to ICS and what impacts they might have on the ICS
- Common tools include:
  - Windows Sysinternals (suite of tools such as ProcMon, Proc Explorer, AutoRuns, and TcpView) for system analysis
  - Wireshark and tcpdump to monitor network communication
  - REMnux for a variety of tools including the ability to set up fake DNS servers for interaction with the malware

Interactive Behavior Analysis (Dynamic)
IBA used to be identified as dynamic analysis, but terminology adapted over time. The general approach is to set up an analysis system and let the malware execute on the system while watching its behavior. You will not learn everything there is to know about the malware, but you'll gain the most important information useful for NSM and IR.

Note: Malware can take a while to fully execute, so be prepared to let it run for awhile.

Common Strategy:

1. Save the current state of the VM
2. Open monitoring tools such as Wireshark and ProcMon
3. Introduce and execute a malicious file into the environment
4. Observe the changes on the system with available tools
5. Post analysis, pause the system, and revert to a clean image

Reference:
http://technet.microsoft.com/en-us/sysinternals/bb545021.aspx
Good read: http://irhowto.wordpress.com/2010/04/15/quick-and-dirty-malware-analysis/
Great presentation: https://www.blackhat.com/presentations/bh-dc-07/Kendall_McMillan/Presentation/bh-dc-07-Kendall_McMillan.pdf

Static Properties Analysis

- Static properties are those that can be examined for malware without deep analysis/execution:
  - For example, a digital hash is a static (or atomic) property of a file
  - Detecting if the file is packed (or a program to modify/obfuscate malware and its contents)
  - Portable Executable (PE) file formats can give information on data and time of compilation as well as information about functions that are imported or exported by the file to the system
- Useful for quickly creating IOCs or establishing blacklists

Static Properties Analysis
These are types of information you can gather without executing the malware. It's the quickest way to get to useful information, but it does not reveal as much as the other forms of analysis. When we did the PDF analysis, some of those methods were static properties analysis (pulling out strings, identifying metadata, getting a digital hash, and so on).

Static Properties Analysis Common Tools

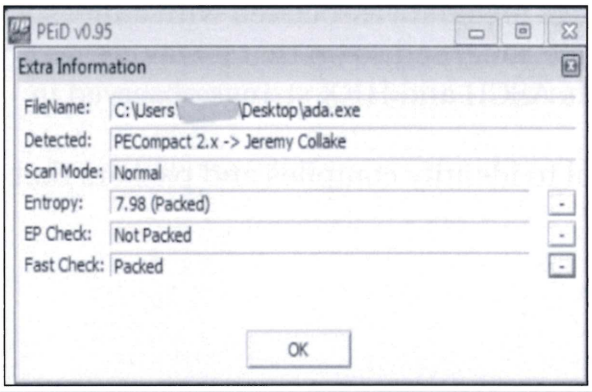
- Common tools for static properties analysis:
  - **PEstudio:** Identifies functionality of a file based on PE analysis and indicates if it's malicious, based on preset rules
  - **ExifTool:** Extracts metadata associated with a file, such as the file path/location/time zone/and so on that it was created
  - **Strings:** Extracts ASCII and HEX strings observed in a file
  - **PEiD:** A free tool to identify compiles and packers that have been used on a file

Static Properties Analysis Common Tools
Many of these tools are available in the REMnux distribution you have for the class. They are tools that view the piece of malware and extract information without actually executing or interacting with the malware.

Good reading on PE Analysis: https://blog.malwarebytes.org/intelligence/2014/05/five-pe-analysis-tools-worth-looking-at/

Example: PEiD

- Hun-Ya Lock wrote a great GREM Gold paper for the SANS Reading Room titled "Using IOCs in Malware Forensics"—it is worth reading
- Here is an excerpt showing a screen shot of PEiD and how it can detect packers:



Example: PEiD
Here is a good example of PEiD in use as shown by Hun-Ya Lock for his SANS Gold paper. Notice how the tool was pointed at the file ada.exe, and PEiD detected that it was packed. It looked at a variety of methods including a fast check and an entropy check. Entropy checks attempt to determine the randomness of code. If code is extremely random, it's probably obfuscated or packed because human readable code contains a lot of patterns. PEiD determined that this was packed by PECompact 2.x, which was made by Jeremy Collake.

Reference:
http://www.sans.org/reading-room/whitepapers/forensics/ioc-indicators-compromise-malware-forensics-34200

Fully Automated Analysis

- Provide quick assessments of malware by running it in a virtual environment and recording details such as processes created, files dropped, Registry keys manipulated, and network traffic:
  - Not as detailed as performing analysis by hand
  - Does not require as much malware knowledge/training by analyst
- Fully automated analysis usually comes in two forms (in-house versus online)
- Online (off network but still free):
  - Automated malware analysis sites
- In-house:
  - Custom sandboxes

Fully Automated Analysis
Especially for small ICS security teams, automated malware analysis is important. It's useful to everyone, but the more automation you can use, the more powerful your analysts will be. Online analysis is doable, but in ICS environments, in case you run across specialized threats, it's better to do it in-house by setting up your own sandboxes (which is all the online sites do).

Outsourced usually offers more speed and ease of use. In-house offers more privacy and detailed analysis.

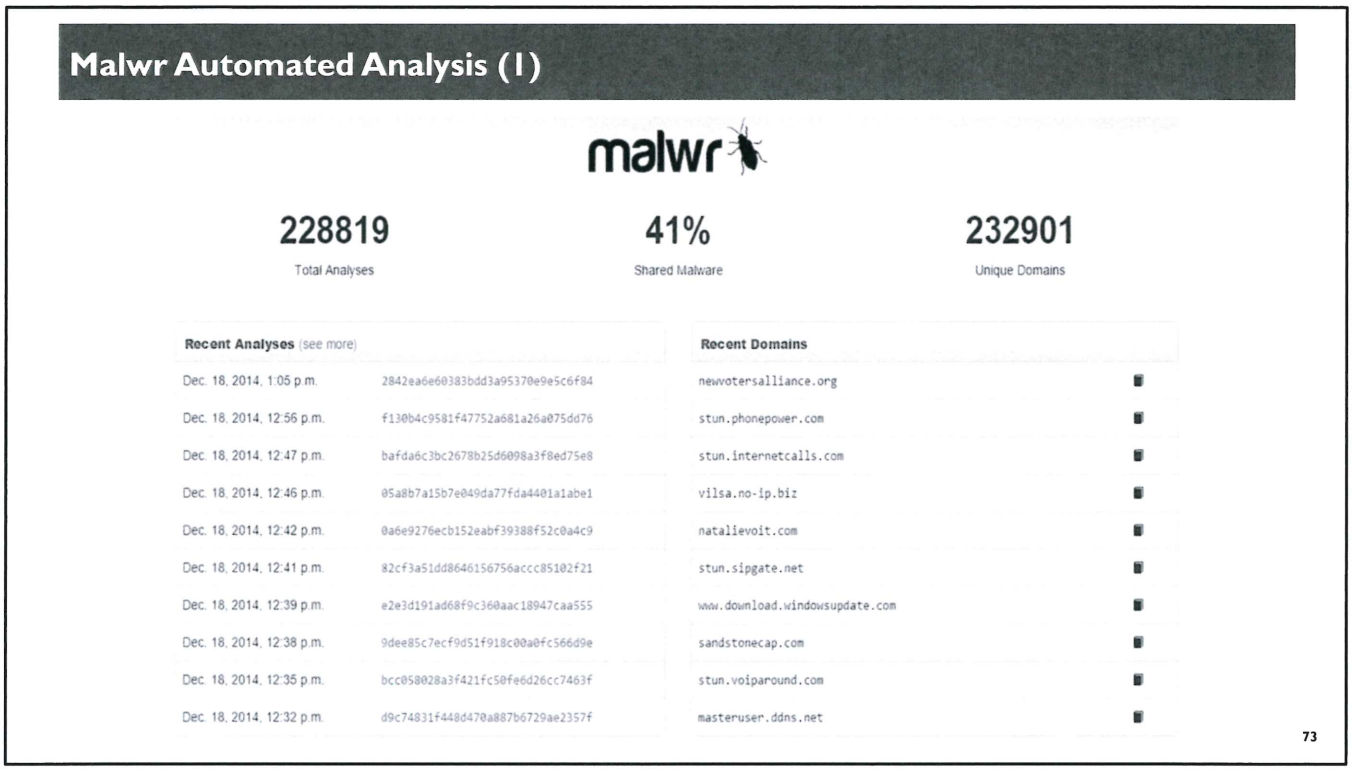
Custom means having the ability to configure it to architecture, settings, and software you have in your environment. That is, you could install an HMI on it if you want.

Automated Malware Analysis Sites

- Multiple websites exist where files, URLs, and digital hashes can be submitted to determine whether the file is malicious:
  - These sites automatically analyze the malware by running them in sandboxes or comparing them to previously obtained samples
- A few common websites:
  - VirusTotal.com
  - Malwr.com
  - Hybrid-analysis.com
- The biggest problem and concern with automated malware analysis sites for ICS environments is that adversaries who are performing advanced campaigns or targeted attacks monitor these sites' submissions

Automated Malware Analysis Sites

Here are some sites to use if you cannot set up something internal to your organization and your concern is less related to tipping the adversary about your analysis and more about protecting the ICS in a quick manner. These are not generally recommended for anything important, but if it looks like a common threat or accidental infection, these can be great. You can also look at recent submissions to see whether anyone has already submitted the threat you are facing; maybe the analysis was already done and/or you have partners in defense.



Malwr Automated Analysis (1)
Here is the web page for malwr, which shows recent analyses performed (clicking the hashes brings up the reports) and recent domains (used by many organizations for blacklists).

Malwr Automated Analysis (2)

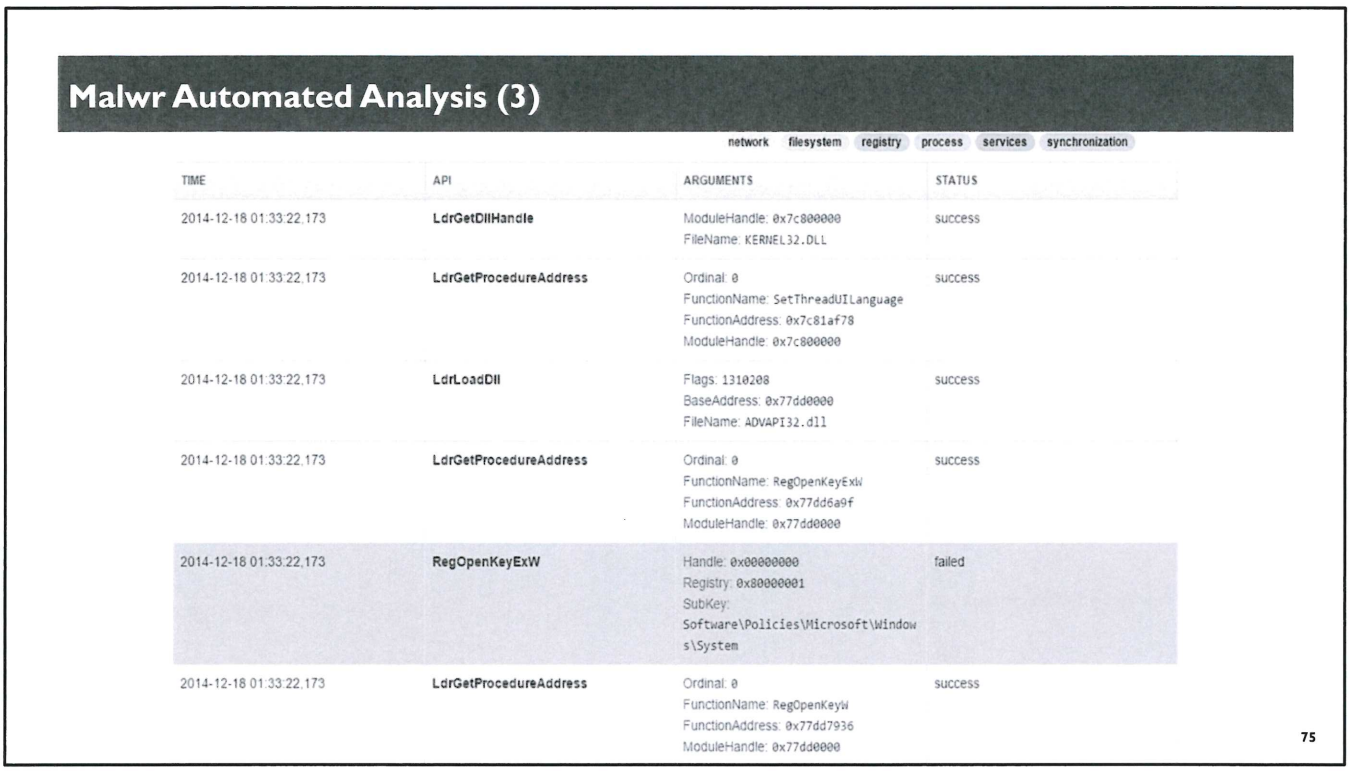
File Details

FILE NAME	Patch.exe
FILE SIZE	518753 bytes
FILE TYPE	PE32 executable (GUI) Intel 80386, for MS Windows
MD5	05a8b7a15b7e049da77fda4401a1abe1
SHA1	7c73bca2b0ef90cbbbb06cf5408841e8a37228b3
SHA256	b4bf7b5dfdc8202b402e22d56b5715cf7a90f0bd3eda5510ff1de432208a9025
SHA512	26e3f894a822cf5c92e59f8e925efabdeabf5da95da921b34f40dfbdcafa4fbe04add77e6bee4ad5220aad107b725433874b87c42c63e606f82a13b796f26cbe
CRC32	7E3BBEEC
SSDEEP	12288:8vuK8XOlKRBh/4gBdyVs7pGR+64SVe/X86R4Dpx1N:8v9GOlK7h/4gr4JB6xiDpB
YARA	None matched

74

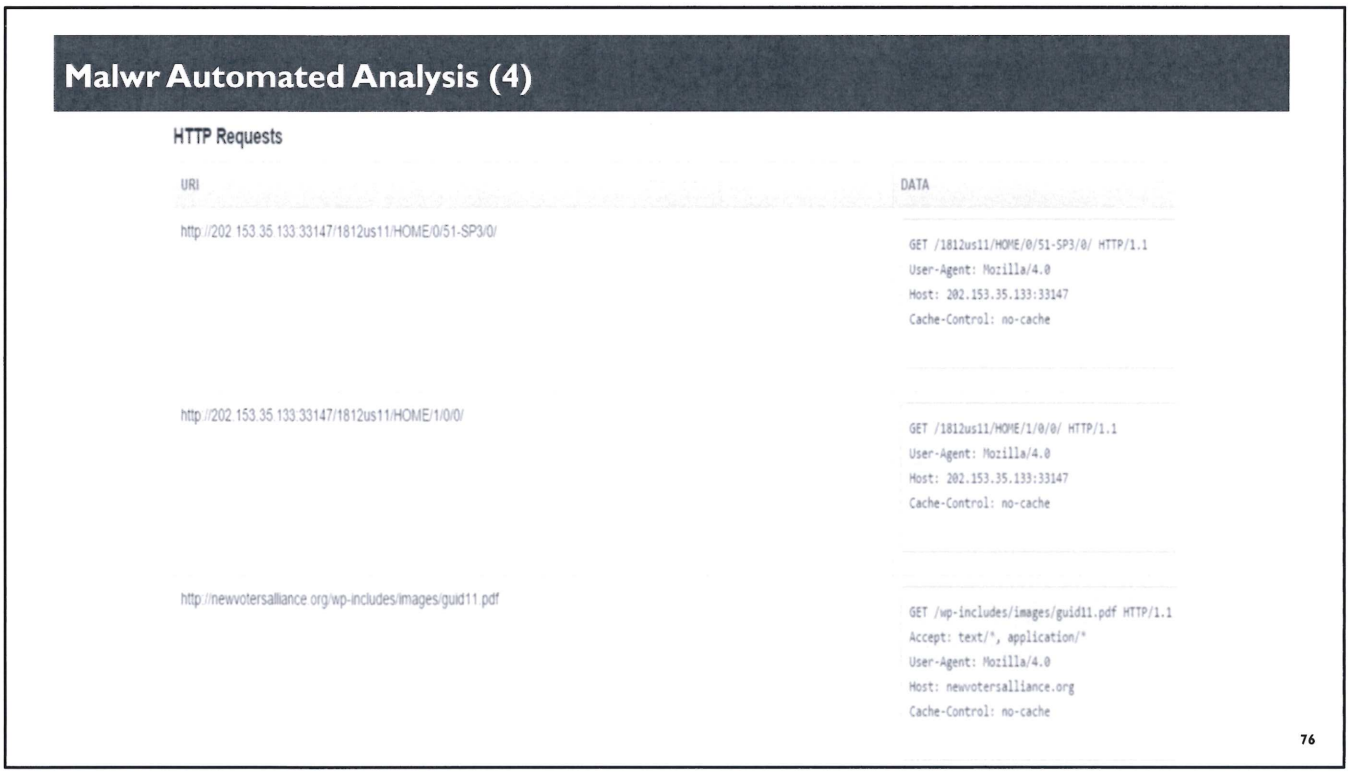
Malwr Automated Analysis (2)

Here is an example of a report that was generated; this is one portion of a report based on a Windows executable that someone thought might be malicious; it was called patch.exe.



Malwr Automated Analysis (3)

The report continues on to show some of the API calls that were made in the malware and different impacts they had on the system, such as Registry key manipulation or installing files. Notice that the red highlighted frame is because the function failed. In automated analysis, systems are not necessarily configured in a way that malware likes, so it is either denied or won't finish its routine. This shows one limitation of automated malware analysis.



Malwr Automated Analysis (4)

This part of the report shows some of the network calls that were made. We see at the bottom, for example, that the malware was trying to pull down a PDF file. For ACDC purposes, the TEM team could perform this analysis and make IOCs for the NSM personnel to make IDS signatures. If you do not have a copy of the PDF and it's not on your network, it may be tempting to go ahead and let the malware connect in a safe environment and pull it down. This is generally not advisable for all the reasons we've touched on as well as alerting the adversary that your network is infected and worth further investigation.

Sandboxes

- A specific type of virtualization that enables programs to be run independently in a controlled set of resources, thus protecting the host system from infection:
  - Allows an analyst to safely run and analyze untrusted code or malware
  - If malware somehow breaks out of the sandbox, it is still contained in the virtual machine protecting the hardware
- The automated analysis websites online use sandboxes to provide their analysis:
  - The benefit of making your own sandbox is the ability to create a private and customizable solution
- Noteworthy open source sandboxes:
  - Cuckoo Sandbox
- Noteworthy paid services:
  - Payload Security's Hybrid-Analysis service

Sandboxes
Sandboxes are a specific implementation of virtualization. At the bottom of the slide, you can see two specific types of sandboxes that come preconfigured with scripts and analysis tools.

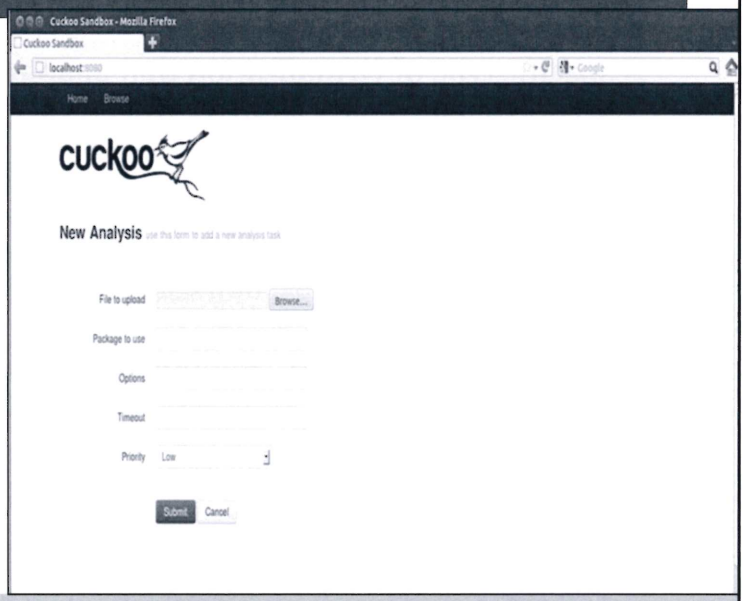
References:
http://blog.zeltser.com/post/1284687696/malware-analysis-tool-frameworks
http://www.cuckoosandbox.org/

Recommended reading:
http://cert.at/static/downloads/papers/cert.at-mass_malware_analysis_1.0.pdf
https://www.syssec.rub.de/media/emma/veroeffentlichungen/2012/12/14/CWSandbox-IEEESP2007.pdf

Recommended books:
Cuckoo Malware Analysis by Digit Oktavianto and Iqbal Muhardianto
Practical Malware Analysis by Michael Sikorski and Andrew Honig

Cuckoo Sandbox

- Automatically runs the malware and returns reports that include:
  - Memory dump of the sandbox for memory analysis
  - Screen shots of the environment when the malware ran
  - Networking information such as DNS and C2 requests
  - Files and processes created, deleted, downloaded, and modified
- Based on Python and initially can be time-consuming to set up but then provides fast analysis:
  - Also can be tailored to automatically produce strings and information for YARA signatures or use your YARA rules



Cuckoo Sandbox
The Cuckoo Sandbox is a great choice, but it can be a task to set up. If you're just starting with sandboxes, I'd recommend Zero Wine (shown in a few slides) but Cuckoo is the better choice overall, especially for customization.

It is an automatic analysis platform and can be scripted to extend its features for a powerful asset for any ICS security team. Memory dumps, YARA signatures, networking options, etc. make it very powerful and flexible.

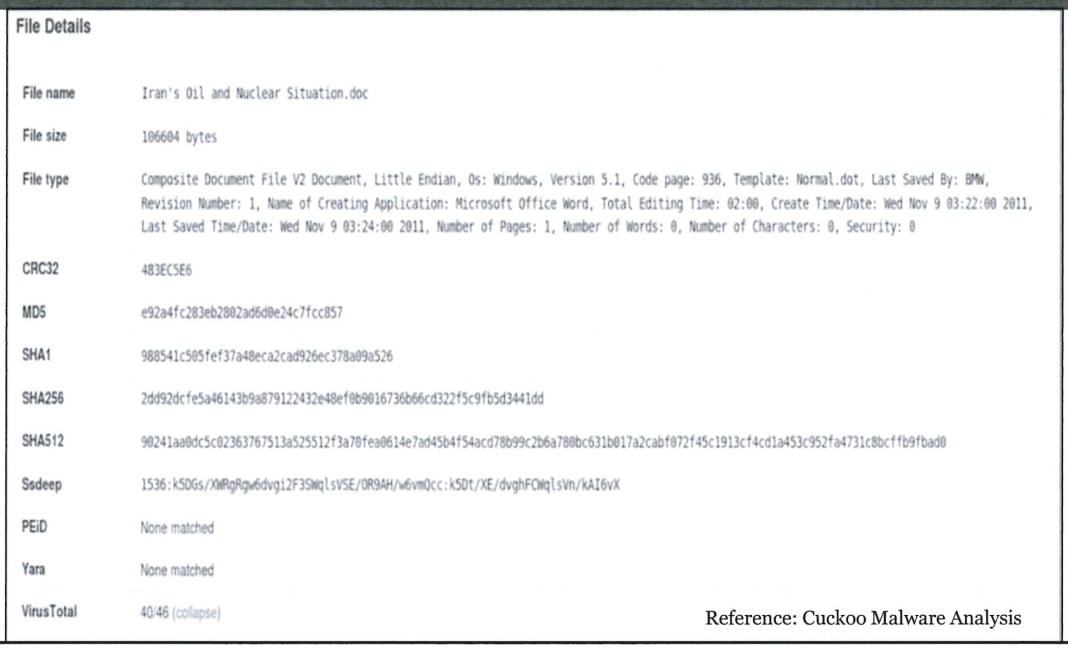
Cuckoo natively runs a number of extensions, such as:

- Windows executables
- DLL files
- PDF documents
- Microsoft Office documents
- URLs
- And various other types of scripts and files

Here is a useful guide to setting up Cuckoo: http://www.behindthefirewalls.com/2013/07/how-to-install-cuckoo-sandbox-on-ubuntu.html

Cuckoo Sandbox Interaction Post-Installation
After Cuckoo is set up, it provides a GUI on the localhost to submit files and information. For example, you can set priorities around malware so that you can submit any number of samples into a queue. When an interesting piece of malware comes in, you can set a high priority that jumps it to the front of the line.

Cuckoo Sandbox Sample Output



File Details

File name	Iran's Oil and Nuclear Situation.doc
File size	106604 bytes
File type	Composite Document File V2 Document, Little Endian, Os: Windows, Version 5.1, Code page: 936, Template: Normal.dot, Last Saved By: BMW, Revision Number: 1, Name of Creating Application: Microsoft Office Word, Total Editing Time: 02:00, Create Time/Date: Wed Nov 9 03:22:00 2011, Last Saved Time/Date: Wed Nov 9 03:24:00 2011, Number of Pages: 1, Number of Words: 0, Number of Characters: 0, Security: 0
CRC32	483EC5E6
MD5	e92a4fc283eb2802ad6d0e24c7fcc857
SHA1	988541c505fef37a48eca2cad926ec378a09a526
SHA256	2dd92dcfe5a46143b9a879122432e48ef0b9016736b66cd322f5c9fb5d3441dd
SHA512	90241aa0dc5c02363767513a525512f3a70fea0614e7ad45b4f54acd78b99c2b6a780bc631b017a2cabf072f45c1913cf4cd1a453c952fa4731c8bcffb9fbad0
Ssdeep	1536:k5DGs/XWRgRgw6dvg12F3SWqlsVSE/OR9AH/w6vmQcc:k5Dt/XE/dvghFCWqlsVn/kAI6vX
PEiD	None matched
Yara	None matched
VirusTotal	40/46 (collapse)

Reference: Cuckoo Malware Analysis

79

Cuckoo Sandbox Sample Output
Here, we see a sample output from a document that was run titled, "Iran's Oil and Nuclear Situation.doc." This document matched VirusTotal samples from 40 out of 46 antivirus companies. Cuckoo did not connect to the Internet for this information, though. It can be updated with VirusTotal samples to use offline, making it a great and safe choice for ICS analysts.

A lot of other information is available, such as a packet capture for analysis in Wireshark.

When Automated Malware Analysis Fails

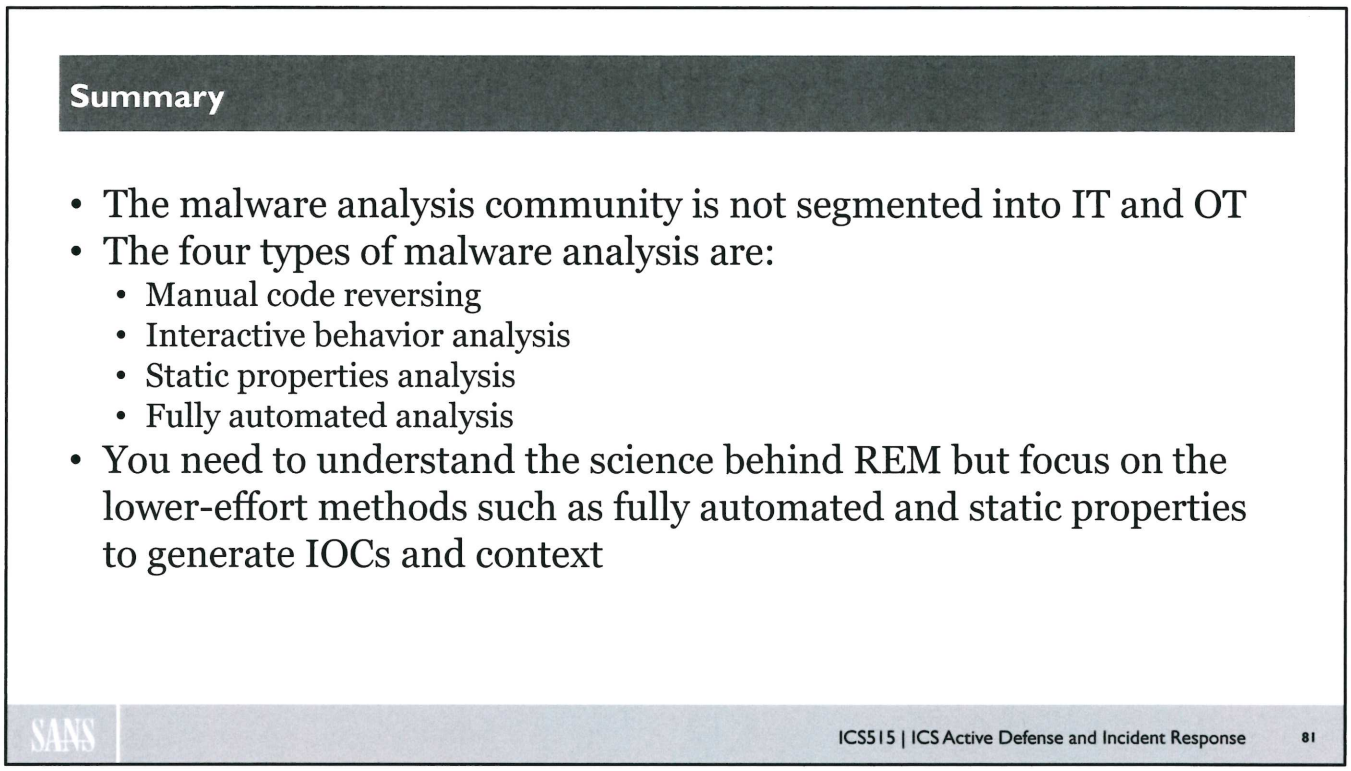
- Automated malware analysis can fail due to a variety of reasons but not limited to:
  - The malware detects it is in a VM and does not execute properly
  - The sandbox has errors or false positives due to the nature of the malware
  - The malware executes only certain ways on specific systems, and thus automated analysis may miss capabilities
  - The malware waits a long time to execute and thus times out the sandbox
- When errors are introduced and automated malware analysis fails, it's good to perform statistical and behavioral analysis to quickly derive information for defense efforts

When Automated Malware Analysis Fails
VM defeating malware seemed to provide a problem for malware analysts when it first came out. Malware that could detect if it were run in a VM essentially made analysis more difficult. However, because of the popularity of virtualization in most large environments and due to the cloud, malware cannot afford to not run on VMs anymore. It's possible to see something like this in an ICS environment, so keep in mind that this may be one of the reasons the malware isn't running in a sandbox.

Some recommended information from Symantec to disguising the fact that your analysis platform is a VM:

- Check the MAC address of the virtual network adapter to reveal the vendor.
- Check certain registry keys that are unique to virtual systems.
- Check to see if helper tools such as VMware tools are installed.
- Check for certain process and service names.
- Check for communication ports and behavior.
- Execute special assembler code and compare the results.
- Check the location of system structures, such as the descriptor tables.

Reference:
http://www.symantec.com/connect/blogs/does-malware-still-detect-virtual-machines



Summary
The malware analysis community really doesn't care about the malware being IT-based or OT-based because there really aren't a lot of differences. ICS will get different types of threat groups and actors, but the types of malware used and the way to do analysis against them are largely the same.

In ICS environments, due to various considerations (such as manning and time), you need to understand the science behind malware analysis but focus on the lower-effort methods such as fully automated and static properties to generate IOCs and context:

- Interactive behavior analysis should most certainly be performed on threats that have not been observed before
- Manual code reversing comes into play eventually, usually through outsourced teams, but it is not timely

Malware analysis is an area that ICS doesn't need to try to re-create, but instead, contribute outliers and interesting case-studies/best-practices to the community.

A few names come up over and over again in the REM community, and they are worth looking into if you are interested in this area. These are Lenny Zeltser and Jake Williams.

Course Scenario

This page intentionally left blank.

Traffic Lights Network – Static Properties Analysis

The Traffic Lights network personnel performed static properties analysis on the malware sample to identify its hash and to determine when it was compiled by the adversary and what API calls it made

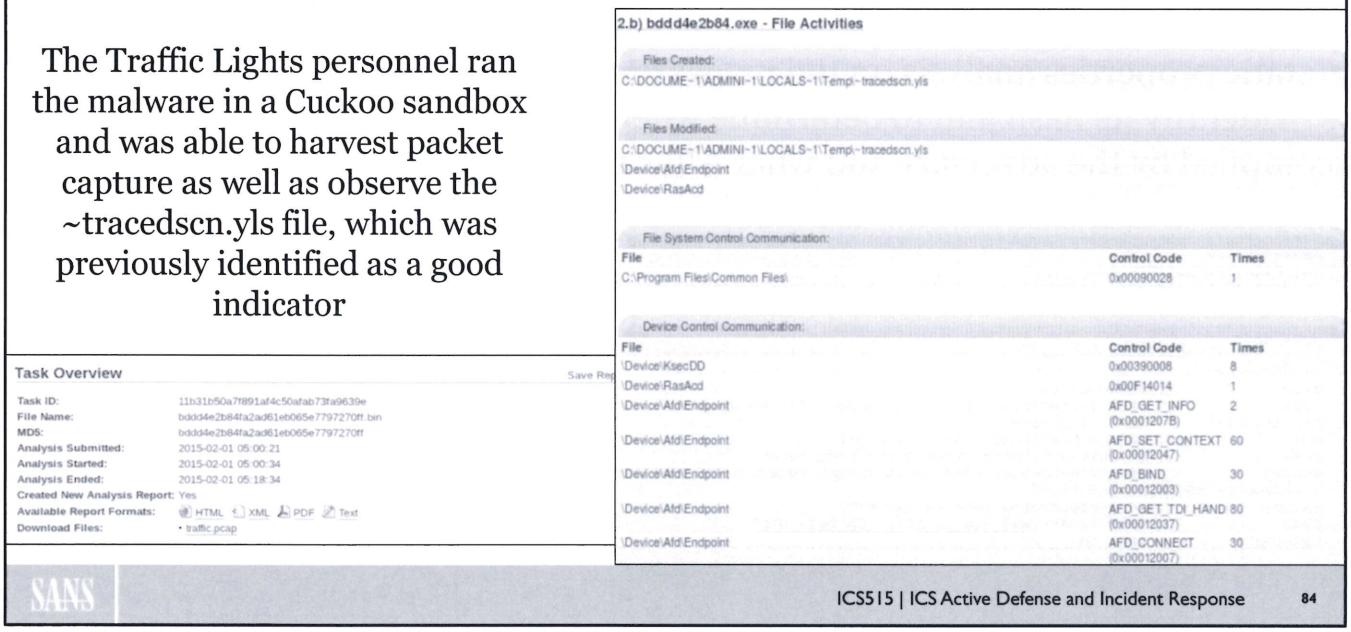
```
remnux@remnux:~/Desktop/ICS515/WARNING/Executable$ pescanner Executable.exe
################################################################################
############
[0] File: Executable.exe
################################################################################
############

Meta-data
================================================================================
============
Size            : 48640 bytes
Type            : PE32 executable (GUI) Intel 80386, for MS Windows
Architecture    : 32 Bits binary
MD5             : bddd4e2b84fa2ad61eb065e7797270ff
SHA1            : b4e657dbe1b1568fafa2595e559507979b17acab
ssdeep          : 768:8b7ovPiB3pqLIcfxkfKHmwW623Bx6uLAOhwsL3kUYPk6ED0Xn8Om/1o:
8fu83pqcoqfOmG23Bx6ukCn3vYP98F
imphash         : 14a84ed6cd7abb9fe0533d2a5ac3076f
Date            : 0x527A27AA [Wed Nov  6 11:27:38 2013 UTC]
Language        : ENGLISH
```

```
[+] Following expected Malware APIs are Detected


    [-] Import Table

            IA: 0x00409134  closesocket
            IA: 0x0040913c  connect
            IA: 0x0040903c  CreateThread
            IA: 0x00409004  GetFileSize
            IA: 0x00409034  GetProcAddress
            IA: 0x004090f0  GetTickCount
            IA: 0x004090a4  GetModuleFileNameA
            IA: 0x00409088  IsDebuggerPresent
            IA: 0x0040911c  Sleep
            IA: 0x00409140  socket
            IA: 0x0040900c  WriteFile
            IA: 0x00409150  WSAStartup

    [-] Entire Executable

              1 times       CreateFile
              1 times       CreateThread
              1 times       GetCommandLine
              1 times       GetFileSize
              1 times       GetModuleHandle
              1 times       GetProcAddress
              1 times       GetTempPath
              1 times       GetTickCount
              1 times       GetModuleFileNameA
              1 times       IsDebuggerPresent
              1 times       LoadLibrary
              1 times       Sleep
              1 times       WriteFile
```


This page intentionally left blank.

Traffic Lights Network – Sandbox Analysis

The Traffic Lights personnel ran the malware in a Cuckoo sandbox and was able to harvest packet capture as well as observe the ~tracedscn.yls file, which was previously identified as a good indicator



2.b) bddd4e2b84.exe - File Activities

Files Created:

C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\~tracedscn.yls

Files Modified:

C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\~tracedscn.yls
\Device\Afd\Endpoint
\Device\RasAcd

File System Control Communication:

File	Control Code	Times
C:\Program Files\Common Files\	0x00090028	1

Device Control Communication:

File	Control Code	Times
\Device\KsecDD	0x00390008	8
\Device\RasAcd	0x00F14014	1
\Device\Afd\Endpoint	AFD_GET_INFO (0x0001207B)	2
\Device\Afd\Endpoint	AFD_SET_CONTEXT (0x00012047)	60
\Device\Afd\Endpoint	AFD_BIND (0x00012003)	30
\Device\Afd\Endpoint	AFD_GET_TDI_HAND (0x00012037)	80
\Device\Afd\Endpoint	AFD_CONNECT (0x00012007)	30

Task Overview

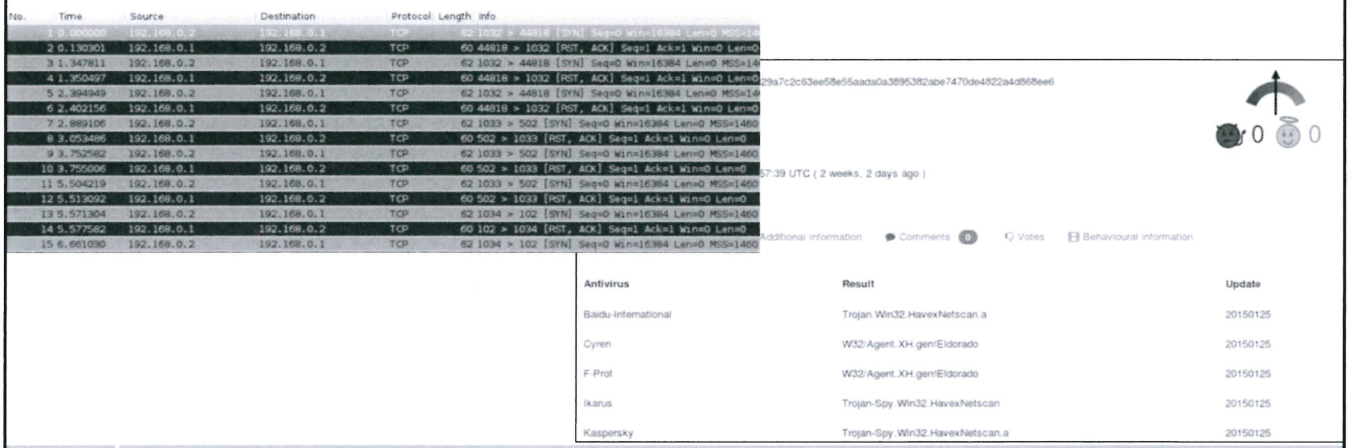
Save Rep

Task ID:	11b31b50a7f891af4c50afab73fa9639e
File Name:	bddd4e2b84fa2ad61eb065e7797270ff.bin
MD5:	bddd4e2b84fa2ad61eb065e7797270ff
Analysis Submitted:	2015-02-01 05:00:21
Analysis Started:	2015-02-01 05:00:34
Analysis Ended:	2015-02-01 05:18:34
Created New Analysis Report:	Yes
Available Report Formats:	HTML XML PDF Text
Download Files:	• traffic.pcap

This page intentionally left blank.

Traffic Lights – AV Sandbox and Network Analysis from the Sandbox

The AV Farm and the network capture from the sandbox helped confirm previously identified activity. This malware is the Havex malware and does connect to ICS ports, thus connecting the observables previously identified

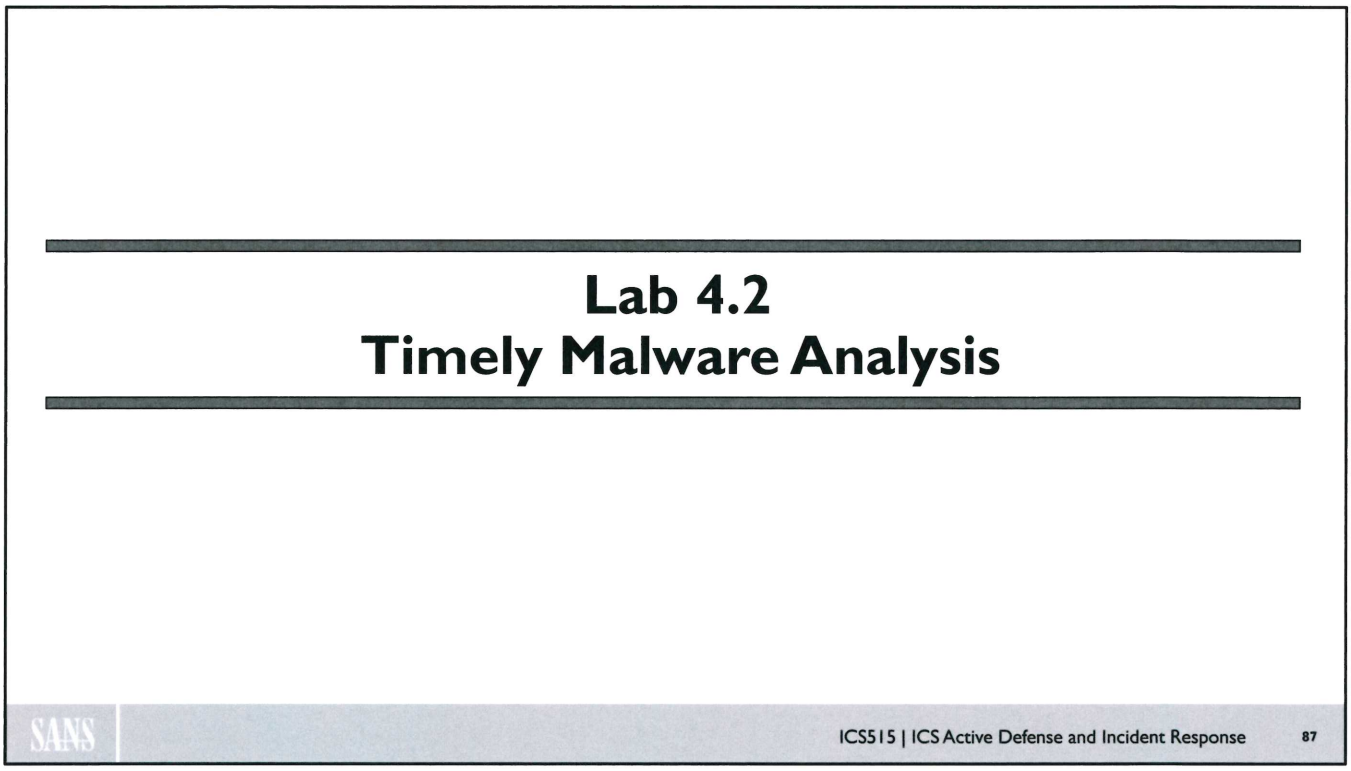


This page intentionally left blank.

Lab 4.2 Requests

- The Traffic Lights personnel would like you to analyze the project file and determine if it is also malicious
  - Havex had no Stage 2 capabilities and this would show that there are two intrusions and that they are not related
- For CWU, you need to analyze the output of the sandbox from executing the phishing email received in the CWU business network and determine if there is malicious activity present

This page intentionally left blank.



Lab 4.2: Timely Malware Analysis
Reference the Lab Workbook.

Lab 4.2 Recap – Traffic Lights

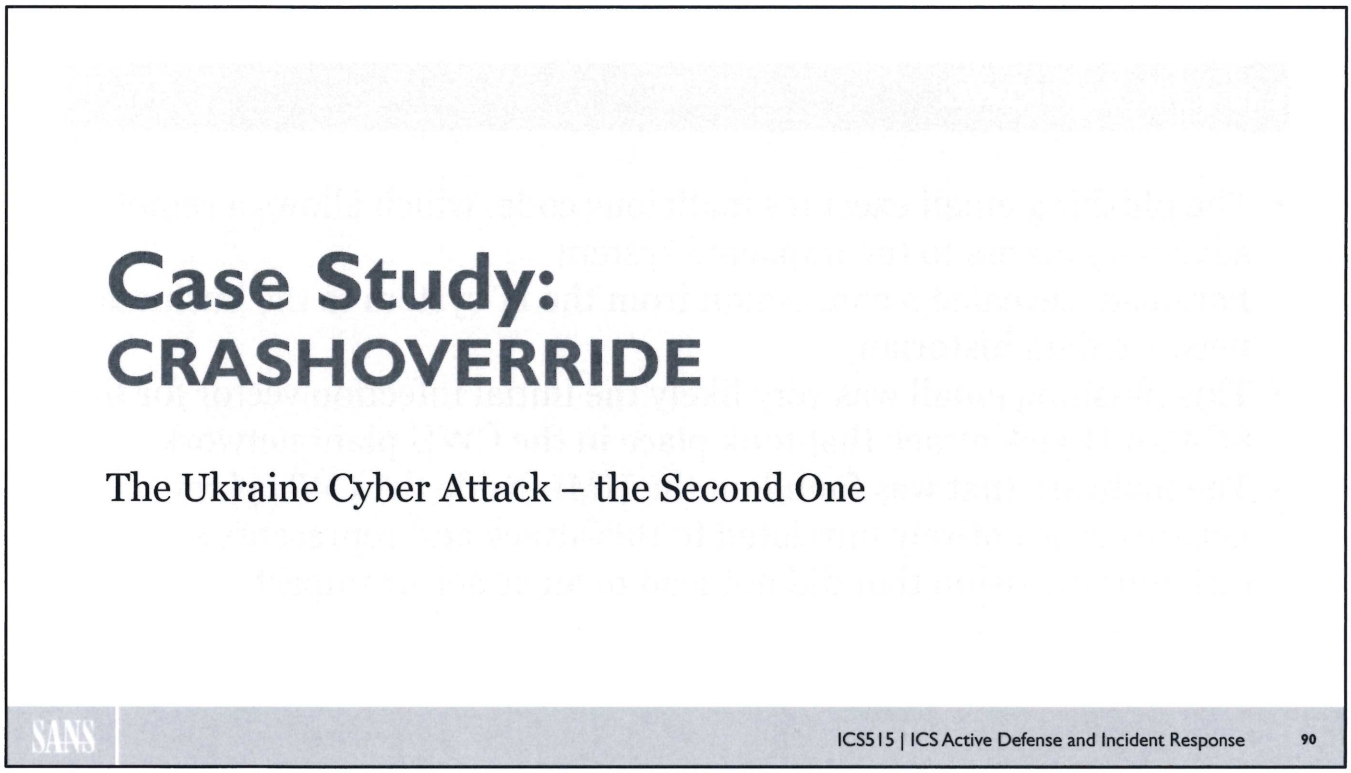
- The Project File manipulation was likely done by an external threat, and there may be an intrusion that went unnoticed in the Traffic Lights network
  - More attention and monitoring is needed as well as a review of the systems
- Havex has no Stage 2 capabilities nor did any of the analysis reveal external communications that would have connected for the adversary to pivot off of the insider threat's infection
- Therefore, we can conclude that these are two separate incidents

This page intentionally left blank.

Lab 4.2 Recap – CWU

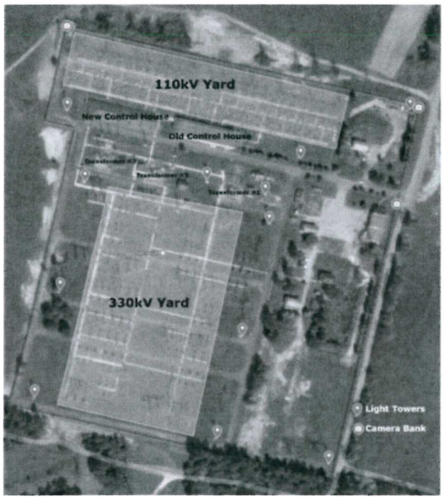
- The phishing email executes malicious code, which allows a remote adversary access to the impacted system
- Forensics revealed a connection from the IT system to the business network data historian
- This phishing email was very likely the initial infection vector for the SCADA Hijack attack that took place in the CWU plant network
- The malware that was found on the HMI inside the CWU plant network was entirely unrelated to this attack and represents a different intrusion that did not lead to an attack or impact

This page intentionally left blank.



This page intentionally left blank.

Ukrainian Power Outage

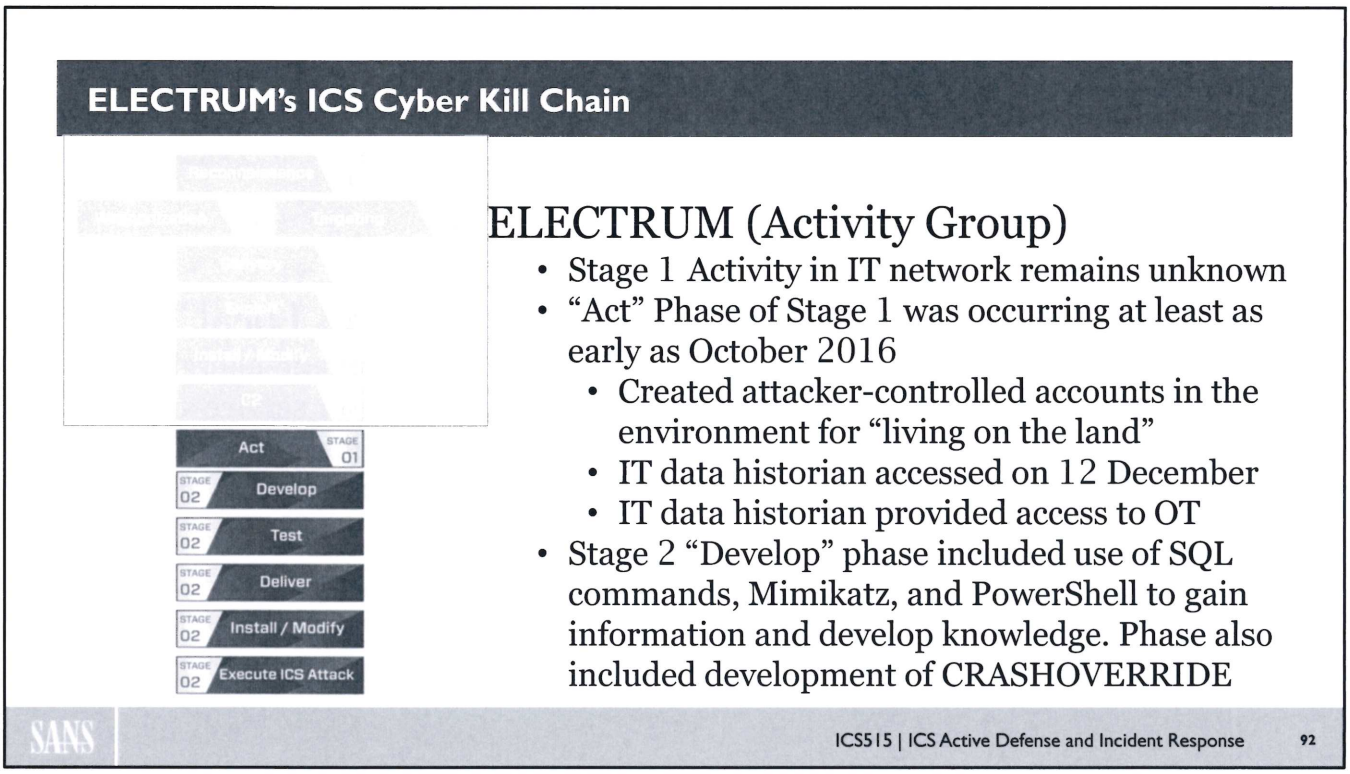


17 Dec 2016, 23:53 Local Time:

- Ukrenergo substation de-energizes
- Resulted in outage for service area
- Utility transferred into manual mode
- Began restoring power in 30 minutes

Ukrainian Power Outage

On December 17th, 2016 the second ever cyber attack to take down operations in a power grid occurred in Ukraine, again. This time only 1 substation was impacted but it was a Transmission level substation which resulted in 3x the power loss of the 2015 attack although less people were impacted and power restoration began as quickly as 30 minutes after the attack.



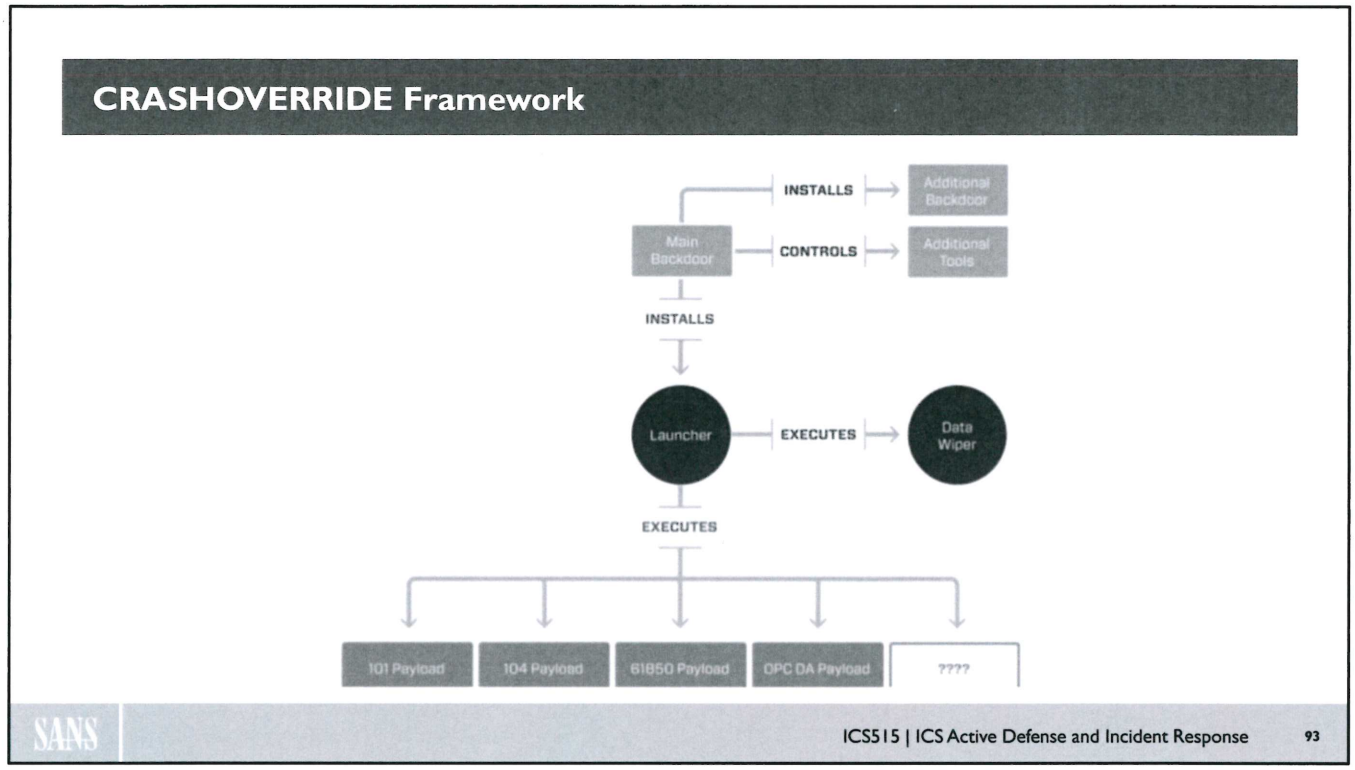
ELECTRUM's ICS Cyber Kill Chain

The Activity Group that created and leveraged the CRASHOVERRIDE malware in the Ukraine 2016 attack was identified by Dragos, Inc. as ELECTRUM. Dragos analyst Joe Slowik released a detailed paper on ELECTRUM's activity in 2018 which may be found here: https://dragos.com/media/CRASHOVERRIDE2018.pdf

Previously undisclosed details about the attack included the "Act" Phase of Stage 1 to identify that a data historian compromise in the IT network led to access in the OT network. From there utilization of common tools and native commands allowed the adversary to develop their knowledge of the environment while separately they also developed CRASHOVERRIDE. The adversaries utilized an OPC server to enumerate the environment which gave them the targeting information to load into CRASHOVERRIDE as a configuration file.

References

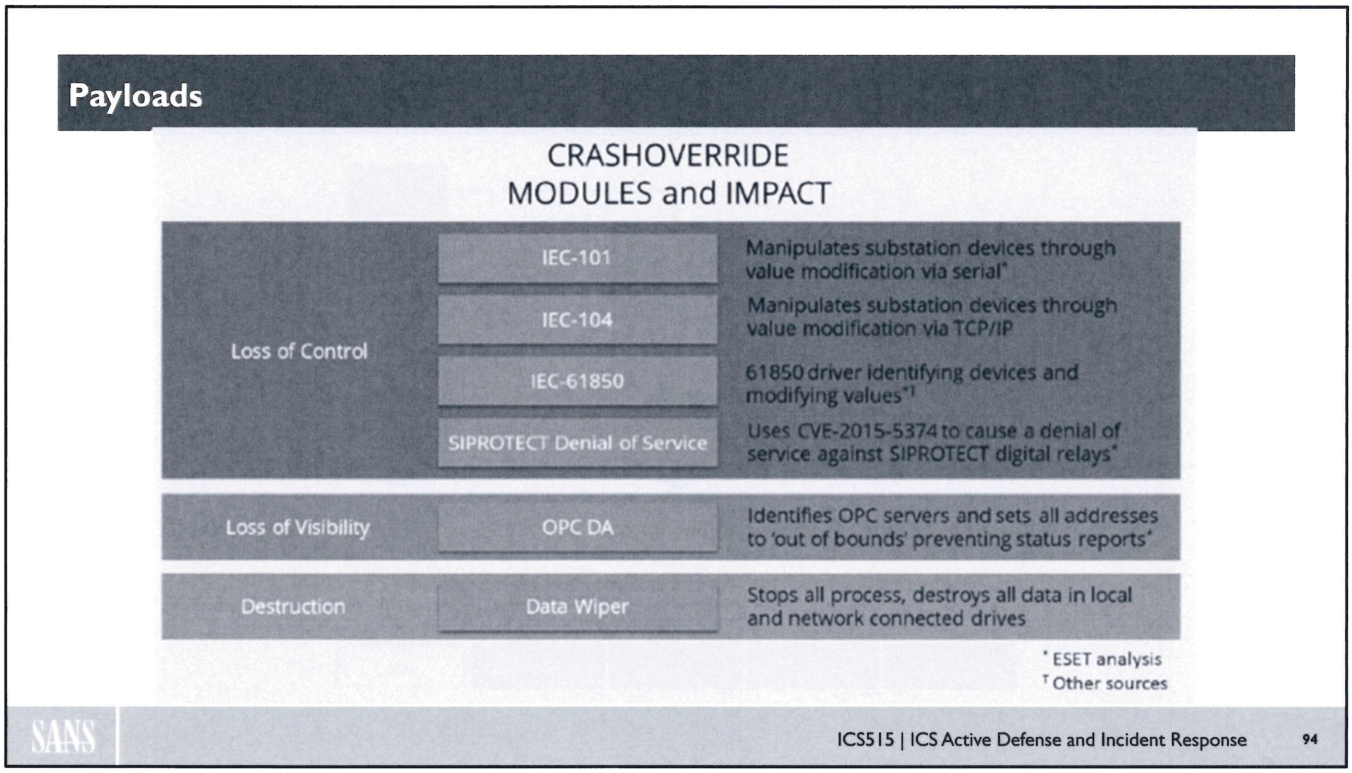
https://www.dragos.com/blog/20180607Electrum.html



CRASHOVERRIDE Framework

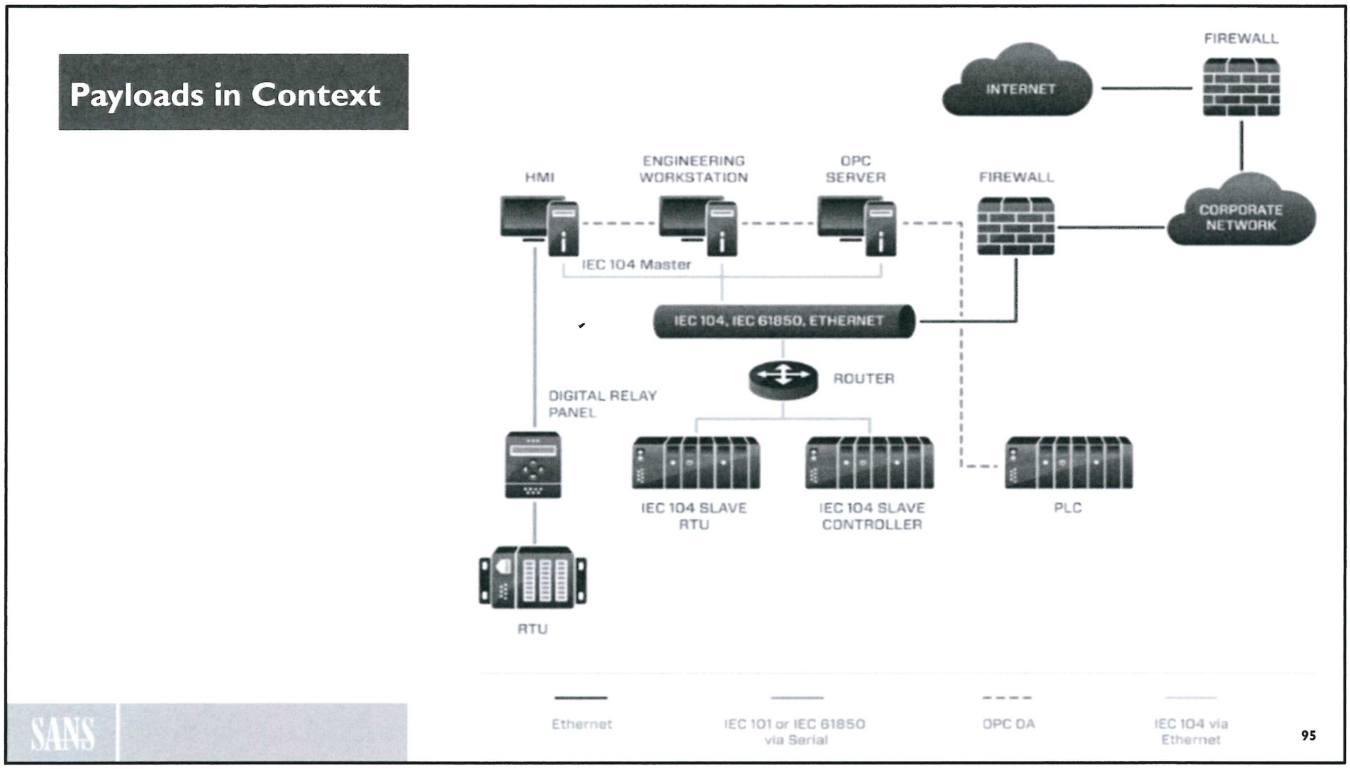
Details of the attack in 2016 were not known until a public report was released by ESET followed by a public report by Dragos, Inc. about an hour after the ESET report. ESET identified the malware as Industroyer assessing that it could target any number of industries and not just electric grid. Dragos disagreed with this assessment and thus named it CRASHOVERRIDE instead of the "industry destroyer" themed choice by ESET. Both worked before the publications to analyze and inform their customers. Dragos also ensured that the US community and various national CERTs around the world knew before ESET released it to the news. Private alerts went out through E-ISAC, EEI, and DHS to the US electric community after notification to them by Dragos which showcased a great community effort.

https://dragos.com/blog/crashoverride/CrashOverride-01.pdf
https://ics-cert.us-cert.gov/alerts/ICS-ALERT-17-206-01



Payloads
CRASHOVERRIDE was the first ever malware specifically designed to disrupt electric grids. It contained a number of modules that ESET and Dragos were able to identify and analyze. The IEC104, IEC61850, and OPC modules though were each capable of causing the attack; however only one, the IEC104 module, was used during the attack. The Dragos researchers put forth an assessment that this was indication that the attackers may have been testing the capability against the Ukrainians and that the other modules were thus available for future attacks. The combination of the modules allow the malware to be scalable across any electric grids using these protocols (Europe, Middle East, and Asia) and would require only slight tailoring to include DNP3 to allow the malware to work in US and the rest of the world. The SIPROTECT vulnerability was not required for the attack to work but could have had physical impacts if the protection equipment, a digital relay in this case, was no longer functioning when electric operators reconnected the equipment. However, no physical damage took place during this attack.

Image Reference:
Dragos, Inc.

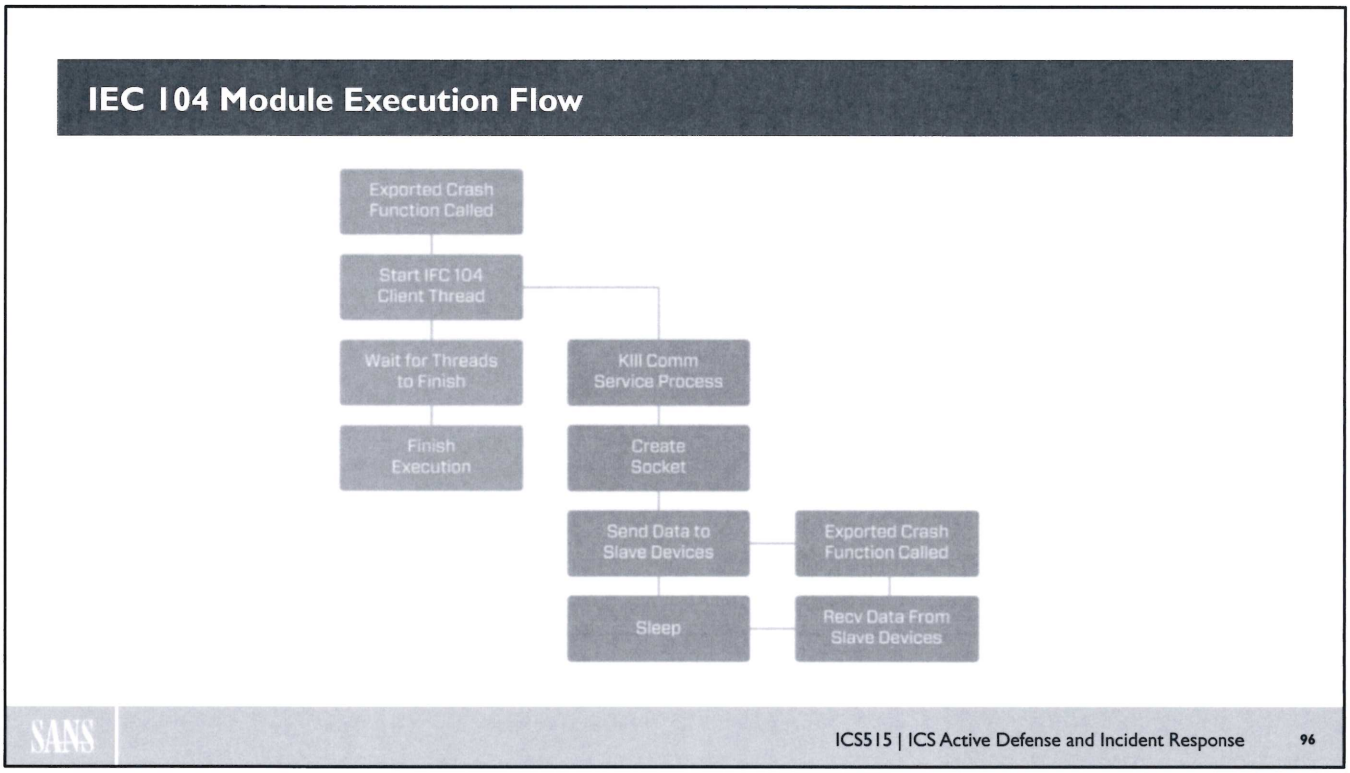


Payloads in Context
Here is an image of a typical electric substation utilizing the protocols that CRASHOVERRIDE targeted. CRASHOVERRIDE effectively sent IEC104 commands by assuming a master position after installing on an HMI and then repeatedly sent open commands through the digital relay to the RTU to force open, and remain open, circuit breakers thus de-energizing the equipment.

The capability is highly scalable but still must be put into place by the adversary meaning that there is a human component to the attack that is required. I.e. this is a Stage 2 effect in the ICS Cyber Kill Chain but there must still be a Stage 1 operation. This is not a wormable capability that would work effectively due to the segmentation and structuring of the networks.

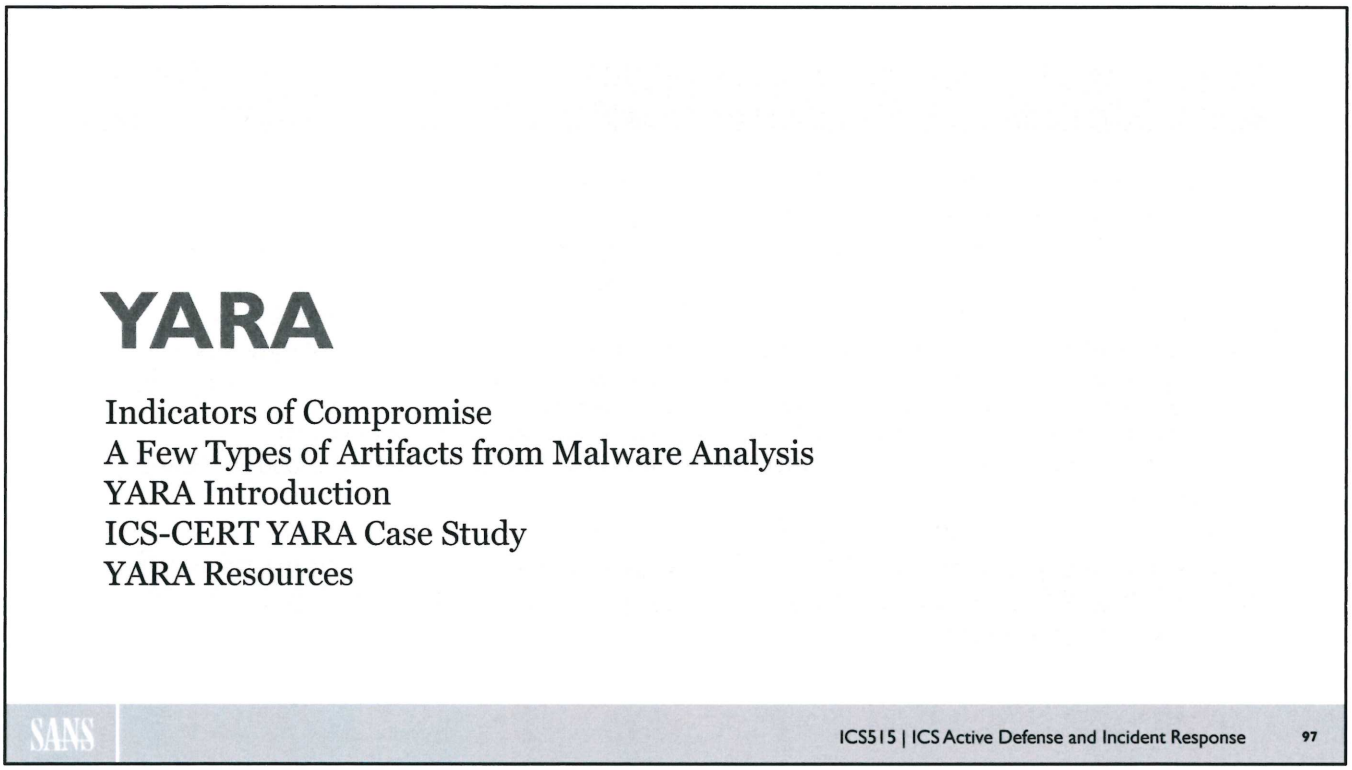
The activity group ELECTRUM was responsible for the CRASHOVERRIDE capability and were able to pivot from data historians, SQL servers in this case, from the IT to the OT network and put CRASHOVERRIDE in place.

Image Reference:
Dragos, Inc.

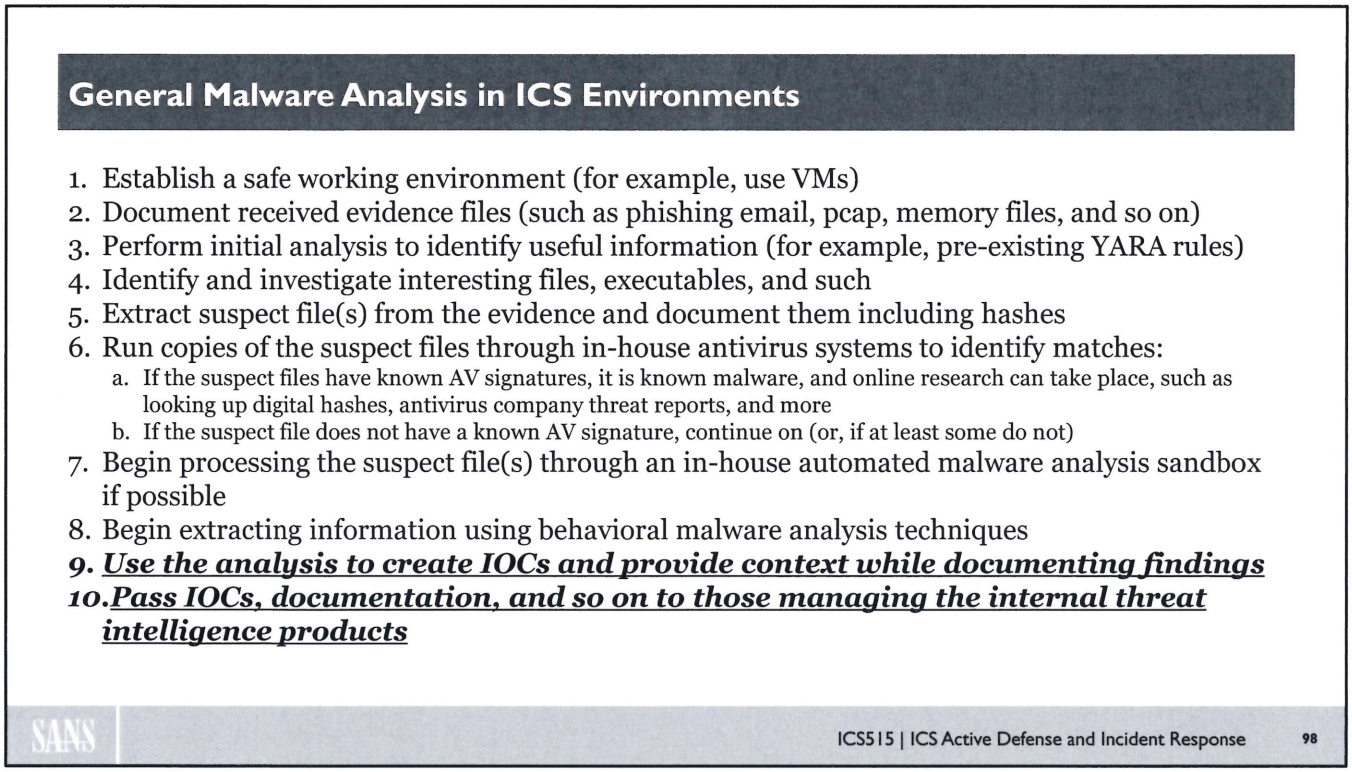


IEC104 Usage:

- Communication between control station and substation
- TCP/IP implementation of IEC 101 with subset of commands
- Features:
  - Master slave architecture
  - On-demand or spontaneous transmission
  - Remote command functionality
  - File Transfer
- Configuration file required
  - Needs a target IP, other value
  - Can contain multiple targets
- Requires *manual staging*
- Manipulates target by changing state to ON or OFF



This page intentionally left blank.



General Malware Analysis in ICS Environments

Most threats involve malware, and today's focus is heavily centered around learning from malware. Even threats that don't rely on malware to impact the ICS may rely on malware to gain a foothold in the ICS or corporate network. The same thought processes and type of analysis that goes into malware analysis is useful to understanding nonmalware-based threats.

To help guide the day, the 10-step process is used throughout the day. This is a general methodology and may not represent the abilities and processes of every ICS security team, but it covers the vast majority of cases.

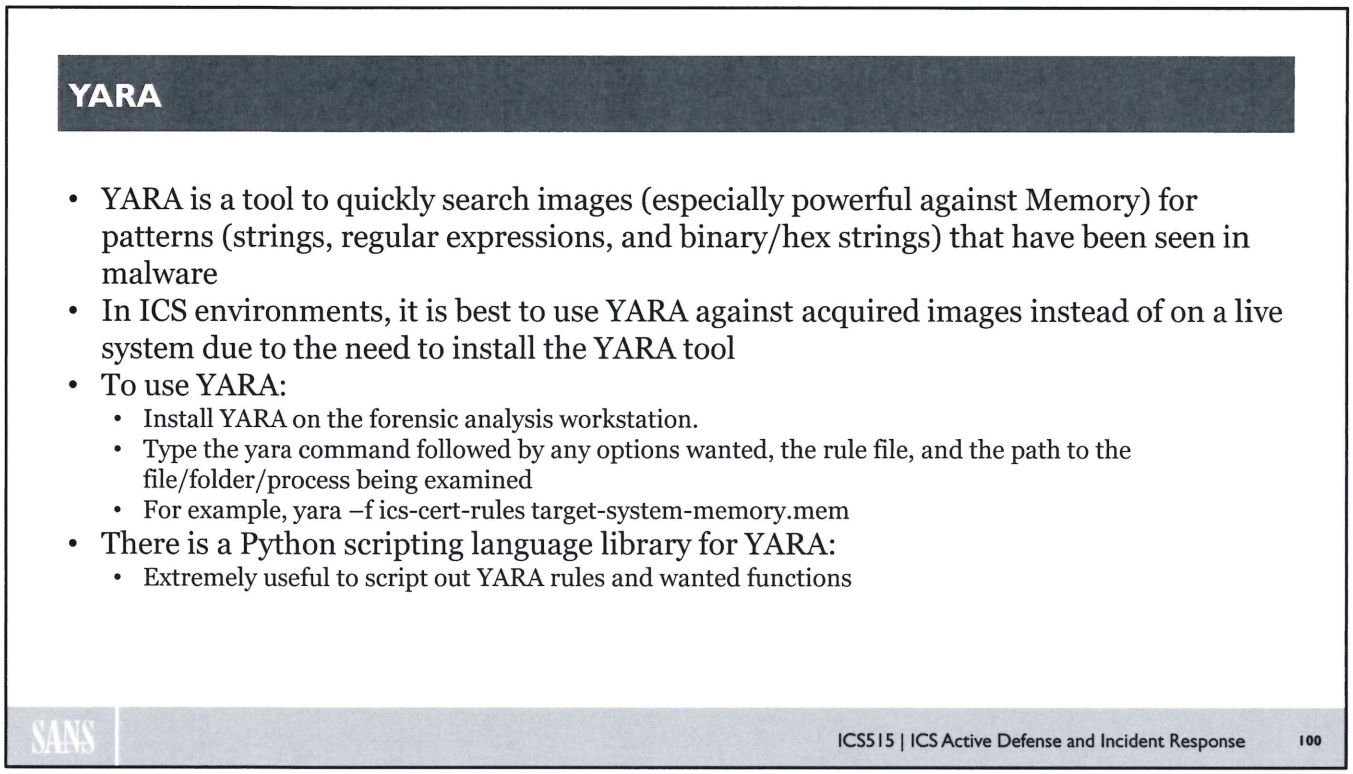
Throughout the 10-step process, analysts should think about what can be done to the architecture to deny or delay the adversary and its capabilities.

A Few Types of Artifacts from Malware Analysis

- A subset of a few types of artifacts that are useful in creating IOCs:
- Malware artifacts:
  - Digital hashes
  - Driver types and signing authorities
  - Imported and exported functions
  - File and executable names
  - Strings
- System interaction artifacts:
  - Processes the malware calls or injects into
  - Registry keys created or manipulated
- Network interaction artifacts
  - Delivery IP addresses
  - C2 IP addresses
  - Protocols utilized and any malformed/manipulated aspects

A Few Types of Artifacts from Malware Analysis
Just a quick list of some of the important things to focus on from documentation regarding malware analysis; these are all useful in IOC development.

Even a simple strings search in a malicious file can return valuable information for creating YARA rules.



YARA

YARA is a great tool and was initially intended for the malware analysis community, but it has powerful implications for incident response. Trying to scope a problem is a lot easier after you understand the threat and have a quick way to identify it in systems. YARA should be run against acquired files and not live systems whenever possible to preserve the evidence.

YARA Uses

YARA has been compared to the grep command:

Has the capability to quickly parse through data

Extremely flexible, including capability to use regular expressions and wildcards

Capability to add metadata, descriptions, and titles to YARA rules allow the rules to be quickly shared between those doing malware analysis or IR

YARA is perfect for incident response in that it allows a quick initial triage against acquired data to determine if there is reason to pass the data to those performing malware analysis:

Helps to identify the scope of the infection and provides confidence that a system is clean

False positives will naturally occur and that is okay

ICS-CERT reports and some threat intelligence reports from vendors are commonly including YARA rules for use in incident response

Reference:

https://blog.malwarebytes.org/intelligence/2013/10/using-yara-to-attribute-malware/

Sample YARA Rule #1

```
rule silent_banker : banker
{
    meta:
        description = "This is just an example"
        thread_level = 3
        in_the_wild = true

    strings:
        $a = {6A 40 68 00 30 00 00 6A 14 8D 91}
        $b = {8D 4D B0 2B C1 83 C0 27 99 6A 4E 59 F7 F9}
        $c = "UVODFRYSIHLNWPEJXQZAKCBGMT"

    condition:
        $a or $b or $c
}
```

Reference: YARA GitHub

Sample YARA Rule #1
Notice here how there is a rule name, metadata about what it is, string conditions, and then conditions needing to be met. That is, if "a or b or c" is met, then the alert returns "true" and the metadata associated with the rule.

YARA Key Points
YARA is a very easy to understand and C-programming-like language. Reading the YARA documentation does not take long and can be of great value: https://yara.readthedocs.org/en/v3.4.0/ .

There are some key points to keep in mind when writing YARA rules. First, there are three types of strings that you can make rules out of: Text, Hex, and Regular Expression. Text strings are enclosed in double quotes, Hex strings are enclosed in curly brackets and can include spaces between bytes or do without them, and Regular Expression strings are enclosed in either double quotes or curly brackets, depending on the type of data (text or hex) being used.

To include comments, you can enclose a multiple line comment in a /* */ or in a single line comment you can just use //

Multiple Types of Rules
YARA allows you to create multiple types of rules. For example, the "global rule" condition allows you to create a rule that applies to all other rules run. This allows you to tailor large sets of rules with a single rule before you run them against files. As an example, if you only want YARA rules to match against a certain type of file such as an .exe, a global rule could be made to set a condition declaring that files must match the .exe type. Then all the other rules run would only match against such files. Additionally, private rules can be created. Private rules do not alert and thus seem useless. However, rules can import other rules and build upon them, which means that private rules can be used as a triggering event for other rules. As an example, one YARA rule might look for a suspicious C2 server but, to limit false positives, a private rule is made that looks for other indicators such as a known malicious process. The C2 YARA rule then could import the malicious process YARA rule and only alert when both rules match. Lastly, rules can take advantage of tags to help analysts filter out rules. For example, an "APT" tag could be made to help analysts focus on only those types of threats faced.

Hex Special Values

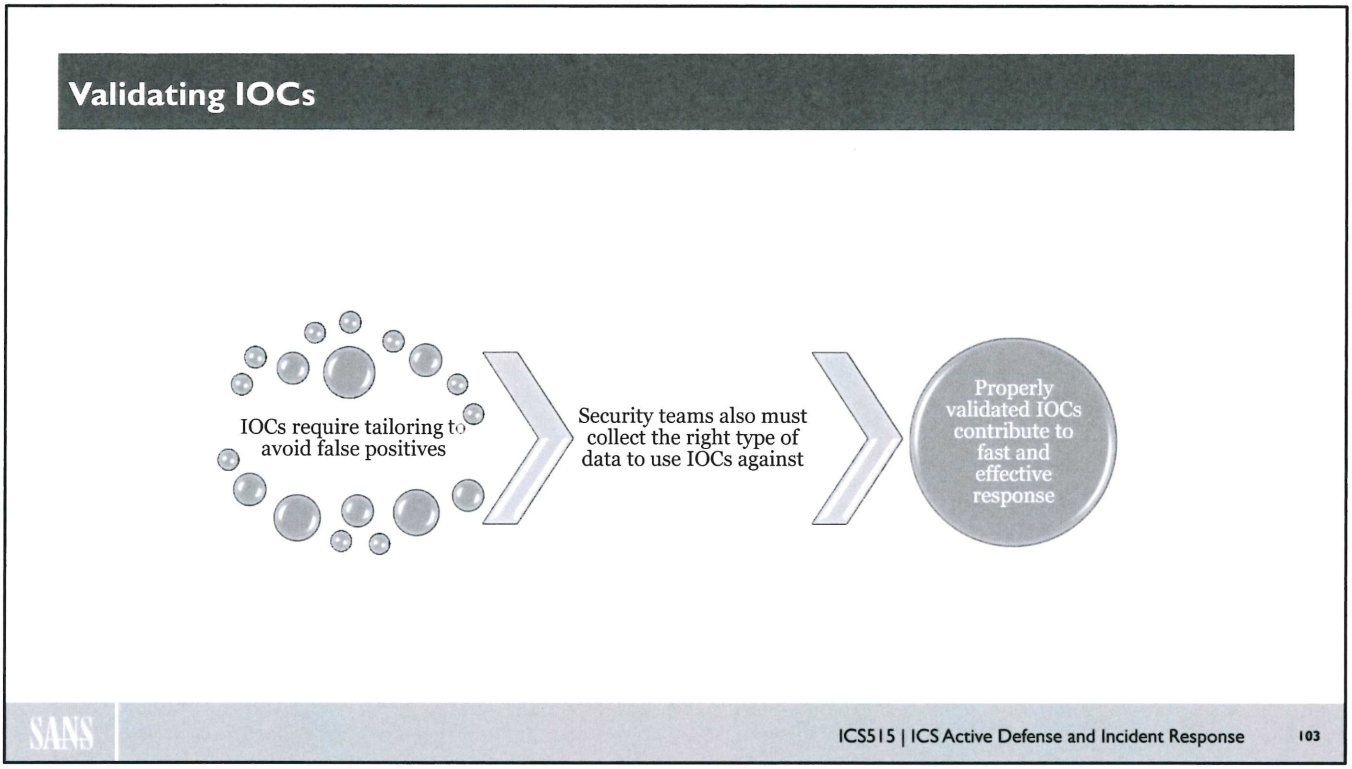
- Hex contains question marks for wild cards
- Jumps can tell you how many bytes can exist before the next sequence is seen
- Alternatives allow for an OR-styled Boolean comparison

```
rule JumpExample
{
    strings:
        $hex_string = { F4 23 [4-6] 62 B4 }

    condition:
        $hex_string
}
```

```
rule AlternativesExample1
{
    strings:
        $hex_string = { F4 23 ( 62 B4 | 56 ) 45 }

    condition:
        $hex_string
}
```


Hex Special Values

One of the great aspects of using Hex strings is the ability to use wild cards. Placing a question mark in the place of a value will return any value in that spot. This can be done for parts of a byte or multiple bytes. Additionally, Hex strings can take advantage of a Jump, which allows a range to be declared. The range states that the values before and after it can be within that range of each other and still match the rule. Alternatives allow for an OR-styled Boolean operator comparison. For an example, see the slide for a sample Jump rule and sample Alternative rule from the YARA documentation.



Validating IOCs

IOCs need a lot of tailoring to avoid false positives. A general rule of thumb is to tailor IOCs to eliminate as many false positives as possible prior to uncovering a threat. Once a threat is identified, accept more false positives in an effort to open up the aperture of a rule to find variants of the malware or threat in the environment. False positives prior to finding a threat are one of the most costly aspects of using IOCs and can quickly discourage the security process. Validate IOCs by testing them against digital images in the environment.

YARA and Scanning Images

- YARA will not give you the same type of warnings OpenIOC will about missing data
- Automation, wide flexibility, and an ability to search various types of images including archives, memory images, disk images, etc. make YARA a favorite among analysts
- Common YARA scanning tools include
  - YARA run from Python
  - VirusTotal Intelligence
  - CrowdResponse

YARA and Scanning Images
When an OpenIOC rule is placed into Redline, it will give an error if there are any missing data types. For example, if Redline sees that only SHA256 hashes are collected and the OpenIOC rule is reliant on MD5s, an error warning will appear. In YARA, though, no such errors exist. However, YARA allows for automation and scripting, which gives it a lot more flexibility than OpenIOCs. For this reason and for being an open standard that analysts can pass around quickly, it has become a favorite among analysts.

YARA is a powerful scanning tool, but there are other tools that leverage YARA as well. For example, CrowdStrike's CrowdResponse allows for the quick leverage of YARA rules, there is a version of YARA specifically for Python, and VirusTotal Intelligence allows YARA rules to be input to match against malware samples submitted by users around the world. Also, its RetroHunt tool lets you scan against the last few weeks of submissions by users.

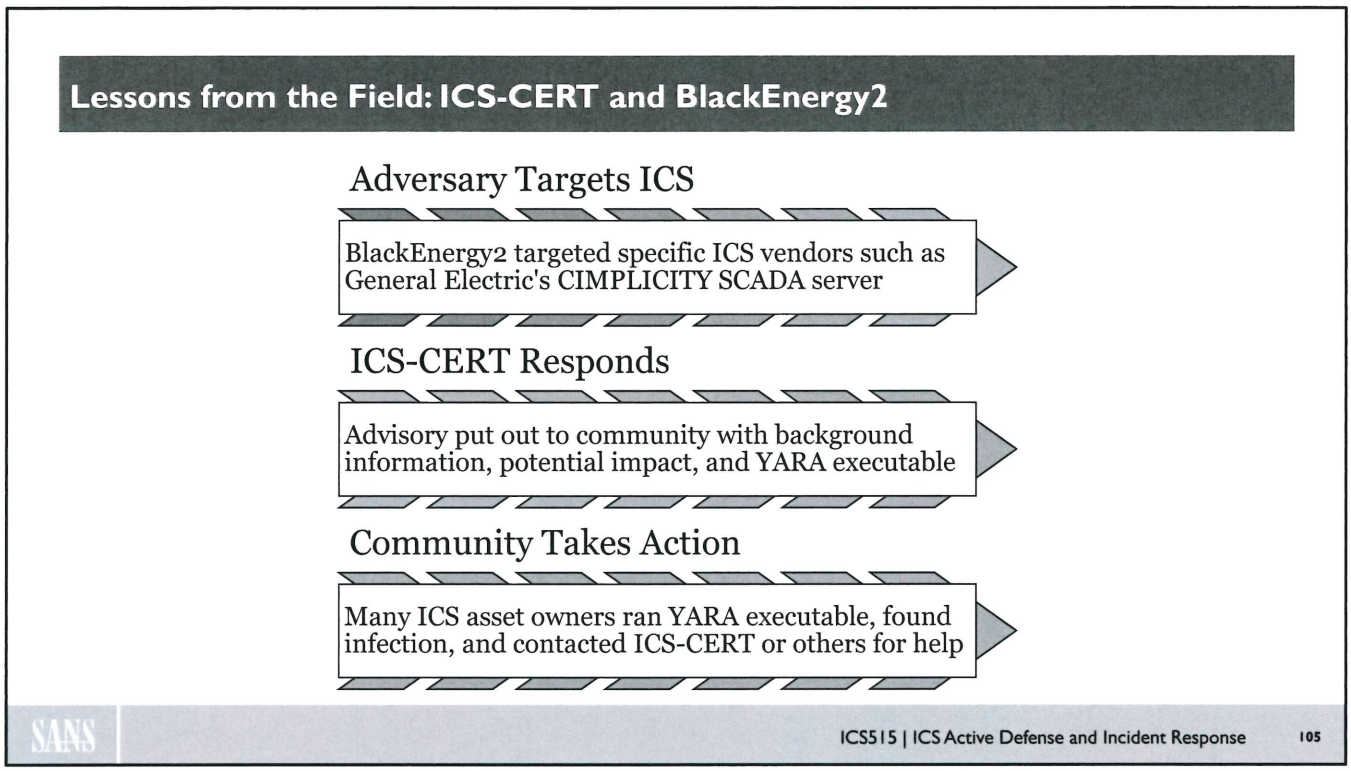
CrowdResponse: https://digital-forensics.sans.org/blog/2014/04/09/signature-detection-with-crowdresponse

Lesson from the Field: VirusTotal Intelligence
VirusTotal Intelligence is an extremely useful service (although a paid service and pricey for many). It allows users to perform advanced searches for strings, bulk hashes, file sizes, times submitted, keywords, and more. It also allows users to create and submit YARA signatures. Then, any YARA signatures that match against submitted pieces of malware alert the user. This allows folks to build YARA signatures for their environment almost akin to Snort signatures and learn when data such as stolen emails, passwords, etc. are seen in other pieces of malware.

The API (application programming interface) in VirusTotal allows massive amounts of lookups against malware automatically each day as well. Users can thus have internal lists of potentially malicious hashes or malware and directly submit or query them from VirusTotal. Automation saves analyst time—use it whenever possible.

Every time a user submits files to VirusTotal, it can be data mined for threat and malware analysts using VirusTotal Intelligence.

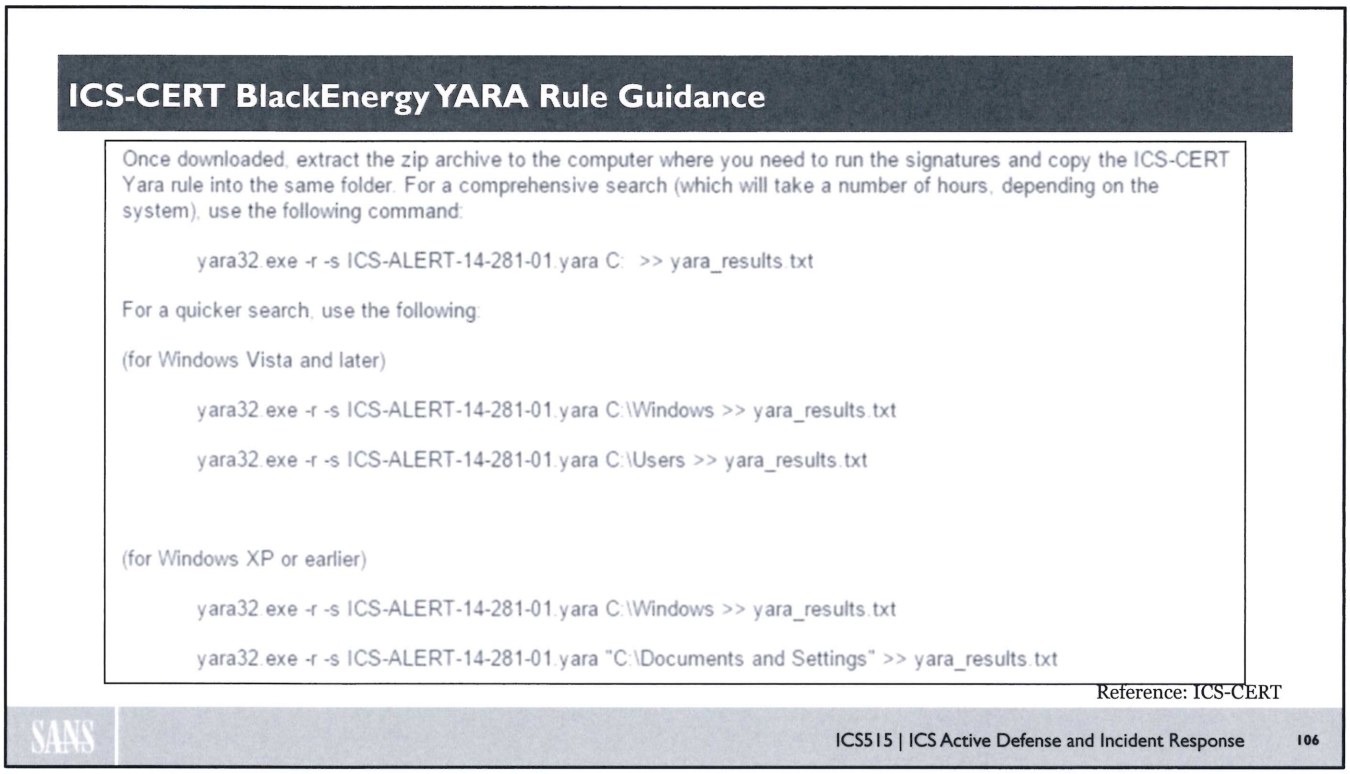
Reference:
www.virustotal.com



Lessons from the Field: ICS-CERT and BlackEnergy2

The ICS-CERT offered a unique case study that required strong analysis and understanding of the community they were part of. In 2014, the third-ever piece of malware to specifically target ICS was discovered—BlackEnergy2. It even contained a plugin to cause destruction of data, which increased the concern around it. However, the ICS-CERT had a huge challenge: most ICS asset owners did not have robust security awareness or incident response plans and they definitely didn't have familiarity with YARA rules and IOCs. How was the ICS-CERT supposed to respond and scope hundreds of impacted facilities then?

The ICS-CERT decided to create an executable version of the YARA IOC. This broke standard practices in Incident Response—you never run things like IOCs and YARA rules on the potentially infected system itself! You always digitally acquire evidence like a disk image or memory and then analyze it elsewhere. But ICS-CERT knew their community and understood their threat. I.e. they understood themselves and the threat well enough to break conventional wisdom in an informed way. Because the community would never acquire digital images and then run YARA signatures (no expertise to do so), they created an executable YARA rule to be run on the system. This helped them, and the community, more quickly respond to an advanced threat that posed a potentially significant impact. This is a valuable lesson learned and case study to any threat intel analyst.



ICS-CERT BlackEnergy YARA Rule Guidance

Here is the guidance that the ICS-CERT put out. It was to run YARA on the system that might be infected. This is a great initial approach, but if you suspect a compromise, run only YARA against acquired evidence and not the system itself unless in a time-critical scenario.

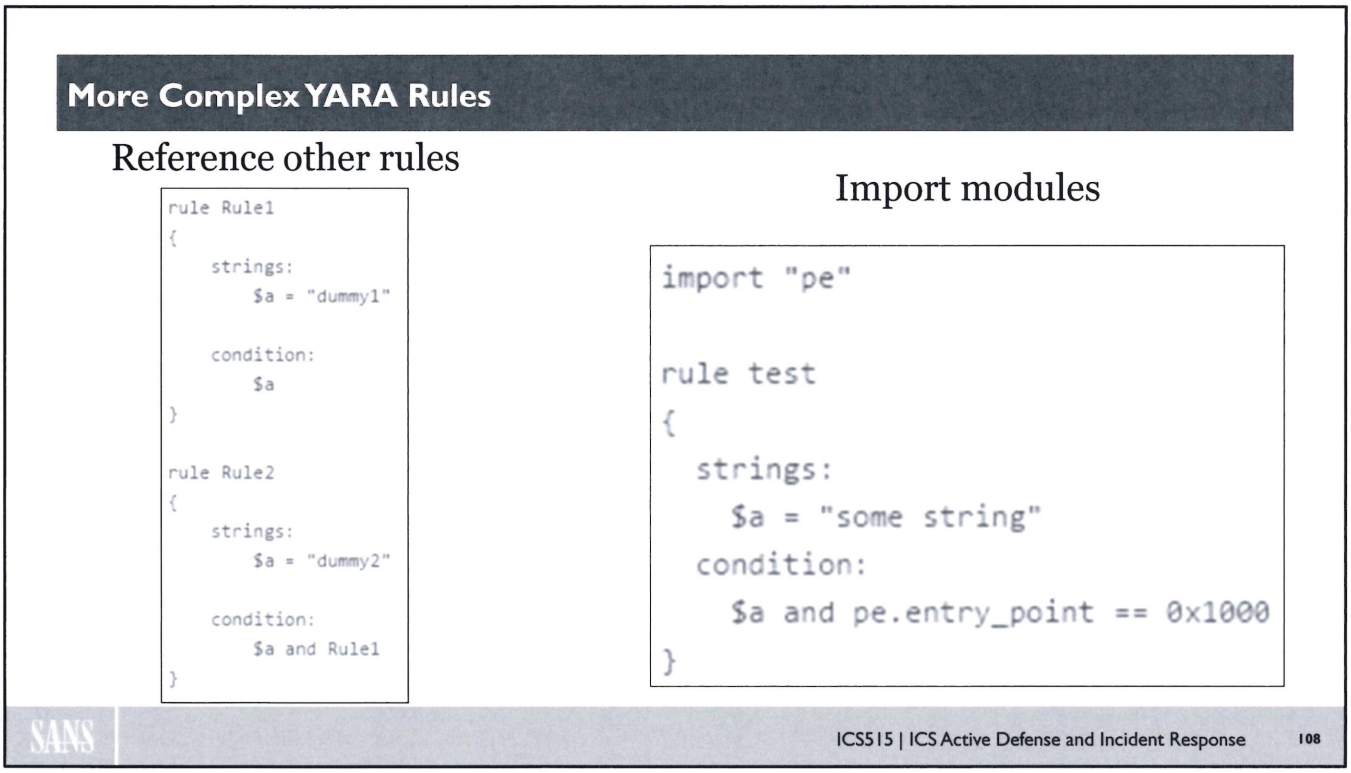
ICS-CERT BlackEnergy YARA Rule Specifics

```
//ICS-CERT rule to detect BlackEnergy2 "light" and BlackEnergy3.  Will not detect BE2 sys variant.
//version 1
 {
        strings:
                $sb1 = {C7 [1-5] 33 32 2E 64 C7 [1-5] 77 73 32 5F 66 C7 [1-5] 6C 6C} //ws3_32.dll
                $sb2 = {C7 [1-5] 75 73 65 72 C7 [1-5] 33 32 2E 64 66 C7 [1-5] 6C 6C} //user32.dll
                $sb3 = {C7 [1-5] 61 64 76 61 C7 [1-5] 70 69 33 32 C7 [1-5] 2E 64 6C 6C} //advapi32.dll
                $sb4 = {C7 [1-5] 77 69 6E 69 C7 [1-5] 6E 65 74 2E C7 [1-5] 64 6C 6C}  //wininet.dll
                $sb5 = {C7 [1-5] 73 68 65 6C C7 [1-5] 6C 33 32 2E C7 [1-5] 64 6C 6C} //shell32.dll
                $sb6 = {C7 [1-5] 70 73 61 70 C7 [1-5] 69 2E 64 6C 66 C7 [1-5] 6C} //psapi.dll
                $sb7 = {C7 [1-5] 6E 65 74 61 C7 [1-5] 70 69 33 32 C7 [1-5] 2E 64 6C 6C} //netapi32.dll
                $sb8 = {C7 [1-5] 76 65 72 73 C7 [1-5] 69 6F 6E 2E C7 [1-5] 64 6C 6C} //version.dll
                $sb9 = {C7 [1-5] 6F 6C 65 61 C7 [1-5] 75 74 33 32 C7 [1-5] 2E 64 6C 6C} //oldaut32.dll
                $sb10 = {C7 [1-5] 69 6D 61 67 C7 [1-5] 65 68 6C 70 C7 [1-5] 2E 64 6C 6C} //imagehlp.dll

        condition:
                3 of them
}
```

Reference: YARA GitHub

ICS-CERT BlackEnergy YARA Rule Specifics

Notice how the rule uses the format like the sample rule shows. This is just one of the rules ICS-CERT included and 2459 Fisher Rd highlights information related to certain suspect files. If three or more of them exist on the system, the alert returns true.



More Complex YARA Rules

There are a number of ways to make more complex YARA rules. A few examples will be shown across the next few slides to note options such as declaring file sizes. Here, though, we see the ability to reference other rules inside the condition of a YARA rule. This allows for very tailored rules and for spidering rules out based on different sets of YARA rule families. For example, a script could be created to run one set of YARA rules against a file sample to determine the type of file being interacted with such as an executable. From there, a set of YARA rules could be applied if it was in fact an executable file. From there, general rules could be applied across different families of malware which, in turn, would cause other YARA rules to be compared against the sample for more specific types of the malware family identified. As an example, a combination of rules could take a file, determine it was an executable that fits into the PlugX malware family, and then match YARA rules to see specifically which sample was being used. This could be done for automation purposes to reduce false positives of simply running all the rules against all the samples encountered.

Importing modules allows you to import portable executable (PE) information, Cuckoo sandbox information, digital hashes, and more. Additionally, there is an easy-to-use module guide to create your own modules for tools or data sets you have. Modules allow YARA rules to move past more simple string matching to having more context, such as identifying a digital hash of a string.

Sample YARA Rule: Uncommon File Size

```
rule Suspicious_Size_chrome_exe {
    meta:
        description = "Detects uncommon file size of chrome.exe"
        author = "Florian Roth"
        score = 60
        date = "2015-12-21"
    condition:
        uint16(0) == 0x5a4d
        and filename == "chrome.exe"
        and ( filesize < 500KB or filesize > 1300KB )
}
```

Reference: Florian Roth

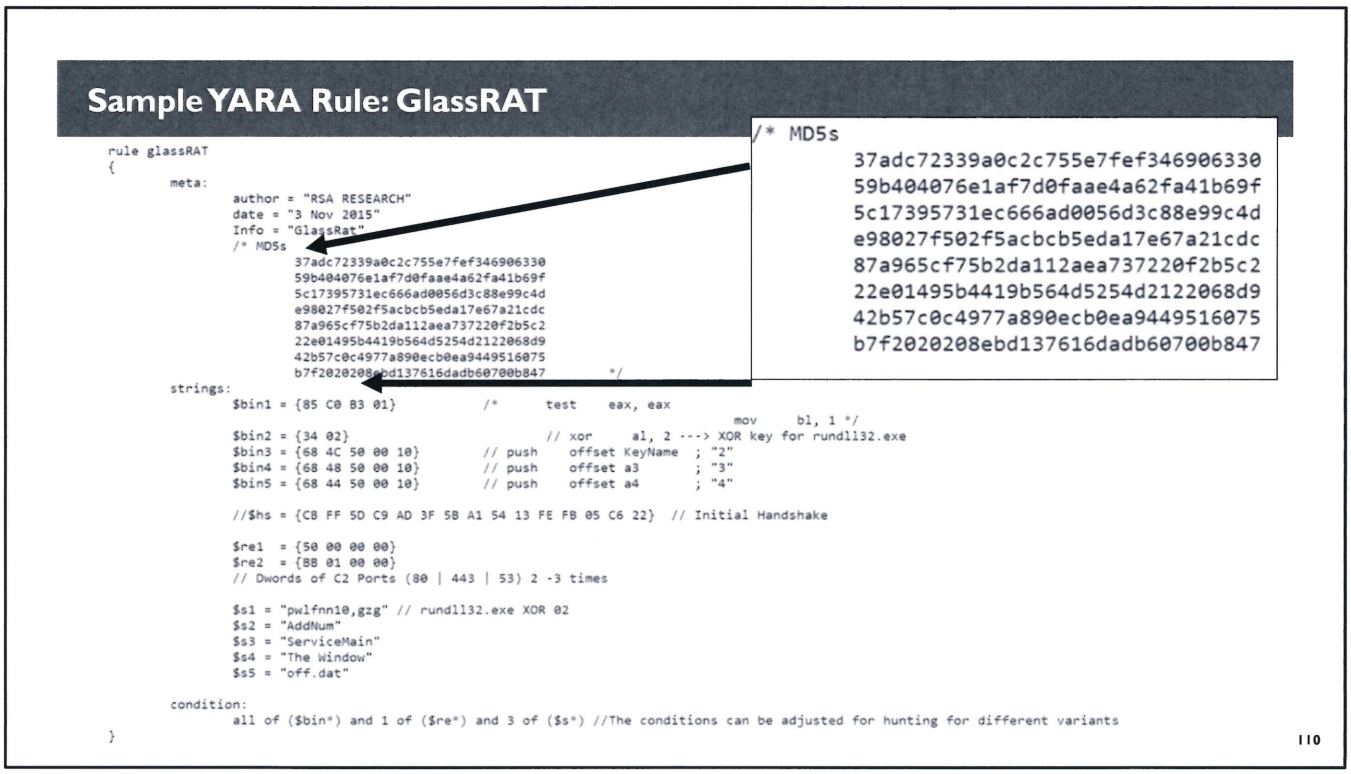
Additional Sample YARA Rule

In this example, Florian Roth created a YARA rule to detect suspicious sizes of known files. To create the list of file sizes, he downloaded samples of malicious files from VirusTotal and determined what the normal range in KB was of good files. For example, he identified that chrome.exe is usually between 500-1300 KB in size but often 10-500 and 1300+ was observed as malicious. This is a great way to make an initial YARA rule that looks for potentially malicious activity. This is not a high-confidence IOC but is useful for hunting for potentially malicious activity.

In the YARA rule, note the "uint16(0) == 0x5a4d". uintXX in YARA designates 8, 16, or 32 bit signed integers to perform an offset or virtual address from. In this case, the HEX "5a4d" is looking for the "MZ" header at offset 0 that is associated with portable executable files.

We also see other aspects of YARA here, including the ability to dictate file size and filenames. Here, the YARA rule is looking for "chrome.exe" with a valid MZ header and only smaller than 500KB or larger than 1300KB.

Source: https://www.bsk-consulting.de/2015/12/22/yara-rules-to-detect-uncommon-system-file-sizes/



Sample YARA Rule: GlassRAT
This is one of the YARA rules that RSA released for GlassRAT. There are a few important things here to highlight that were done extremely well.

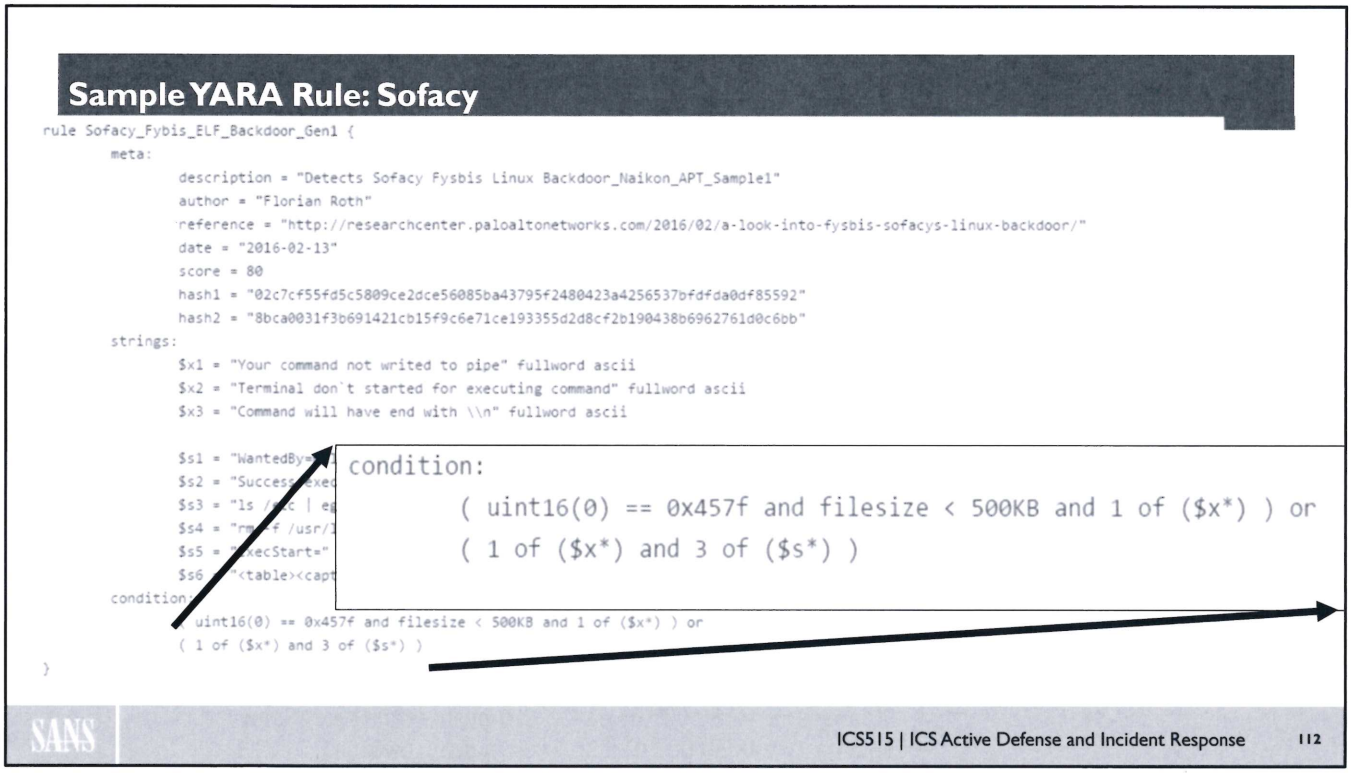
First, in the metadata, the YARA rule specifically gives the MD5 hashes of the samples of GlassRAT analyzed. This helps other analysts know what the YARA rule should and possibly should not (samples that aren't included) work against. Additionally, it gives the samples that analysts can find and analyze themselves or test their other rules against.

Second, the YARA rule has comments (denoted by the "//" for single line comments) that are for the analysts who review the rules to identify what each string encompasses.

Third, the rule segments different types of strings into different groupings based on the variables. As an example, the rule has $bin for like items, $re for like items, and $s for like items. This allows a really nice condition. The condition in this rule denotes that all of the strings in the $bin variable (notice the * for a wildcard which would include all the numbers after $bin), 1 of the $re strings, and 3 of the $s strings must be present. This is an excellent example of a tailored and focused rule that still has flexibility.

For easier viewing, here is the content of the YARA rule:

```
rule glassRAT
{
        meta:
                author = "RSA RESEARCH"
                date = "3 Nov 2015"
                Info = "GlassRat"
                /* MD5s
                        37adc72339a0c2c755e7fef346906330
                        59b404076e1af7d0faae4a62fa41b69f
                        5c17395731ec666ad0056d3c88e99c4d
                        e98027f502f5acbcb5eda17e67a21cdc
                        87a965cf75b2da112aea737220f2b5c2
                        22e01495b4419b564d5254d2122068d9
                        42b57c0c4977a890ecb0ea9449516075
                        b7f2020208ebd137616dadb60700b847        */
        strings:
                $bin1 = {85 C0 B3 01}                   /*      test
eax, eax

                                                        mov     bl, 1 */
                $bin2 = {34 02}                                         //
xor     al, 2 ---> XOR key for rundll32.exe
                $bin3 = {68 4C 50 00 10}        // push    offset KeyName  ; "2"
                $bin4 = {68 48 50 00 10}        // push    offset a3       ; "3"
                $bin5 = {68 44 50 00 10}        // push    offset a4       ; "4"

                //$hs = {CB FF 5D C9 AD 3F 5B A1 54 13 FE FB 05 C6 22}  // Initial
Handshake ---> can be added or removed for hunting for different variants

                $re1  = {50 00 00 00}
                $re2  = {BB 01 00 00}
                // Dwords of C2 Ports (80 | 443 | 53) 2 -3 times

                $s1 = "pwlfnn10,gzg" // rundll32.exe XOR 02
                $s2 = "AddNum"
                $s3 = "ServiceMain"
                $s4 = "The Window"
                $s5 = "off.dat"

        condition:
                all of ($bin*) and 1 of ($re*) and 3 of ($s*) //The conditions can be adjusted
for hunting for different variants
}
```

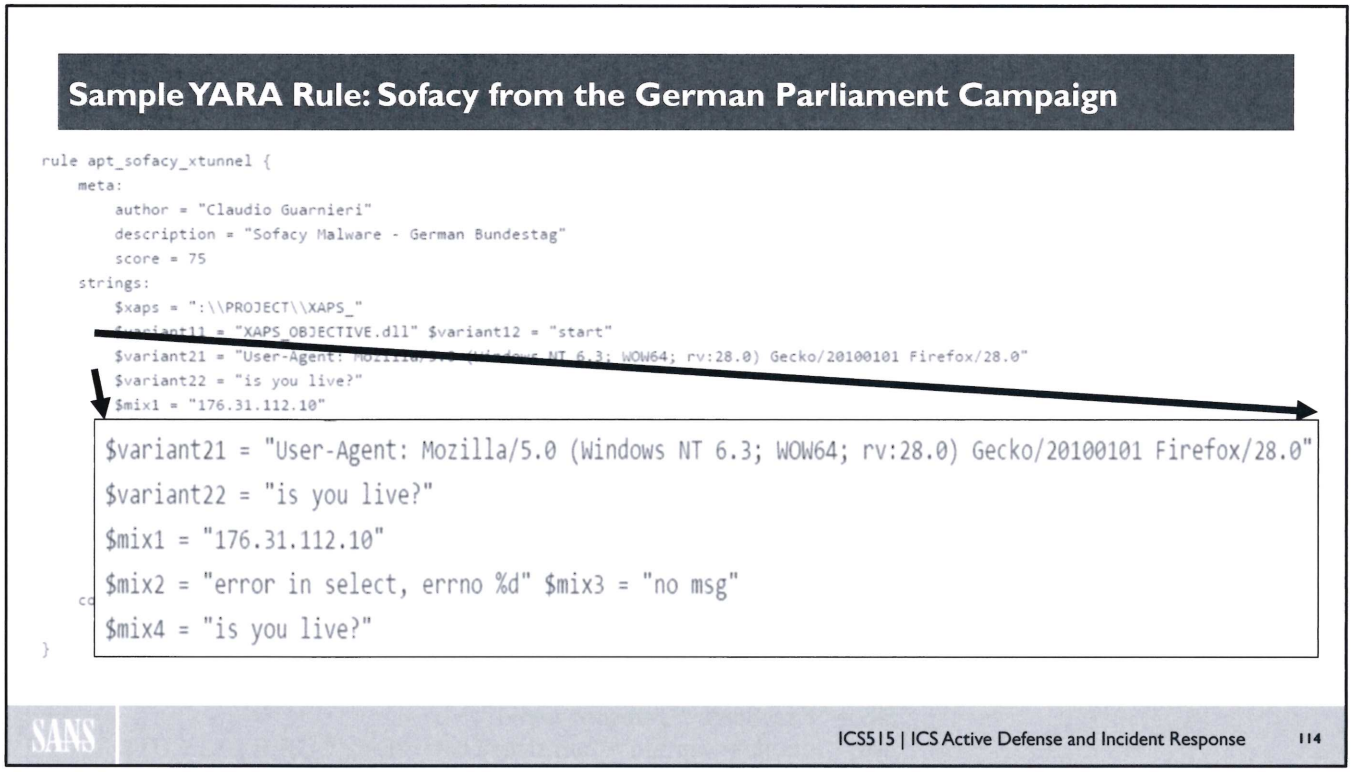


Sample YARA Rule: Sofacy

Here, we see a more complex condition in a rule for Sofacy written by Florian Roth. In this example, there is a validation that the file is an executable and is smaller than 500KB, then it is looking for 1 of the $x variables such as the strings out of the malware or it's ignoring the file save and PE header and looking for 1 of the $x variables and 3 of the $s strings and commands.

```
/*
   This Yara rule set is under the GNU-GPLv2 license (http://www.gnu.org/licenses/gpl-2.0.html) and
open to any user or organization, as long as you use it under this license.

*/
/*
                Yara Rule Set
                Author: Florian Roth
                Date: 2016-02-13
                Identifier: Sofacy Fysbis
*/

rule Sofacy_Fybis_ELF_Backdoor_Gen1 {
                meta:
                                description = "Detects Sofacy Fysbis Linux
Backdoor_Naikon_APT_Sample1"
```

```
author = "Florian Roth"
                reference = "http://researchcenter.paloaltonetworks.com/2016/02/a-
look-into-fysbis-sofacys-linux-backdoor/"
                date = "2016-02-13"
                score = 80
                hash1 =
"02c7cf55fd5c5809ce2dce56085ba43795f2480423a4256537bfdfda0df85592"
                hash2 =
"8bca0031f3b691421cb15f9c6e71ce193355d2d8cf2b190438b6962761d0c6bb"
        strings:
                $x1 = "Your command not writed to pipe" fullword ascii
                $x2 = "Terminal don`t started for executing command" fullword ascii
                $x3 = "Command will have end with \\n" fullword ascii

                $s1 = "WantedBy=multi-user.target' >> /usr/lib/systemd/system/"
fullword ascii
                $s2 = "Success execute command or long for waiting executing your
command" fullword ascii
                $s3 = "ls /etc | egrep -
e\"fedora*|debian*|gentoo*|mandriva*|mandrake*|meego*|redhat*|lsb-*|sun-*|SUSE*|release\"" fullword
ascii
                $s4 = "rm -f /usr/lib/systemd/system/" fullword ascii
                $s5 = "ExecStart=" fullword ascii
                $s6 = "<table><caption><font size=4 color=red>TABLE EXECUTE
FILES</font></caption>" fullword ascii
        condition:
                ( uint16(0) == 0x457f and filesize < 500KB and 1 of ($x*) ) or
                ( 1 of ($x*) and 3 of ($s*) )
}

rule Sofacy_Fysbis_ELF_Backdoor_Gen2 {
        meta:
                description = "Detects Sofacy Fysbis Linux Backdoor"
                author = "Florian Roth"
                reference = "http://researchcenter.paloaltonetworks.com/2016/02/a-
look-into-fysbis-sofacys-linux-backdoor/"
                date = "2016-02-13"
                score = 80
                hash1 =
"02c7cf55fd5c5809ce2dce56085ba43795f2480423a4256537bfdfda0df85592"
                hash2 =
"8bca0031f3b691421cb15f9c6e71ce193355d2d8cf2b190438b6962761d0c6bb"
                hash3 =

"fd8b2ea9a2e8a67e4cb3904b49c789d57ed9b1ce5bebfe54fe3d98214d6a0f61"
        strings:
                $s1 = "RemoteShell" ascii
                $s2 = "basic_string::_M_replace_dispatch" fullword ascii
                $s3 = "HttpChannel" ascii
        condition:
                uint16(0) == 0x457f and filesize < 500KB and all of them
}
```



Sample YARA Rule: Sofacy from the German Parliament Campaign
Here, we have a YARA rule written by Claudio Guarnieri, which identifies the specific variant of malware Sofacy was using when it targeted the German Government (or Bundestag). Notice that the signature is looking for specific user-agents and strings in the malware such as "is you live?". Identifying things like broken English or other languages as well as misspelled words and specific structuring of commands can be a great way to identify a piece of malware; adding in specific user-agents and strings around that variant can further help to eliminate false positives.

```
/*
   This Yara rule set is under the GNU-GPLv2 license (http://www.gnu.org/licenses/gpl-2.0.html) and
open to any user or organization, as long as you use it under this license.

*/
rule apt_sofacy_xtunnel {
   meta:
      author = "Claudio Guarnieri"
      description = "Sofacy Malware - German Bundestag"
      score = 75
   strings:
      $xaps = ":\\PROJECT\\XAPS_"
      $variant11 = "XAPS_OBJECTIVE.dll" $variant12 = "start"
      $variant21 = "User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:28.0) Gecko/20100101
Firefox/28.0"
```

```
$variant22 = "is you live?"
      $mix1 = "176.31.112.10"
      $mix2 = "error in select, errno %d" $mix3 = "no msg"
      $mix4 = "is you live?"
      $mix5 = "127.0.0.1"
      $mix6 = "err %d"
      $mix7 = "i`m wait"
      $mix8 = "hello"
      $mix9 = "OpenSSL 1.0.1e 11 Feb 2013" $mix10 = "Xtunnel.exe"
   condition:
      ((uint16(0) == 0x5A4D) or (uint16(0) == 0xCFD0)) and (($xaps) or (all of ($variant1*)) or (all of
($variant2*)) or (6 of ($mix*)))
}

rule Sofacy_Bundestag_Winexe {
   meta:
      description = "Winexe tool used by Sofacy group in Bundestag APT"
      author = "Florian Roth"
      reference = "http://dokumente.linksfraktion.de/inhalt/report-orig.pdf"
      date = "2015-06-19"
      hash = "5130f600cd9a9cdc82d4bad938b20cbd2f699aadb76e7f3f1a93602330d9997d"
      score = 70
   strings:
      $s1 = "\\\\.\\pipe\\ahexec" fullword ascii
      $s2 = "implevel" fullword ascii
   condition:
      uint16(0) == 0x5a4d and filesize < 115KB and all of them
}

rule Sofacy_Bundestag_Mal2 {
   meta:
      description = "Sofacy Group Malware Sample 2"
      author = "Florian Roth"
      reference = "http://dokumente.linksfraktion.de/inhalt/report-orig.pdf"
      date = "2015-06-19"
      hash = "566ab945f61be016bfd9e83cc1b64f783b9b8deb891e6d504d3442bc8281b092"
      score = 70
   strings:
      $x1 = "PROJECT\\XAPS_OBJECTIVE_DLL\\" ascii
      $x2 = "XAPS_OBJECTIVE.dll" fullword ascii

      $s1 = "i`m wait" fullword ascii
   condition:
      uint16(0) == 0x5a4d and ( 1 of ($x*) ) and $s1
}
```

Course Scenario

This page intentionally left blank.

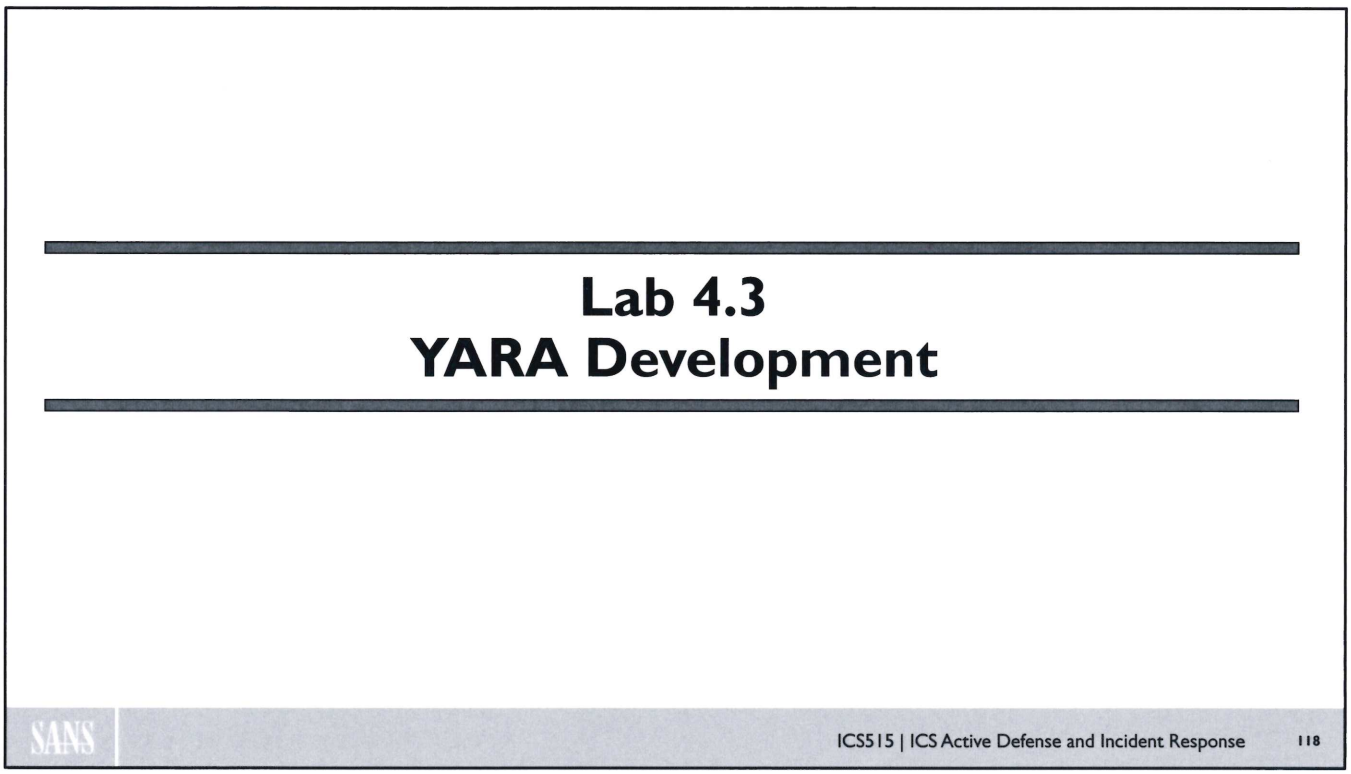
YARA Development

The incidents at the CWU and at the Traffic Lights network are obviously entirely unrelated. Sharing indicators between these cases would be useless. However, creating a few indicators to help scope the environment would be appropriate.

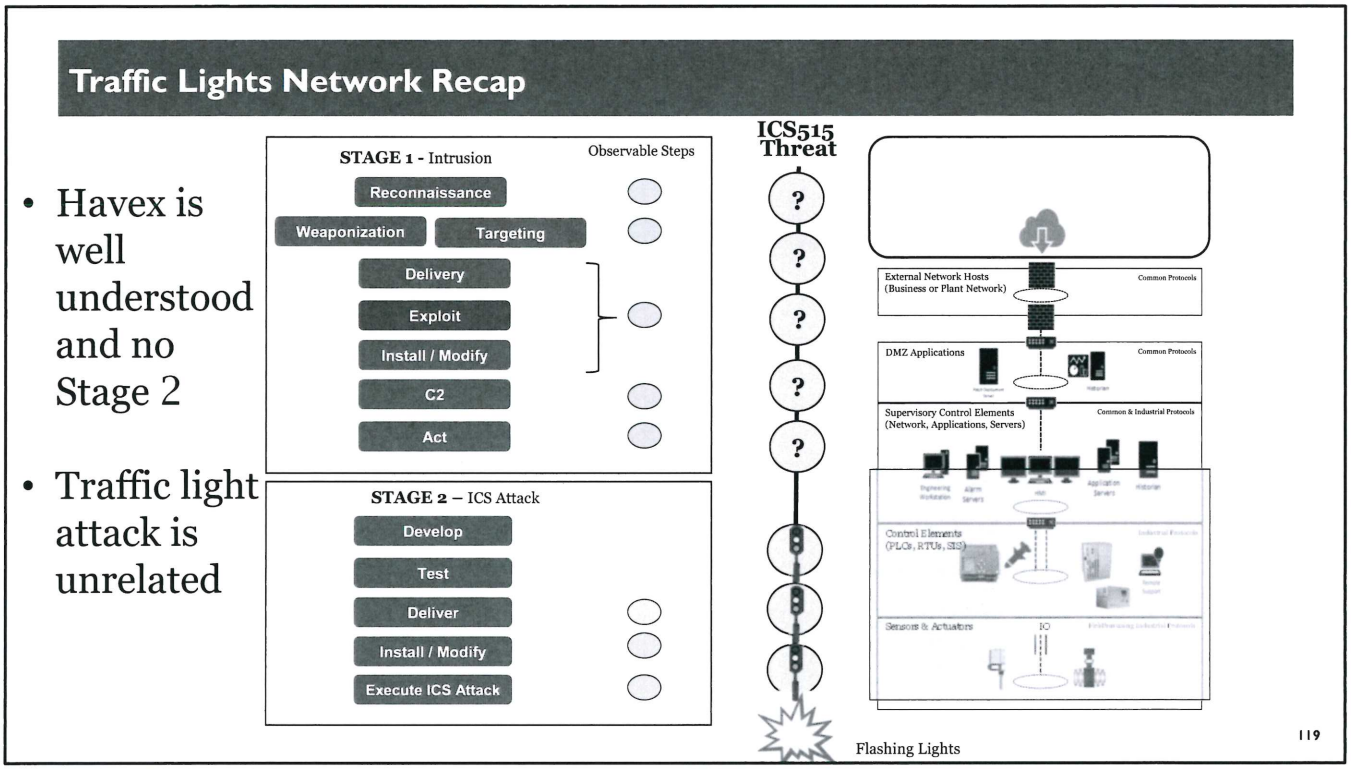
The attack on the CWU network was a SCADA Hijack. The best recommendations are additional network monitoring with analytics to detect this behavior and two-form authentication on the communications in and out of the network.

An indicator has already been made for the malware in the CWU plant network; therefore, an indicator is only needed for Havex for the city.

This page intentionally left blank.

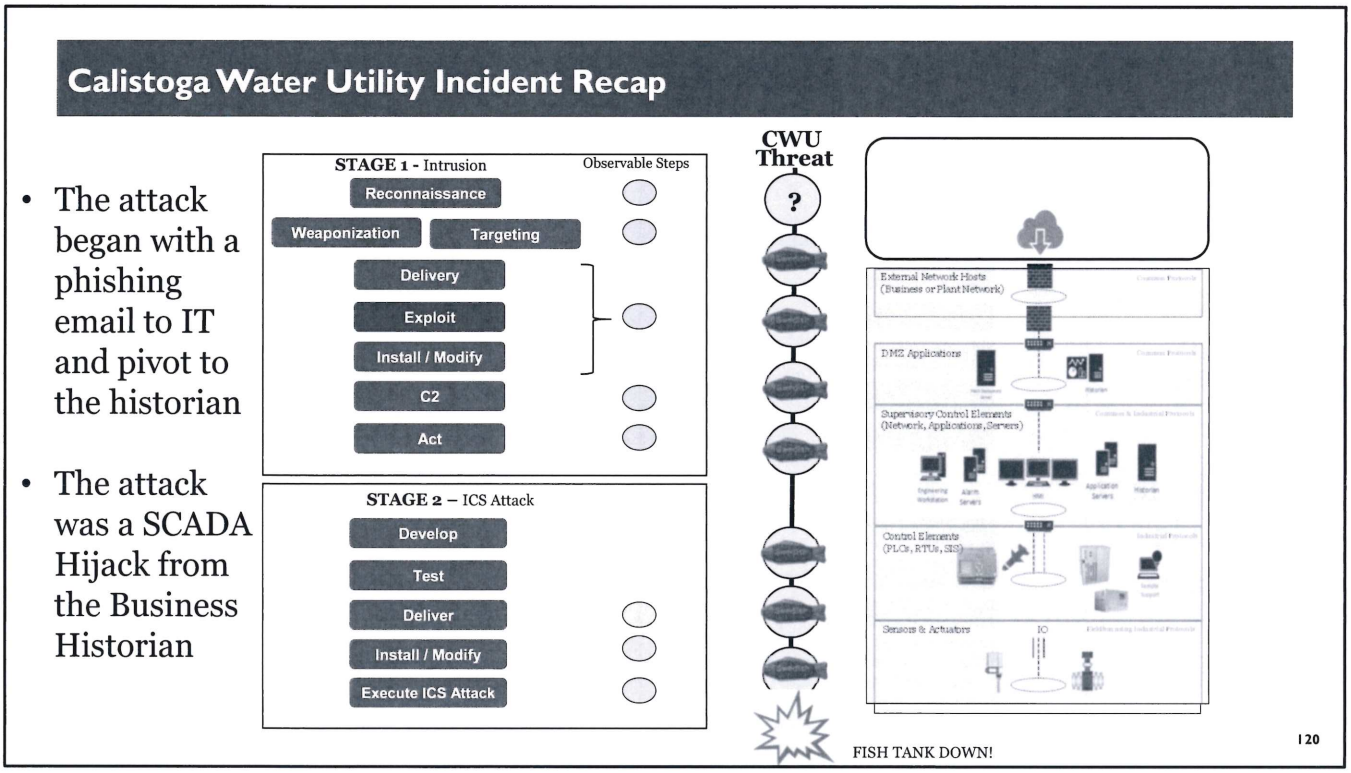


Lab 4.3: YARA Development
Reference the Lab Workbook.



Traffic Lights Network Recap

Havex was in our environment and, from all appearances, it wasn't delivered via phishing emails but likely USB transfer. We would want to check for trojanized installers and Internet-connected websites as well from what we understand about the intrusion. But the intrusion wasn't an attack; it was only espionage and Stage 1. The Traffic Lights flashing was due to a modified project file specific to our environment. Because it was specific, we are reasonably confident that the adversary at some point had been in our environment with Stage 1. If the timing lined up correctly, Havex may have been a means to collect that data; however, in this environment, we know the timing didn't line up where an adversary would have had time to do anything. The project file was obtained by the engineer. It is apparent that there's a different threat maybe still present in our environment that stole information needed to craft the project file that the engineer eventually used. This would be cause to not just link Havex and the project file together and realize that we have two separate threats. With an understanding now of the environment, though, our personnel could clean up the Havex infection, scope the environment to ensure success with our YARA rules, and then start looking for indications of activity related to the Traffic Light.



Calistoga Water Utility Incident Recap

We know through our investigation that there was a random infection in the ICS that was unrelated to the attack. The attack itself began with a phishing email to IT that then gained access and pivoted to the historian that connected IT to the ICS. The attacker used the RDP available from the IT historian to hijack the SCADA environment and perform the attack.

Course Scenario Goals – Course Closure

- Water Plant:
  - ~~Make a topology map of the water utility's plant network.~~
  - ~~What is the root cause analysis of the attack on the water plant?~~
  - What is the intrusion and infection vector that led to the attack?
  - ~~Are there any other intrusions and~~ Are they related to the attack?
- Traffic Lights:
  - What is the root cause analysis of the attack on the Traffic Lights network?
  - Are there any other intrusions and are they related to the attack?
- General:
  - ~~What activity groups could reasonably target the ICS represented in the city?~~
  - ~~Would it be useful to share indicators from one attack to the other?~~
  - What lessons learned would be useful to share between the organizations?

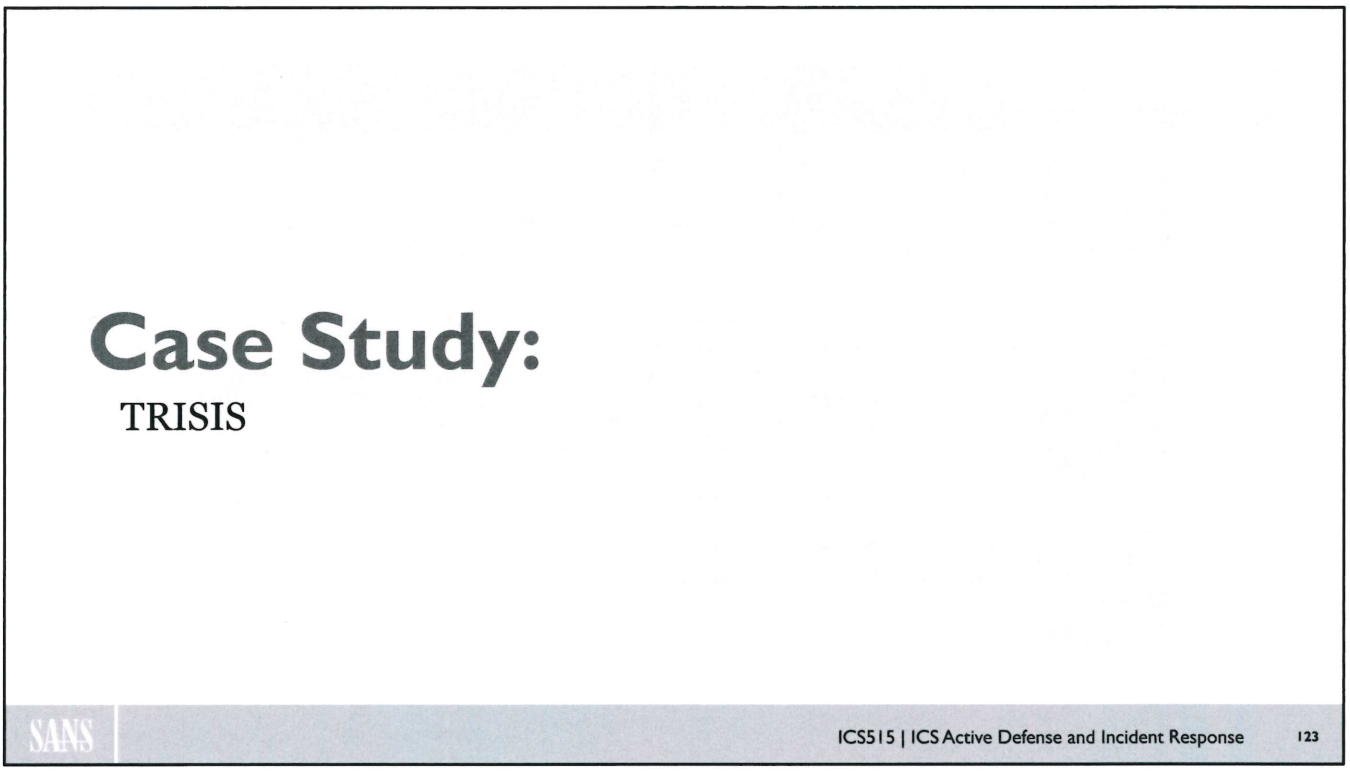
Course Scenario Goals – Course Closure

Identifying connections such as RDP and VPNs between the IT and ICS networks as well as any in the ICS that might connect elsewhere would be a key finding. Additionally, security practitioners could try to limit such communications or enforce 2 form authentication. In addition, monitoring for tradecraft associated with usage of legitimate communications and services that are used in a malicious way would be a good additional detection capability.

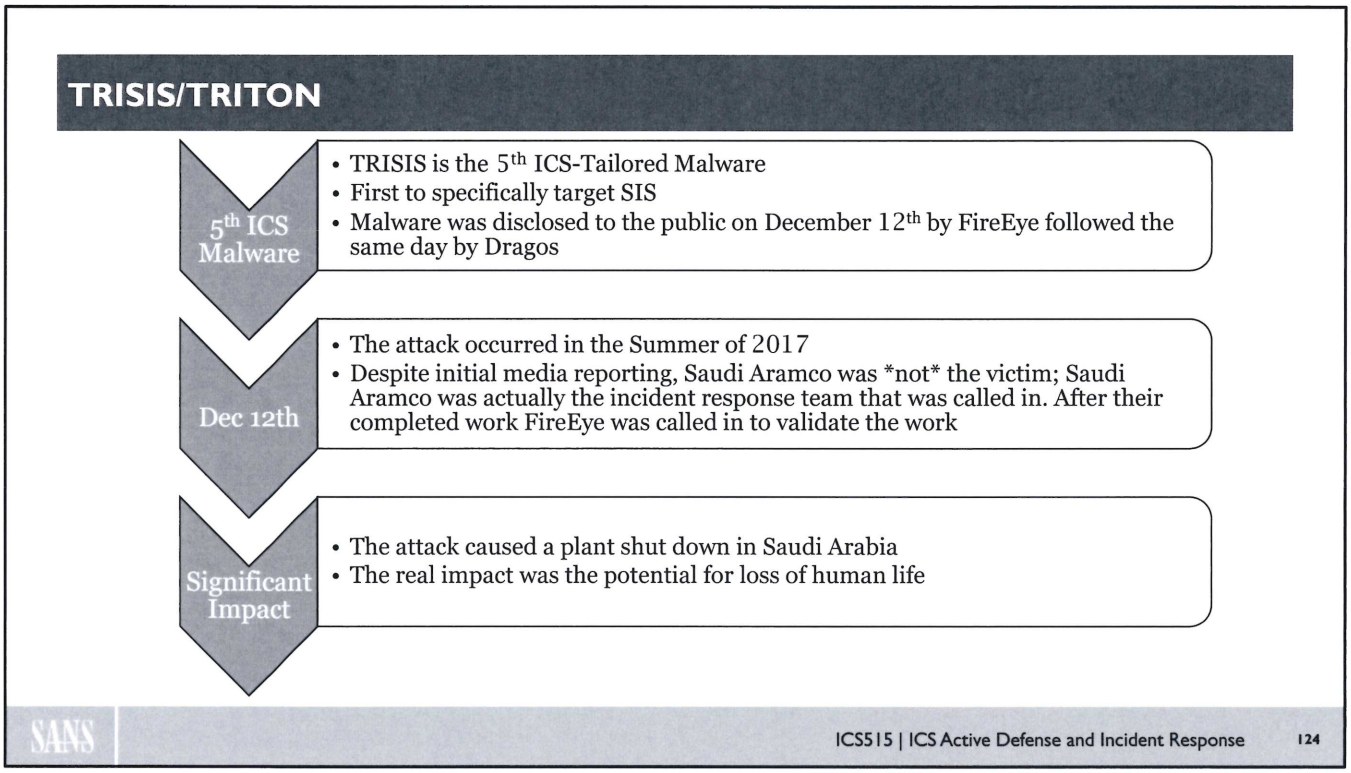
You Win!

- Root cause analysis allowed defenses to be put in place for the future
- No time was wasted on indicators that didn't need to be shared and used
- Other intrusions were found and cleaned up
- Mean time to recovery was maintained at an acceptable level

Congrats. You rock! You worked through complex scenarios to ensure that no foreign powers tampered with the water utility, ensuring success at the NAC. The aquarium took 2nd place in the competition—but that was because George forgot to clean a tank (dammit George!) but not because of any outages or issues. Well done, team.



This page intentionally left blank.



TRISIS/TRITON

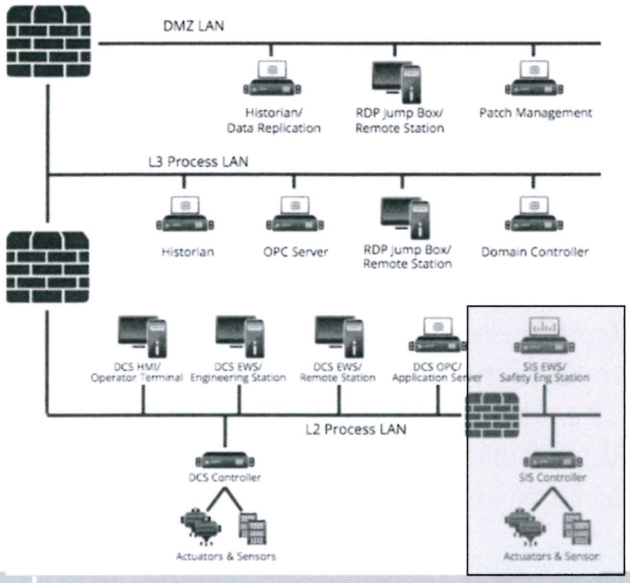
TRISIS, also known as TRITON, is the fifth family of ICS-tailored malware and was used to attack a safety instrumented system (SIS) in the Middle East. It is the first piece of malware that was specifically designed to target human life although the attack failed and caused an operational outage instead. The public community learned about TRISIS on December 12th 2018 when FireEye released their report on TRITON immediately followed by a report by Dragos, Inc. with their naming convention of TRISIS.

Both firms independently found and analyzed the malware but worked to inform customers and the community privately ahead of the public disclosure. Once FireEye moved forward with a public disclosure Dragos did as well with their analysis. The common name used for the malware is TRISIS largely because TRITON was already a malware family name from years ago though FireEye did not realize that until after they named this capability.

References:

https://www.fireeye.com/blog/threat-research/2017/12/attackers-deploy-new-ics-attack-framework-triton.html
https://dragos.com/blog/trisis/TRISIS-01.pdf

Safety Instrumented Systems



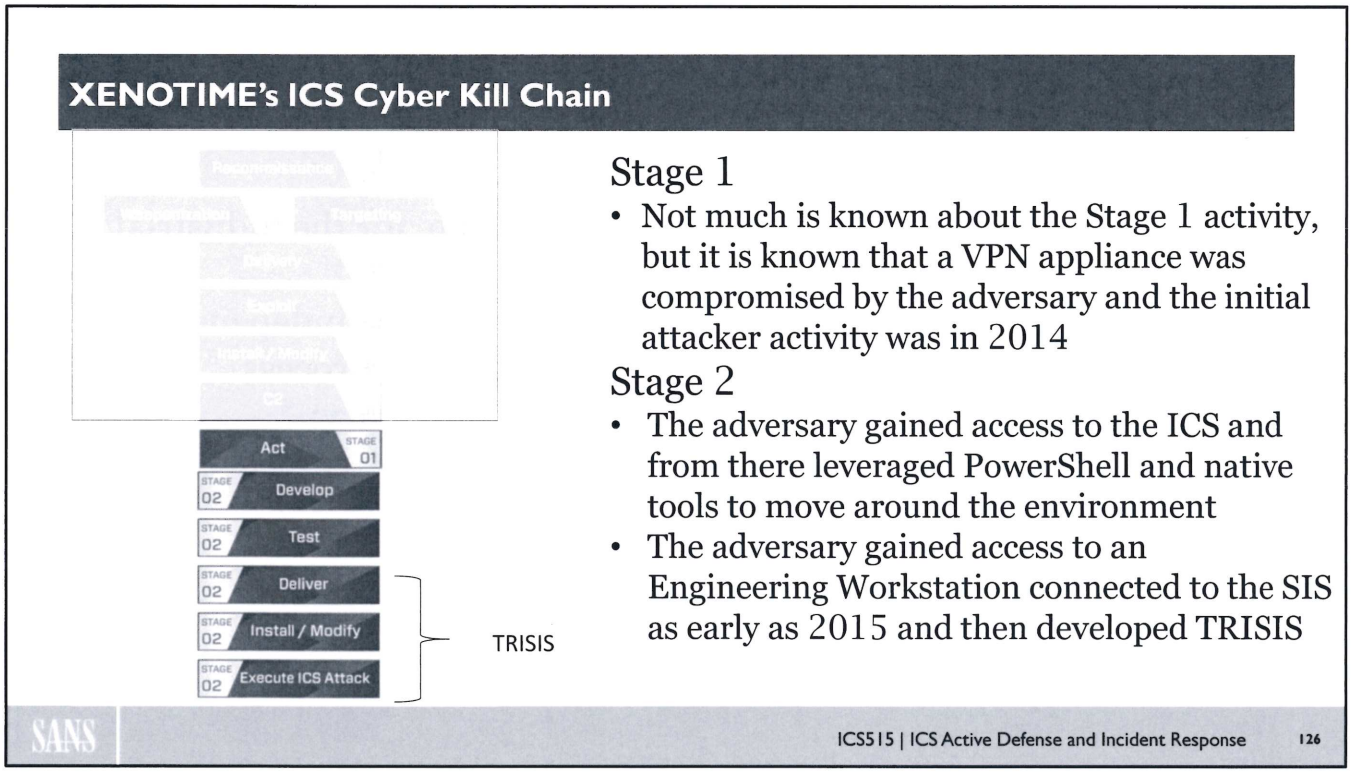
- Failsafe for the industrial process
- **Should** be independent of industrial process
- Not arbitrary:
  - Hazard / Operability Studies
  - Process Hazard Analysis
  - FMEA

SANS ICS515 | ICS Active Defense and Incident Response 125

Safety Instrumented Systems (SIS)
SIS are purpose built systems put in place after a process hazard analysis is done on the specific process it is trying to make safe. It is a specific form of engineering and is very thorough. Therefore an attack against one SIS is not instantly scalable to an attack against another. However, the tradecraft on how to impact one serves as a blue print to other adversaries.

The sole purpose of the SIS is to protect human life. Protection systems protect equipment, SIS protect humans. The targeting of the SIS by the adversary either was intentionally designed to kill people or just simply ok with that scenario. Either way it is aggregious.

Image Reference:
Dragos, Inc.

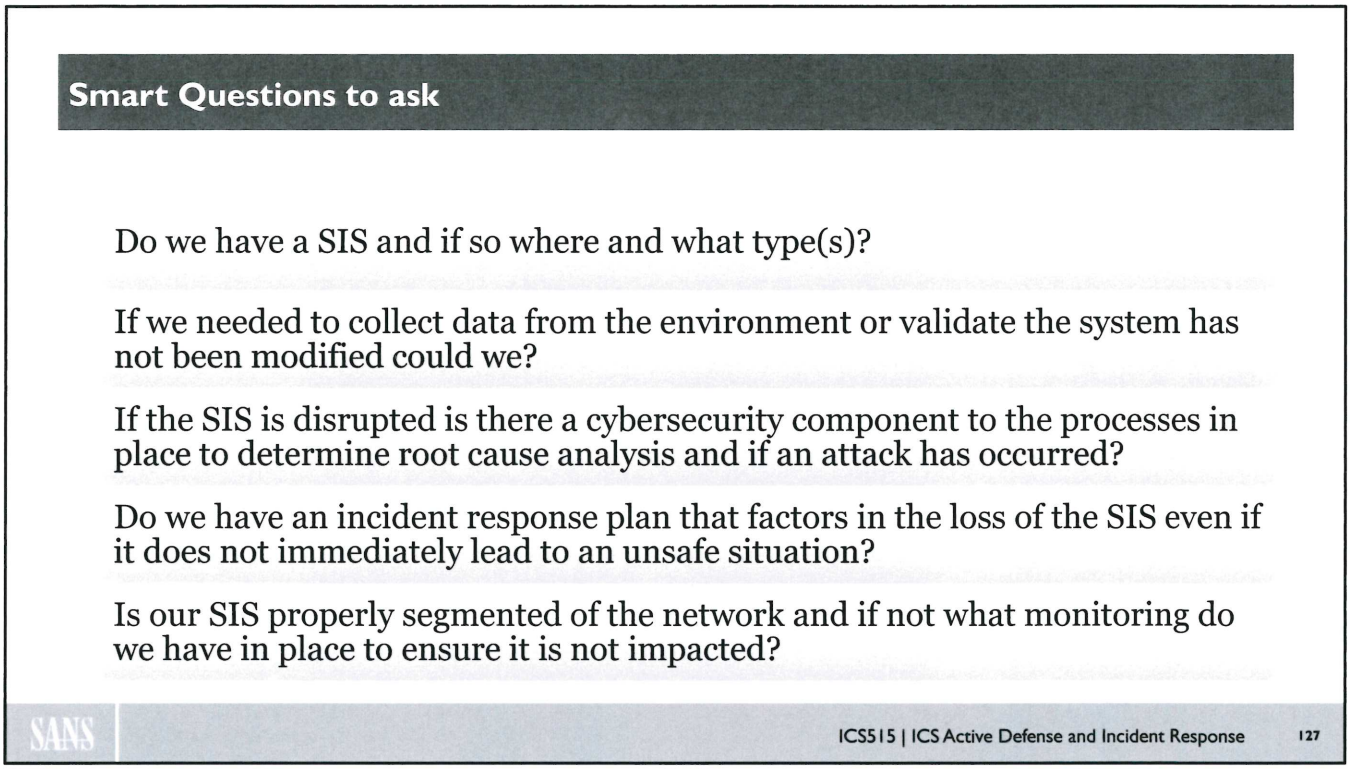


XENOTIME's ICS Cyber Kill Chain
The Activity Group behind TRISIS was discovered by Dragos as XENOTIME. Dragos tracked XENOTIME at the Saudi Arabia based facility but also has confirmed publicly that there are at least 6 other victims of XENOTIME (though no other SIS malware has been deployed yet). The victims are all in the petrochemical and oil and gas industries and compromises of the ICS have taken place in each—including access of network connected SIS. These compromises have been in Europe, the Middle East, and the US.

In the 2017 attack the adversary had access to the Engineering Workstation of the SIS and stole off logic and information about the SIS. The adversary would have then had to have bought or gained access to physical hardware and the appropriate software for the Triconex SIS that was deployed at this site. Following that they would have developed and tested the capability before delivering it to the target. When they re-delivered the capability they disguised the malware as "Trilog.exe" which is a legitimate installer name. From there it ran compiled python scripts which allowed it to speak legitimate Tristation protocols and searches for a location in memory on the Triconex controller to then add a rootkit. This rootkit allows the capability to remove the safety logic and add the adversaries safety logic. In this case it would remove the safety functionality from the SIS meaning no one would know anything had changed. Fortunately for the defenders, the adversaries messed up a portion of the code which caused that specific version of firmware for that specific implementation of Triconex to crash. With the lost of the SIS the rest of the plant shut down; the SIS performed its job of failing safe because the adversary made a coding error.

Image Reference:
Dragos, Inc.

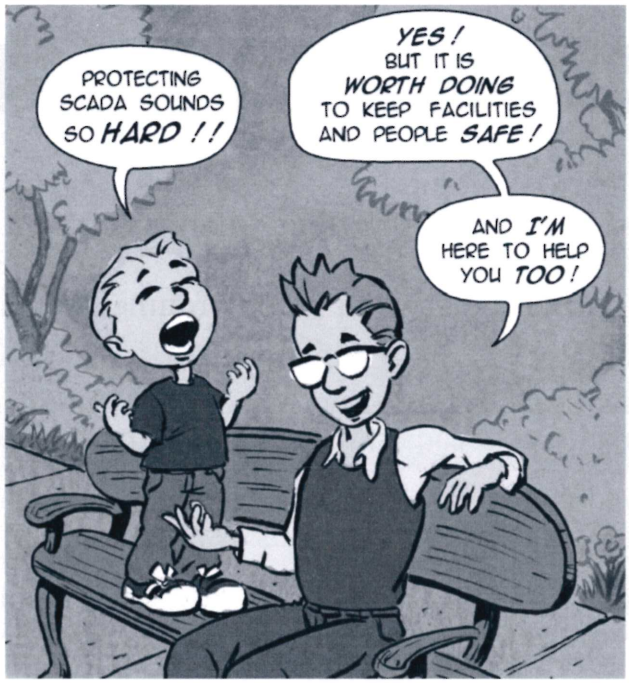
References
https://www.dragos.com/adversaries.html
https://www.dragos.com/blog/20180524Xenotime.html



Smart Questions to Ask
Everyone should always learn from such attacks and determine if they are at risk from this attack or similar styled attacks. Additionally incident response scenarios and plans should be adapted based on learning new adversary tradecraft.

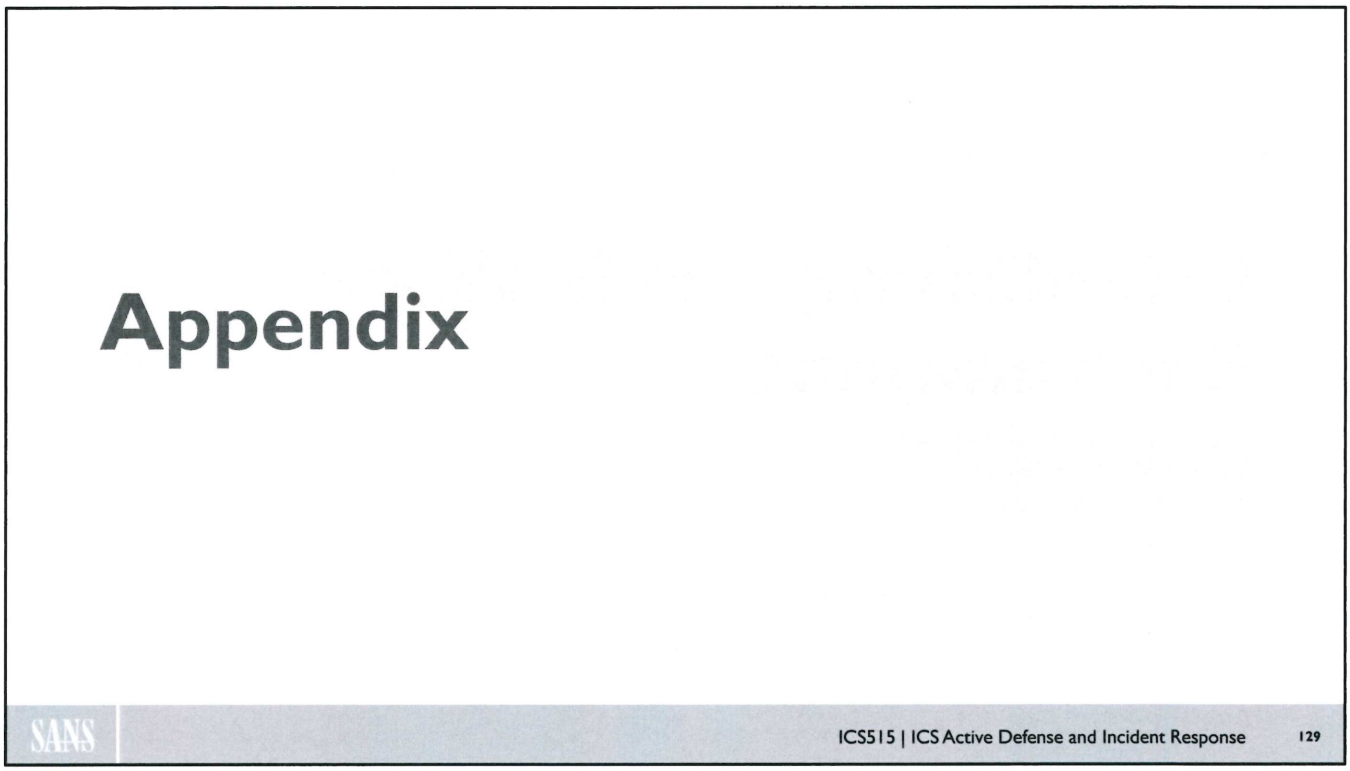
Combining adversary tradecraft from your threat model with an ICS Cyber Kill Chain view of your defensive coverage can help identify gaps in security that can be used to prioritize and recommend new defenses.

END

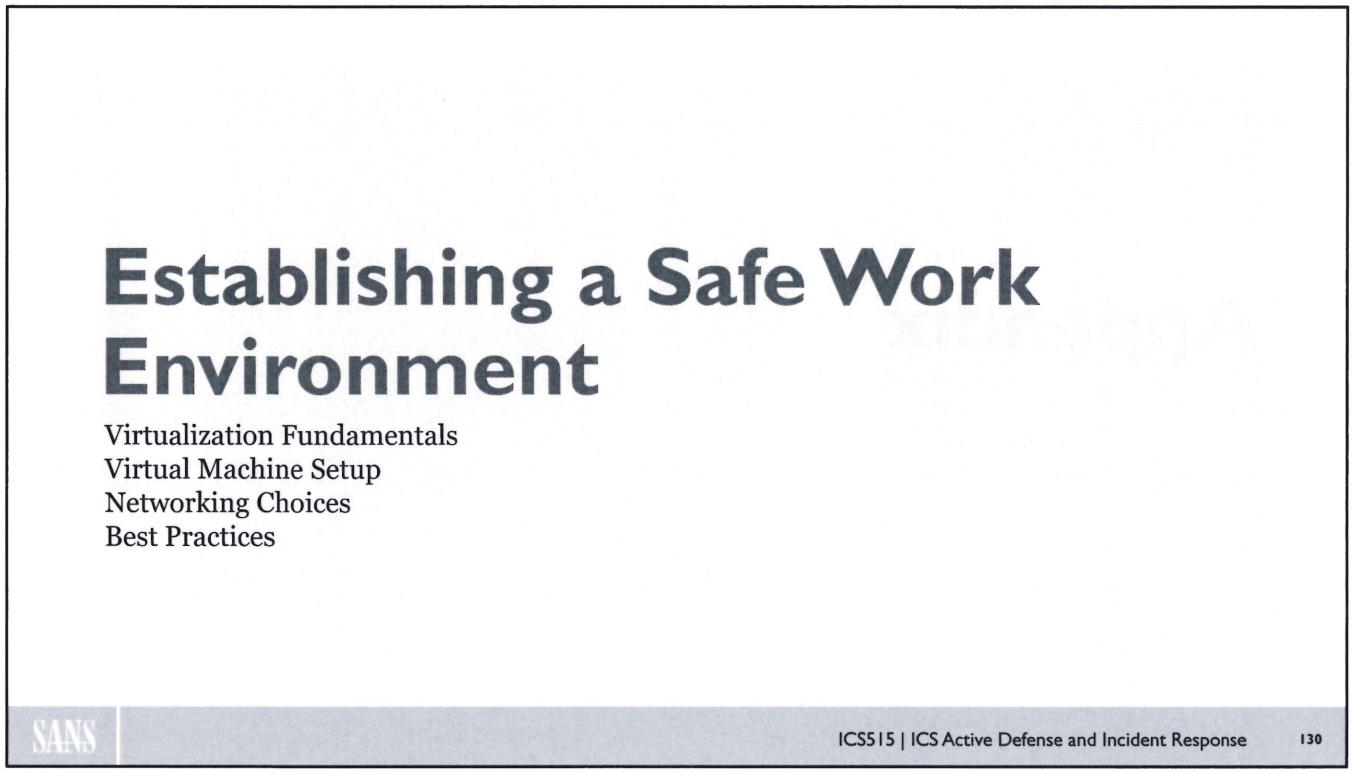


END

That's the end of the slides. Tomorrow, prepare to be tested by combat! Or challenge day—whatever.



Appendix
These slides are for useful notes and additions that may assist students but will not be presented in class.



This page intentionally left blank.

General Malware Analysis in ICS Environments

1. **Establish a safe working environment (for example, use VMs)**
2. Document received evidence files (for example, phishing email, pcap, memory files, and so on)
3. Perform initial analysis to identify useful information (for example, pre-existing YARA rules)
4. Identify and investigate interesting files, executables, and such
5. Extract suspect file(s) from the evidence and document them, including hashes

General Malware Analysis in ICS Environments

This section focuses on the first step: establishing a safe work environment. We've been using VMs throughout the course, but there are a few things to better understand for the purpose of malware analysis.

Virtualization and its Value to TEM

- Virtualization allows Threat and Environment Manipulation (TEM) personnel to build a safe working environment to analyze and run malicious code:
  - Unless the malware has zero-day exploits specifically targeting VMs/hypervisors, it cannot escape the VM
  - Ensure the host computer is not connected to a production network for added safety
- Different types of virtualization software exist, such as:
  - VMware (workstation and server are free, but the code is proprietary)
  - Xen (open source and based in Linux)
  - Microsoft Hyper-V (a feature on Windows Server 2012 and later platforms but not meant as a standalone product)
  - VirtualBox (open source and standalone)

Virtualization and its Value to TEM

Virtualization makes fast analysis of malware possible. Previously to virtualization, the method for performing malware analysis was running malware on systems and then reimaging them every time you needed to reset the system or clean it. VMs are much more efficient.

Zero days for VMs are extremely rare, extremely expensive, and not a normal concern; however, it is best to make sure that the host computer is not connected to a production network for added safety.

Snapshots

- One of the most powerful features of VMs is the snapshot feature
- This enables analysts to install and prepare the VM to an appropriate place and take a "snapshot" that can be reverted to later:
  - For example, prepare an environment, take a snapshot, introduce malware, examine it, and revert to the previous snapshot; this ensures that the machine returns to a clean state and everything is as if the malware was never introduced
  - Snapshots are also useful for saving your work mid-analysis and coming back to it later or testing different tools
- Take snapshots often so that work is not lost and always has a "clean" snapshot
- Snapshots are stored as memory files in the VM manager's directory; they are safe to store even if there is malicious data in them because they are effectively just memory dumps

Snapshots

As mentioned in the previous slide, reimaging physical machines is not quick or good for your sanity. Snapshots allow a system to quickly be reverted to a defined point in time. VMs can have multiple snapshots, allowing them to be flexible for malware analysis.

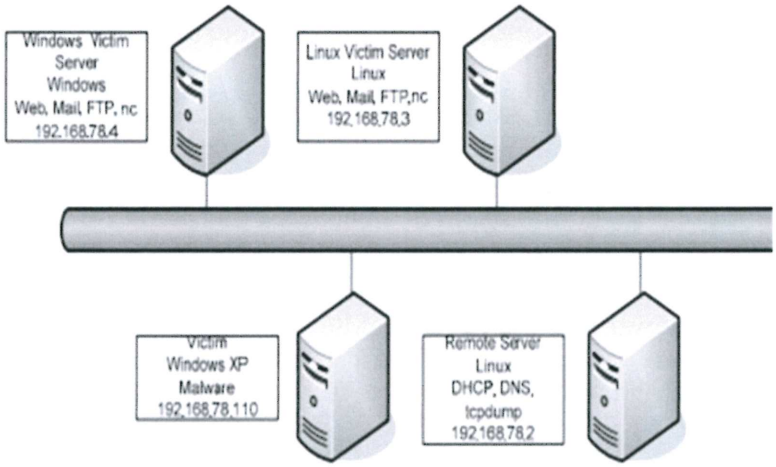
Networking Choices

- Bridged:
  - This causes the VM to share the same NIC as the host system, meaning that as far as the network is concerned, the VM and the host are the same system—obviously a bad choice for malware analysis
- NAT:
  - The Network Address Translation open means that the host's IP address is the routable IP address for the VM
  - The host NIC is responsible for interpreting network traffic and routing it internally to itself or to the VM's private IP
- Host-only:
  - This creates a private network inside of the host environment
  - There is no switch or routing enabled by default, and only systems configured to the same subnet can communicate
  - This is the best option for malware analysis because it gives flexibility to build a virtual network for analysis

Networking Choices

Here is a quick explanation of the networking choices just in case you are interested. They all have their different roles, but for malware analysis the preferred option is host-only networking. You can create a host-only LAN to connect multiple systems either all on the same system or across multiple systems. Using multiple physical machines would require networking gear and a single system is preferred. If you want to have a complex lab setup, using a server with a hypervisor is a good choice.

Example: Sample VM TEM Lab Setup



Reference: "Malware Analysis: An Introduction"

Example: Sample VM TEM Lab Setup
Here is a sample lab setup; all these systems could be running on the same system with each as a VM. They are all on the 192.168.78.x range and thus allow communication between them. Notice that there is a victim system with the malware on it, systems for the malware to communicate with, and a system to perform analysis of the entire session with tcpdump.

Reference:
http://www.sans.org/reading-room/whitepapers/malicious/malware-analysis-introduction-2103

Emulating Control Systems

- Control systems can be emulated through various configurations, scripts, and otherwise fake systems
- Control systems can also be emulated when they are software-based:
  - For example, an HMI installed on a Windows VM can be placed in the malware analysis host-only network and examined
  - For example, an OPC client running in a VM network was used to analyze Havex interactions and OPC scanning
- Honeypots:
  - Honeypots using software such as the Digital Bond SCADA HoneyNet can be introduced to a malware analysis network and simulate services for malware to interact with
  - Adversaries may detect for honeypots, but no ICS malware has been documented to perform such checks before

Emulating Control Systems
In an ICS environment, you need to not only know how a threat might impact a Linux or Windows system, but also how it might impact the control systems. Installing an HMI or OPC client on a system can help determine a threat's interaction or impact on it. Emulating control systems with scripts or software-based honeypots can also provide useful analysis.

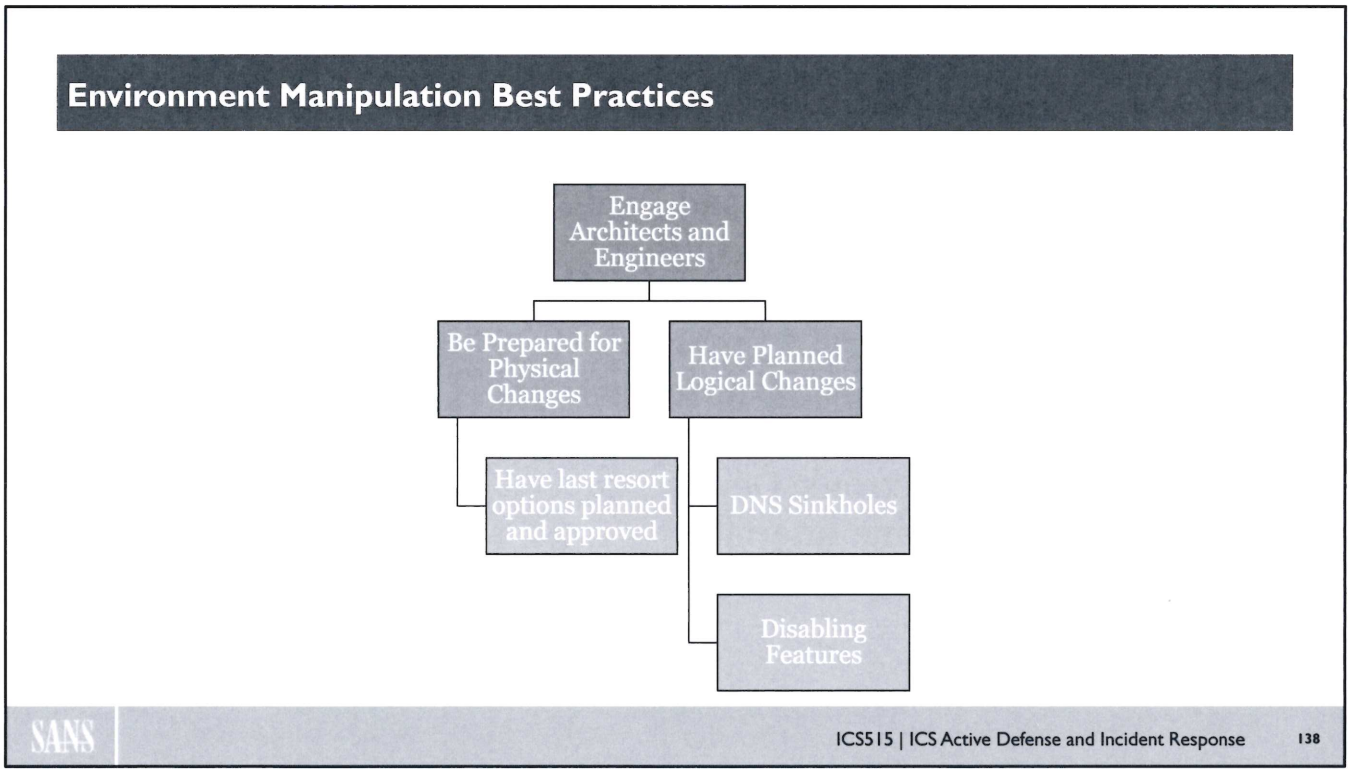
Expendable physical hardware (such as out of lifecycle PLCs) can be networked to a standalone malware analysis network to replicate the control system environment and perform highly accurate examinations of malware and its impact on the physical environment. This is a tactic more useful for analyzing malware impacts on systems than it is in analyzing malware—software will work fine for analyzing the malware itself. Physical hardware such as a Raspberry Pi can also emulate control systems (such as the CYBATIWorks kit).

Reference:
http://phreaklets.blogspot.com/

ICS Acronym List

- SCADA: Supervisory Control and Data Acquisition
- DMS: Distribution Management System
- EMS: Energy Management System
- DCS: Distributed Control System
- PLC: Programmable Logic Controller
- RTU: Remote Terminal (or Telemetry) Unit
- IED: Intelligent Electronic Device
- HMI: Human Machine Interface
- BMS: Building Management System

This page intentionally left blank.



Environment Manipulation Best Practices

We did not cover the concept of manipulating your environment too much today. This is very specific to your ICS. But you should know of some best practices. Basically, you need to be prepared for either physical or logical changes.

Engaging engineers and architects for the ICS and networked environment (IT and OT) have been mentioned multiple times in the course, but it cannot be overstated. Just by engaging these individuals, you can determine requirements, identify knowledge bases, and encourage security across multiple teams. It is critical to prepare for any environment manipulations well in advance of an incident; nothing should occur to the environment without proper planning and testing ahead of time. Practicing environment changes (logical and physical) in mock incidents is a great way to train all personnel involved across ACDC and those involved in passive defense and architecture. Always include management and the decision makers as well.

Physical changes to ICS that can be made generally revolve around the ability to segment systems or provide redundant systems:

- If, through the course of TEM analysis, it is determined that certain types of PLCs are vulnerable to expected threats, recommendations can be made to segment those systems together for NSM to monitor them closely
- In addition, extra safety precautions may be put around systems if available
- Network security is important for safety and should be included in preparation and thought processes

Another type of physical change that is doable in an ICS environment is the ability to physically disconnect from various networked connections:

- This is not a step to be taken lightly and cannot be thought of as true separation
- Make sure your vendors know about this course of action and reserve it for threats that are particularly dangerous and require networked connectivity or have the ability to use VPNs to pivot between the ICS and other sites or vendors
- Physical disconnection requires knowing the chokepoints into the network and disconnecting WAN access

Logical network changes are easier to perform than physical network changes and often can play a role during incidents:

- Most incidents involve malware that communicates with a C2 server (or multiple) for commands, to give access to an adversary, or to exile data.
- Logical network changes can be made at locations such as the switch, router, and firewall to enable MAC filtering, port security, or otherwise enforce stricter policies than would otherwise be acceptable outside of an incident.

Logical network changes take place all the time when firewalls are updated to block certain types of protocols or IP addresses. The understanding that this can be done during an incident is vital:

- There is always a balance in IT between gathering intelligence and blocking an attack. In ICS, there is less of an argument. Gathering intelligence can be great, but it is preferred that this is done in a malware analysis lab and not a live production network for obvious reasons.
- Another effective type of logical network change for environment manipulation that impacts the adversary is DNS sinkholing.

DNS sinkholes are DNS servers that are configured to give out false information:

- When a piece of malware tries to resolve the domain for its C2 server, the DNS can redirect it to an IP address that does not exist so that network connectivity can never occur.
- One technique used by malware to defeat IP blocking is called *fast flux,* which enables malicious sites to quickly use different IP addresses resolved to the same domain, making it harder to block DDoS styled attacks; however, the domain name stays the same thus with a sinkhole is null and void.

DNS sinkholes should be one of the go-to tools for environment manipulation:

- They can be delegated to TEM personnel to maintain because they pose little risk to the environment.
- TEM personnel can maintain DNS sinkholes even against threats that have not impacted the environment yet. When a threat that is already sinkholed is introduced to the ICS, the sinkhole alerts personnel to its presence while pre-emptively neutralizing its primary functions.

TEM analysts can take a sample of the malware and run it in the malware analysis lab to examine it. By modifying the hosts file on the system, it can tell the malware that the domain name for the C2 server is an IP address to an analysis machine:

- This allows TEM personnel to identify the malicious commands that interact with the malware and test kill commands.

Good reading for understanding DNS sinkholes better and a case study of how this was done to neutralize the malware conficker:
http://www.sans.org/reading-room/whitepapers/dns/dns-sinkhole-33523

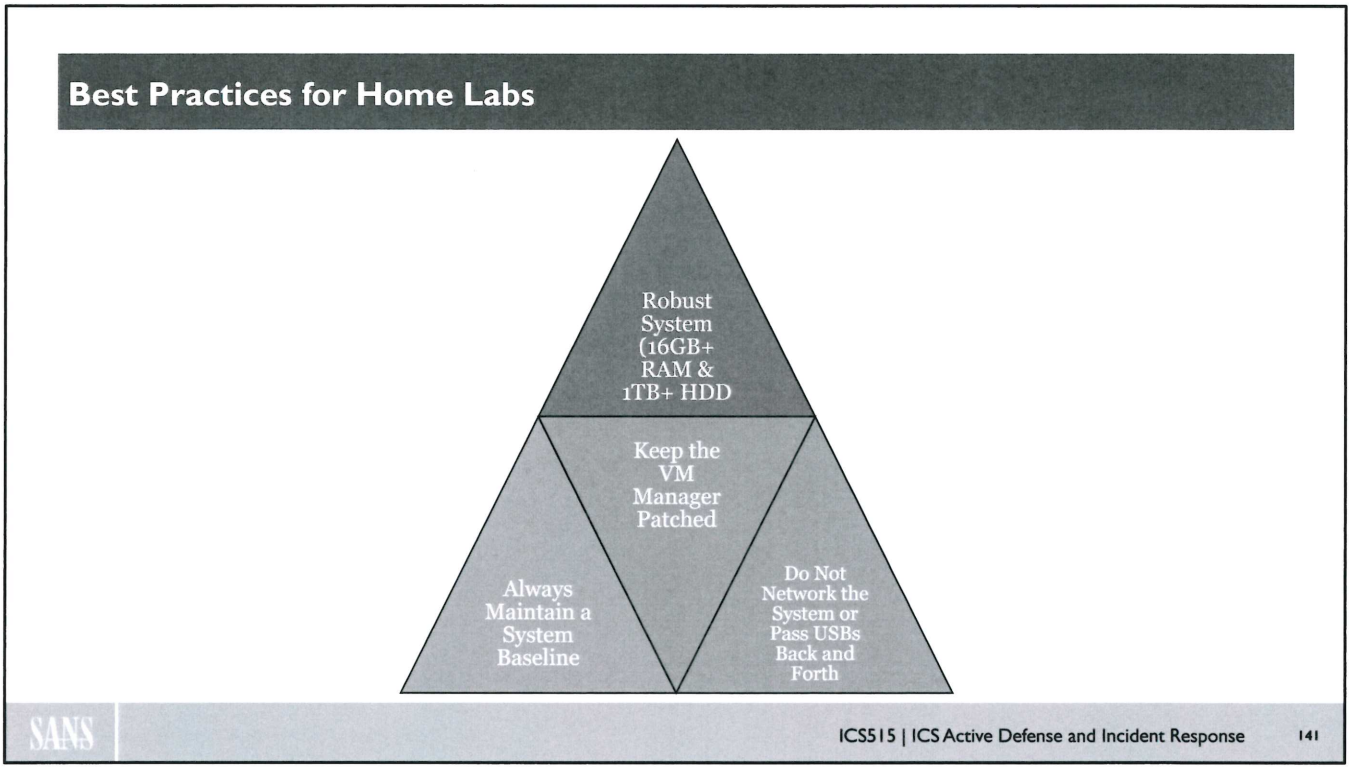
False Information

- False information, or counterintelligence, can also be used in the environment, but there are many concerns with doing so in an ICS environment:
  - It is possible to disguise systems to look like other systems and put up fake services
  - For example, when a piece of malware queries a host and discovers it's a Windows machine, it uses a Windows exploit; but if it's a Linux system disguised as a Windows host, the exploit fails and defenders can set up logging for this
  - The problem is that it complicates asset identification and NSM efforts while introducing the possibility that OT engineers or operators might also become confused (or supervisory systems) and introduce errors
- It's generally not advisable to use false information on the network but be aware that its doable
- False information can more aptly be put out on websites like LinkedIn to counter adversary attempts to look up information, such as the type shown in Day 1

False Information

The real takeaway here is that false information is a viable defense tactic, but generally, it is a bad return on investment for defenders. It can introduce a lot of complications and is meant only for organizations with mature defense postures.

Caution: Creating fake project files and OS spoofing could lead to confusion and operational issues can arise from their improper use.



Best Practices

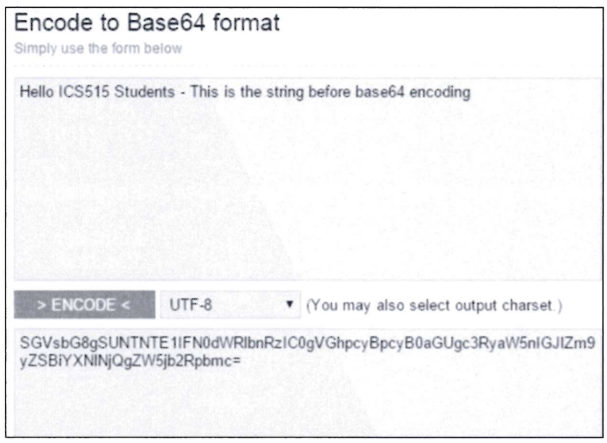
You'll want to ensure that you have the proper hardware and system resources to create a malware analysis lab either on one or multiple systems. Make sure it is networked only to other analysis systems and essentially air gapped (the only time that should really be stated in this course). Be careful on how you move malware and files back and forth between the system and other systems, and never connect the system directly or indirectly to the production network. Firewalls, DMZs, etc. are not enough. Have actual separation.

Because of the robustness and security in VMs, it's fine to use USBs or CDs to transfer in data, but for extra safety, never let it go from a Production network to USB to Malware analysis and back again. (Use a separate network to download and transfer files or use one-time-only USBs or read-only CDs)

Ensure the Hypervisor and the guest VMs remain patched and ensure clean original baseline VMs are available to use in full system restore efforts when things go wrong.

Unraveling Base64

- Various pieces of malware will encode C2 communications and attempt to exfil data off the environment in Base64
- It is not difficult to decode Base64, but it can become complex when various character sets are used or it is encoded multiple times
- Scripting and automation is useful such as MITRE's ChopShop



Unraveling Base64

Base64 is a favorite of many types of malware as it is useful to hiding text strings from analysts. There are many ways to decode malware (see this article for a few types such as XOR and ROT: https://blog.malwarebytes.org/intelligence/2013/03/obfuscation-malwares-best-friend/) but Base64 continues in use especially for strings in exfil.

It is not difficult to decode one Base64 encoding, but malware authors will sometimes use multiple runs of Base64, which can give analysts some difficulty if processing the information by hand or simple tools. For this reason, it's useful to automate and script the functionality out. One of the tools useful for this is MITRE's ChopShop found here: https://github.com/MITRECND/chopshop

ChopShop can be used for a variety of protocols, but it also has a B64.py script that is useful for Base64 decoding.

Extracting Files from Acquired Evidence

- Files can be extracted from network exfil to determine the nature of the threat, but files, and the malware itself, can also be extracted in the same way:
  - The initial delivery or the spreading of malware through network traffic can be a great source to identify the malware
  - When not captured in network traffic, malware can usually be extracted from infected files, logical images of the physical system, and memory dumps
- Many paid industry Incident Response tools have the capability to pull files out of digital evidence; however, two well-tested free methods as well can be useful:
  - NetworkMiner
  - Foremost

Extracting Files from Acquired Evidence
There's a number of ways to pull files from evidence, but two quick tools are NetworkMiner and Foremost.

Foremost

- Free and open source command-line tool run in Linux distributions to carve data out of digital images (physical hard disks or memory)
- Developed by Kris Kendall and Jesse Kornblum of the USAF Office of Special Investigations and is one of the most widely used tools in the Forensic/Incident Response community
- Easy-to-use tool that carves data based on headers, footers, and data structures:
  - Defined in a configuration file and can be modified to include new file types or ignore certain types of files
- The easy customizability makes Foremost perfect for carving data in an ICS environment:
  - Many extensions and file types are proprietary to vendors that can be added to Foremost by simply knowing the header/footer of the file (can be viewed and confirmed in a HEX editor)

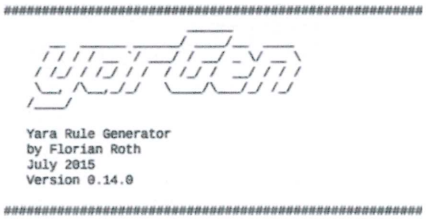
Foremost
Foremost is an extremely common tool to use for extracting files from samples. It can remove files from network captures, memory, and other data sources such as images of physical drives. It extracts files based on the header and footers of files and is perfect for ICS environments because you can tailor what headers and footers it looks for by altering its .conf file.

In the lab, we'll go into this more deeply.

Reference:
http://foremost.sourceforge.net/

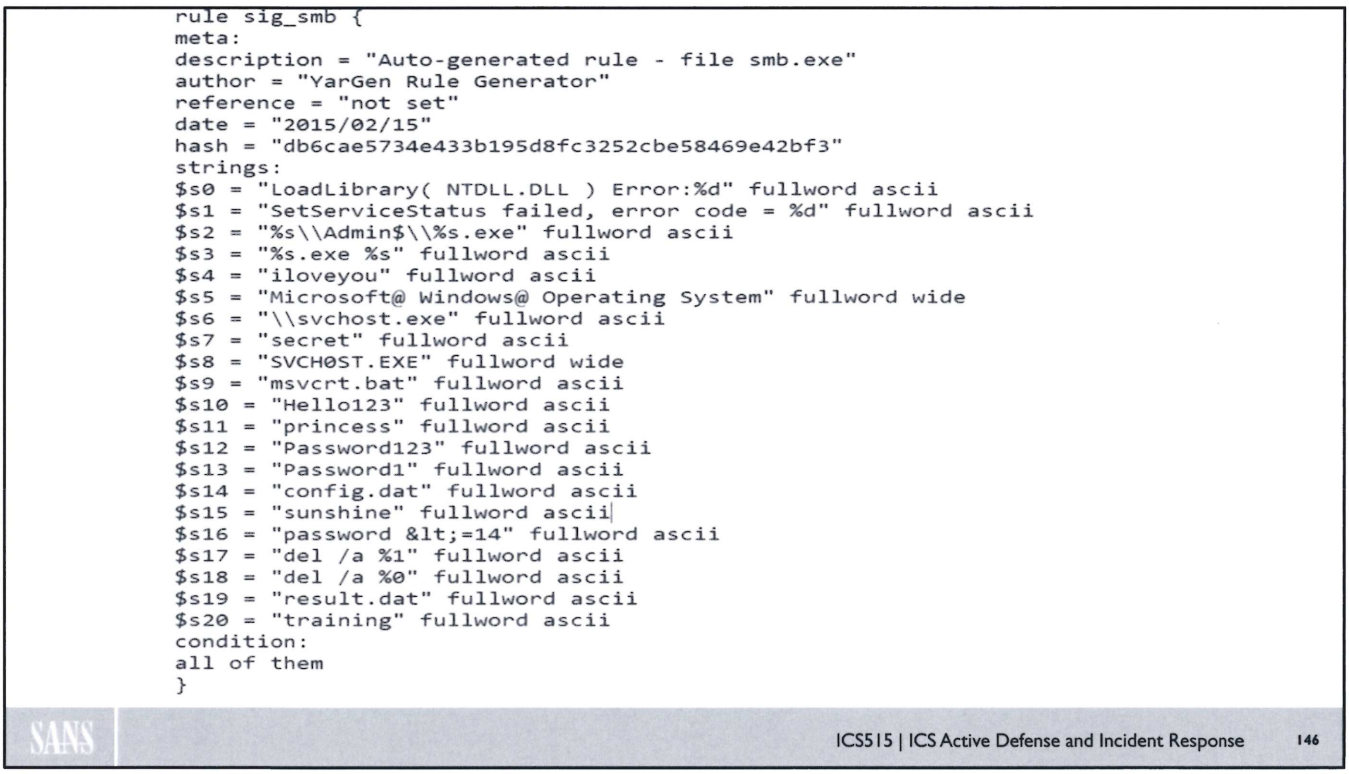
yarGen

- yarGen is a CLI tool
  - Created by Florian Roth
- Takes sample files and creates YARA rules based off of the strings inside the sample
- The rules are often very specific to that sample to reduce false positives
- Useful for automating quick rule creation and searching but not for high-confidence IOCs



yarGen
yarGen was created by Florian Roth and is a great command line interface (CLI) tool to create YARA rules based off of one or more files (generally malware samples). The tool searches through the file for readable strings and creates a YARA rule out of those strings. There are multiple options to allow some flexibility with the tool; one of the hallmarks of the tool is that is it produces a YARA rule with the condition of "all of them" noting that all of the strings must be present. This helps to reduce false positives but often makes the YARA rules extremely specific.

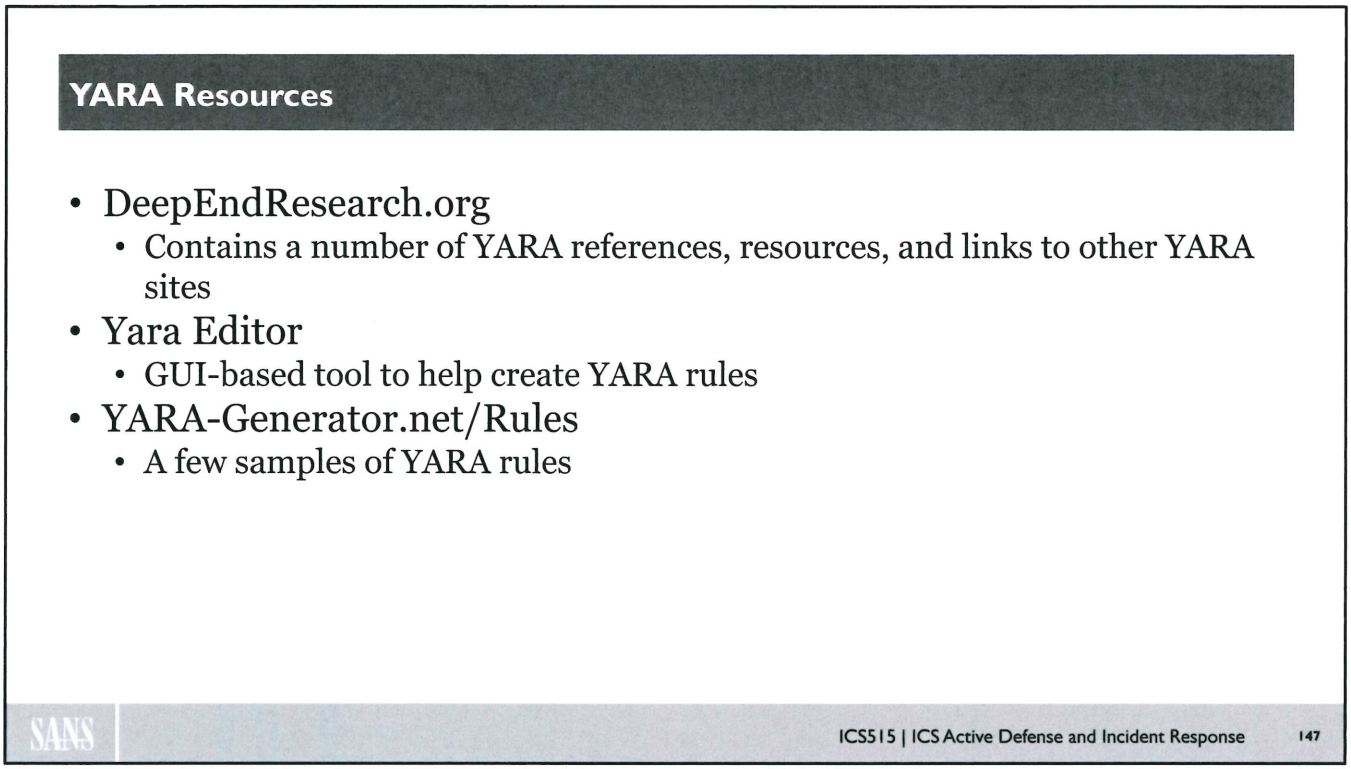
Reference:
https://github.com/Neo23x0/yarGen



Sample yarGen Rule
Here, yarGen was run on a sample of smb.exe (note the description that this is an auto-generated rule). Florian Roth made this rule, which shows some of the focus points of yarGen. First, the options chosen when running yarGen forced each string to require the full word (no partial completions) and ASCII representation of each string. Additionally, as is commonplace with yarGen, it requires the condition of "all of them". This helps keep the false positives to a minimum but note that this rule is only going to work on a variant of this sample that has all the same strings as the original variant that are represented here. I.e. it's good for automated generation to make very specific YARA rules so that you do not have high false positives, but these are likely not good rules to find new variants of the same malware.

You can tailor yarGen to include other fields such as the magic header, file size, and designate a maximum number of strings that are included. Very importantly, you can also exclude good strings. To do this, you create a list of strings and feed it to yarGen when running the tool to exclude good strings such as "Microsoft" which would show up often.

Reference:
https://github.com/Neo23x0/yarGen



YARA Resources
DeepEndResearch.org is a great place to find additional YARA resources. The specific link is: http://www.deependresearch.org/2013/02/yara-resources.html

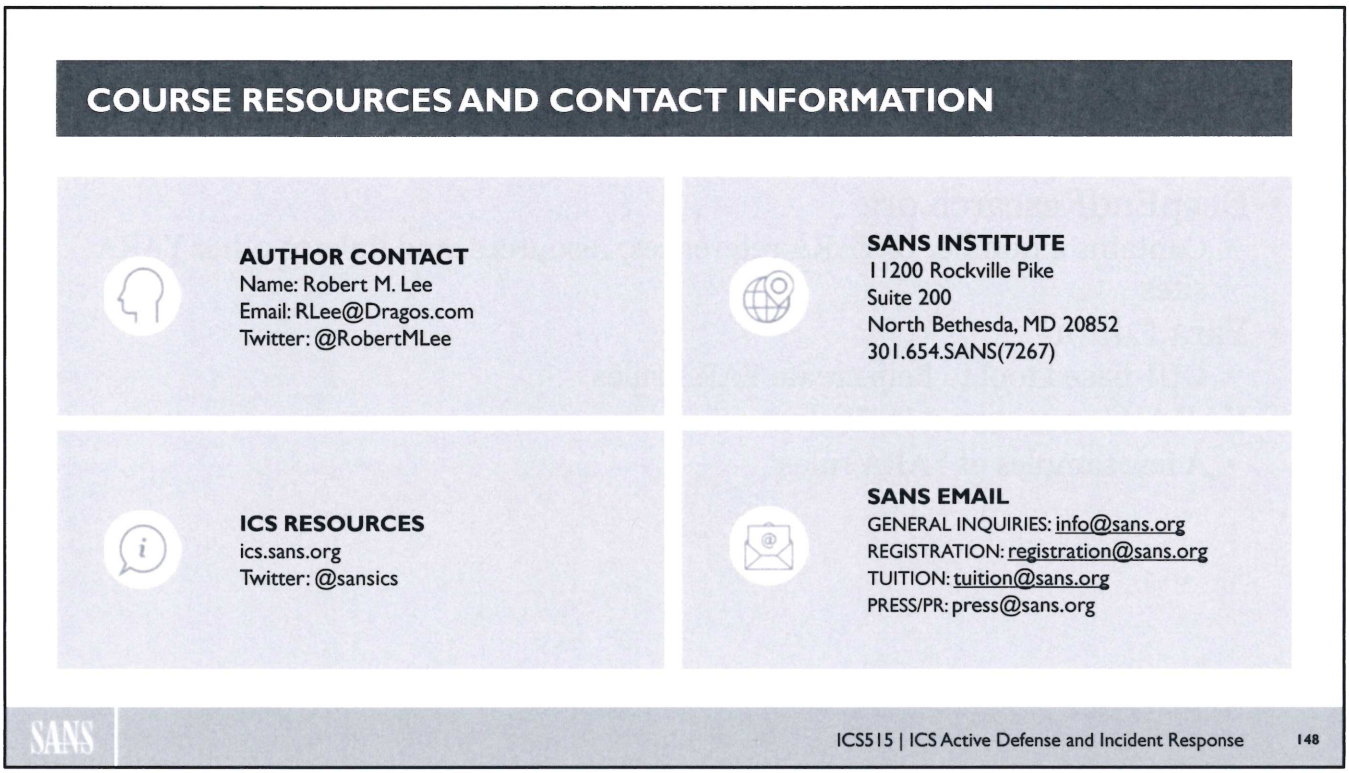
Additionally, SANS conferences are a great place to identify new available resources in the community; more and more analysts are presenting YARA rules tied to their research. For example, the BeEF YARA rules identified earlier were presented at SANS DFIR Prague Summit.

Yara Editor is another great tool to have a GUI to help create YARA rules: https://code.google.com/p/yara-editor/

YARA-Generator.net should not be confused with yaragenerator, but it has a few samples rules available to help analysts understand what others have written before. It's always good to see what's out there already to help you create more new and complex rules: http://www.yara-generator.net/rules

Extensions for YARA
YARA is relatively new and quickly expanding, but there are some extensions that should be considered if you are looking to create more advanced YARA rules. Without going into each extension in depth, be aware that there are extensions to interlink tools such as IDA Pro with YARA. Additionally, yarascan is a plugin for the memory analysis tool Volatility to bring YARA functionality to it. Likewise, there are extensions such as yextend that allows YARA rules to deflate archived content and then scan it. Another useful tool is libyara, which is a library to integrate YARA features into C and C++ projects.

Yextend: https://github.com/BayshoreNetworks/yextend
Libyara: http://yara.readthedocs.org/en/v3.4.0/capi.html
Other less tested tools like YARApcap: https://github.com/kevthehermit/YaraPcap



This page intentionally left blank.

This page intentionally left blank.

This page intentionally left blank.

Index

A

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Y

Z