INCONTROLLER: New State-Sponsored Cyber Attack Tools Target Multiple Industrial Control Systems

Cyber Espionage (CE) | Critical Infrastructure (CI)

Fusion (FS)

April 13, 2022 09:12:08 PM, 22-00008528, Version: 3.0

Executive Summary

- Mandiant Threat Intelligence analyzed a set of novel industrial control system (ICS)-oriented attack tools—which we call INCONTROLLER—built to target specific Schneider Electric and Omron devices that are embedded in different types of machinery leveraged across multiple industries.
- INCONTROLLER is a collection of three separate Python-based frameworks, which we individually track as TAGRUN, CODECALL, and OMSHELL. They contain capabilities related to disruption, sabotage, and potentially physical destruction.
- We are also tracking two additional tools affecting Windows-based systems that may be related to this threat activity: ICECORE and an exploit for CVE-2020-15368.
- INCONTROLLER is very likely state sponsored. We are unable to link the activity to existing clusters of threat activity, but we note that the activity is consistent with Russia's historical interest in ICS.
- This malware poses a critical risk to organizations leveraging the targeted equipment. Organizations should take immediate action to determine if the targeted ICS devices are present in their environments and begin applying vendor-specific countermeasures, discovery methods, and hunting tools, which we describe in this report.

Threat Detail

New Version Details

Version 2, April 13, 2022: This report has been updated with additional appendices containing INCONTROLLER malware analyses.

Introduction

This is a developing issue and Mandiant will update this report or release new content and analysis as it become available.

This report is discussing information related to DOE ARES REPORT ARES-22-0330-01.

In early 2022, Mandiant, in partnership with Schneider Electric, analyzed a set of novel industrial control system (ICS)-oriented attack tools—which we call INCONTROLLER—built to target machine automation devices. The tools can interact with specific industrial equipment embedded in different types of machinery leveraged across multiple industries. While the targeting of any operational environments using this toolset is unclear, the malware poses a critical risk to organizations leveraging the targeted equipment. INCONTROLLER is very likely state sponsored and contains capabilities related to disruption, sabotage, and potentially physical destruction.

INCONTROLLER represents an exceptionally rare and dangerous cyber attack capability. It is comparable to TRITON, which attempted to disable an industrial safety system in 2017; INDUSTROYER, which caused a power outage in Ukraine in 2016; and STUXNET, which sabotaged the Iranian nuclear program around 2010 (21-00018084). To help asset owners find and defend against INCONTROLLER, we have included a range of mitigations, discovery methods, and hunting tools throughout this report. As future modifications to these tools are likely, we believe behavior-based hunting and detection methods will be most effective.

INCONTROLLER is comprised of three main Python-based components:

Tool	Description
TAGRUN	A tool that utilizes an open-source OPC UA Python library to scan for OPC servers, enumerate OPC structure/tags, brute force credentials, and write OPC tag values.
CODECALL	A framework that communicates using Modbus—one of the most common industrial protocols. CODECALL contains modules to interact with, scan, and attack at least three Schneider Electric programmable logic controllers (PLCs).
OMSHELL	A framework with capabilities to interact with and scan some types of Omron PLCs. The tool can also interact with Omron's servo drives, which use feedback control to deliver energy to motors for precision motion control.

Table 1: Description of INCONTROLLER tools

INCONTROLLER Was Built to Manipulate and Disrupt Industrial Processes

Industrial automation networks rely on a variety of equipment that enable operators to translate information and instructions into chains of physical actions. Given the diversity of assets present in industrial networks, industrial automation equipment typically speaks different languages across different portions of the network, which is possible using standardized industrial communication protocols.

INCONTROLLER includes three tools that enable the attacker to send instructions to ICS devices using industrial network protocols, such as OPC UA; Modbus; Codesys, which is used by EcoStruxure Machine Expert and SoMachine; and Omron FINS. While the tool's capabilities could enable the actor to communicate with a variety of products from different original equipment manufacturers (OEMs), the actor developed modules for specific controllers from Schneider Electric and Omron. The targeted equipment consists of machine automation solutions whose use cases span from supporting simple, repetitive machines to complex modular machines in distributed architectures:

- OPC servers
- Schneider Electric Modicon M251, Modicon M258, and Modicon M221 Nano PLCs
  - Other devices leveraging Modbus and Codesys may also be affected
- Omron NX1P2 and NJ501 PLCs and R88D-1SN10F-ECT servo drive
  - Other devices from NJ and NX PLC series may also be affected

We highly doubt that the threat actor would target these devices at random. It is more likely they were chosen because of reconnaissance into specific target environment(s). We note that this would be consistent with previous ICS malware, such as TRITON, which targeted a critical safety system that was almost certainly identified prior to compromising the target's industrial environment (21-00026094).

INCONTROLLER: Tooling Overview

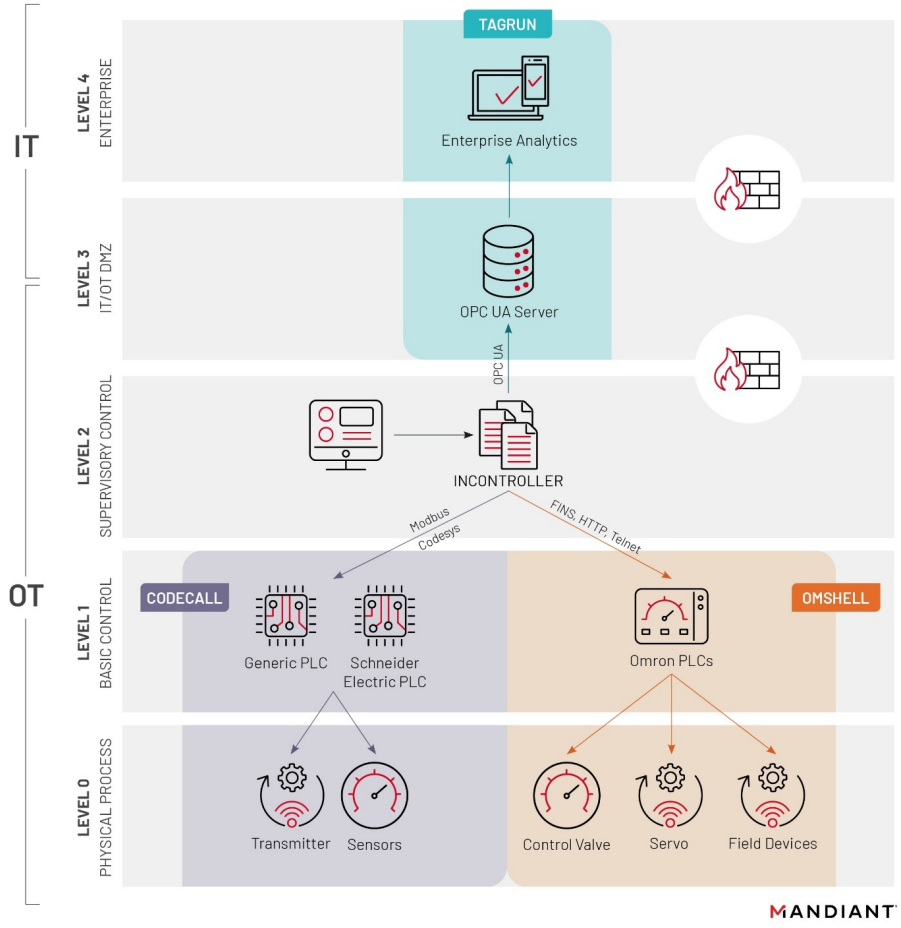

Figure 1: INCONTROLLER tooling overview

INCONTROLLER was developed to allow the attacker to easily add new features to the malware over time. Additionally, the tools are written using the Python programming language, possibly to take advantage of available open-source projects to develop tool features, such as the OPC UA Python library. Mandiant has identified three main INCONTROLLER tools: TAGRUN, CODECALL, and OMSHELL.

TAGRUN

TAGRUN's capabilities, such as the ability to scan for and enumerate OPC UA servers, suggests a reconnaissance role. OPC acts as a central communications protocol to collect and store data from ICS assets in industrial environments. Access to this data can provide attackers with a detailed overview of production systems and control processes. The tool was likely developed for reconnaissance, but it can also write and change tag values, which could be used to modify data to either support an attack or mask process changes. TAGRUN also verifies whether the target environment is running a Windows operating system and provides different ping commands depending on this check's return value. This suggests

that the actor may use non-Windows devices to execute TAGRUN.

CODECALL

CODECALL communicates with ICS devices using the Modbus protocol, which potentially gives it the ability to interact with devices from different manufacturers. However, the tool contains a specific module to interact with, scan, and attack Schneider Electric's Modicon M251 (TM251MESE) PLC using Codesys, which is used by the company's proprietary EcoStruxure Machine Expert protocol. We have reason to believe the tool also targets Schneider Electric's Modicon M221 Nano PLC and the Modicon M258 PLC, and it potentially affects additional devices leveraging the protocol.

OMSHELL

OMSHELL is designed to obtain shell access to Omron PLCs, including Omron NX1P2, NJ501, R88D-1SN10F-ECT servo drive, and possibly other similar devices from the NJ/NX product lines. The framework is modular, which means the attacker can develop and deploy additional capabilities into the tool.

- The tool includes a capability to enumerate and extract data from targeted devices or to restore back-up configurations to prevent response efforts and identification of nefarious activity by defenders. We note that the backups generated by the malware are XOR encrypted, which indicates that for some reason the actor may not want them to be found.
- The tool also includes x86/ARM backdoors that listen on specified ports and provide arbitrary command execution capabilities.

Each of these tools was designed to be executed and operated via the command line without any persistence mechanism and requires attacker interaction. This implies that the tools are not intended to trigger automated changes (outside of CODECALL's ability to load and use macros), but instead require an individual with remote access to the victim environment(s) to perform real-time attacks. This approach provides the actor with flexibility to control changes and adapt the attack to potential responses from a defender.

We have reason to believe that indicator-based detections would not be effective at detecting INCONTROLLER in victim environments, in part because the attacker would almost certainly modify or customize the tool prior to using it in a specific victim environment. Instead, defenders should focus their efforts on behavior-based hunting and detection methods for these tools.

Potential Supporting Windows Tooling

We are also tracking two additional tools affecting Windows-based systems that may be related to this threat activity. It is possible that these tools could be used to support the overall attack lifecycle in an INCONTROLLER attack by exploiting Windows-based systems in IT or operational technology (OT) environments. To expand detections for INCONTROLLER activity, we include YARA rules to identify these tools in Appendix E.

- One of the tools exploits CVE-2020-15368 in the AsrDrv103.sys driver, which would result in installation and exploitation of a vulnerable driver. ASRock motherboards may be leveraged in some human-machine interfaces (HMIs) and engineering workstations in OT environments.
- The other tool, which we track as ICECORE, is a backdoor written in C++ that performs command and control (C&C) over SSL. Its capabilities include surveying system information using WMI, arbitrary command execution, read/write file operations, directory enumeration, and read/write registry entries.

Attack Scenarios

As our observations are based on intrusion activity stages outside specific breaches, the characteristics of potential victims of INCONTROLLER activity are currently unclear. It is both feasible that each tool is meant for use across different intrusions, or the actor may use the three tools to attack a single environment. We highlight that the devices targeted by INCONTROLLER are often integrated in automation machinery (e.g., a milling machine or press) and could plausibly be present in a variety of industrial sectors and processes even without the user's explicit knowledge.

We developed three cyber physical attack scenarios that highlight a range of possible outcomes from an attack using INCONTROLLER. In each of the three cases, TAGRUN could have been used at earlier stages to enumerate the victim environment, identify its targets, and learn about the physical process.

Figure 2: INCONTROLLER attack scenarios

The impact of these scenarios would depend on the nature of the victim facility and the extent of the attacker's understanding of and interaction with the controlled physical process. We note that our current understanding of INCONTROLLER is still limited given that we believe we identified the tools at an early stage of the attack lifecycle, and that the code was developed with an extensible structure that can support new features implemented by the author.

INCONTROLLER Is Very Likely State-Sponsored Malware

We believe INCONTROLLER is very likely linked to a state-sponsored group given the complexity of the malware, the expertise and resources that would be required to build it, and its limited utility in financially motivated operations. We are unable to associate INCONTROLLER with any previously tracked group at this stage of our analysis, but we note the activity is consistent with Russia's historical interest in ICS (21-00023804). While our evidence connecting INCONTROLLER to Russia is largely circumstantial, we note it given Russia's history of destructive cyber attacks, its current invasion of Ukraine, and related threats against Europe and North America.

- We identified limited initial indications, including a grammatical anomaly, which suggest the malware was developed by Russian speakers.
- Since at least 2014, Russia-nexus threat actors have targeted ICS assets and data with multiple ICS-tailored malware families (PEACEPIPE, BlackEnergy2, INDUSTROYER, TRITON, and VPNFILTER).

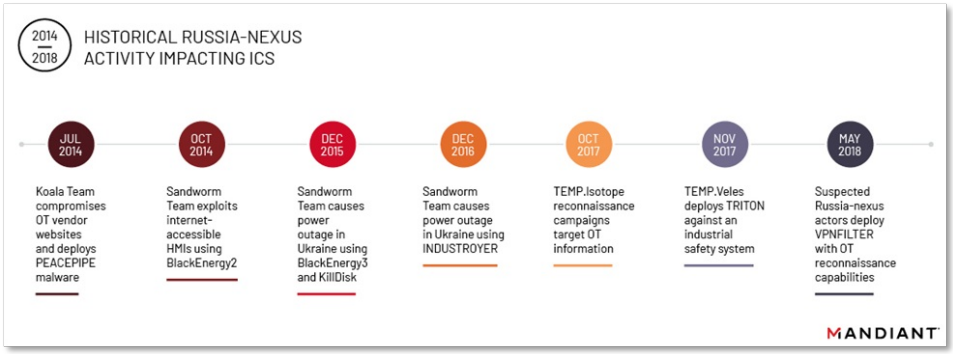

Figure 3: Historical Russia-nexus activity impacting ICS

- INCONTROLLER's functionality is consistent with the malware used in Russia's prior cyber physical attacks. For example, the 2015 and 2016 Ukrainian blackouts both involved physical process manipulations combined with disruptive attacks against embedded devices (17-00006337 and 16-00001698). INCONTROLLER similarly allows the malware operator to manipulate physical processes, while also containing denial-of-service (DoS) capabilities to disrupt the availability of PLCs.

Recommendations

While the nature of any potential intended targets remains uncertain, INCONTROLLER poses a critical risk to organizations with compatible devices. The targeted devices are embedded in multiple types of machinery and could plausibly be present in many different industrial sectors. Given the consistencies with prior Russia-nexus threat activity, we suggest that INCONTROLLER poses the greatest threat to Ukraine, NATO member states, and other states actively responding to Russia's invasion of Ukraine. Organizations should take immediate action to determine if the targeted ICS devices are present in their environments and begin applying vendor-specific countermeasures.

We also recommend that at-risk organizations conduct threat hunts to detect this activity in their networks. For this we provide threat hunting guidance to identify activity associated with INCONTROLLER and YARA detections for the supporting Windows tools.

Mitigations

OPC UA

We recommend several steps to mitigate risk and counter malicious activity in environments using this protocol:

- Proper segmentation of IT and OT networks to aid in preventing attackers pivoting from corporate networks into industrial environments.
- Allow listing accepted primary/subordinate devices, behavior patterns, and commands to aid in establishing approved baselines and detecting anomalies with the aid of network monitoring.
- Implementation of an industrial firewall with deep packet inspection to aid in controlling access and approved capabilities.
- Implementation of ICS-aware intrusion protection systems to aid in monitoring for function codes from potentially malicious sources.
- Monitoring and blocking of external traffic to OPC UA ports, when possible, to aid in detecting anomalous traffic and prevent external network traffic directed at OPC UA-associated ports. Associated traffic and behaviors that could indicate risk include:
  - Use of non-OPC US communication over ports 4840 and 443.
  - Traffic from non-recognized devices.
- Enabling and aggregating audit logs for OPC servers and clients.
- Periodic reviewing of audit logs for inconsistent or nefarious connections, security options negotiations, configuration changes, and user interaction.

Schneider Electric

To help keep your Schneider Electric products secure and protected, it is in your best interest that you implement the cyber security best practices as indicated in the Cybersecurity Best Practices document provided on the Schneider Electric website: Recommended Cybersecurity Best Practices White paper | Schneider Electric.

Additionally, ***Cybersecurity Guidelines for EcoStruxure Machine Expert, Modicon and PacDrive Controllers and Associated Equipment User Guide*** could help you ensure that only legitimate users can access your Schneider Electric product: Cybersecurity Guidelines for EcoStruxure Machine Expert, Modicon and PacDrive Controllers and Associated Equipment, User Guide | Schneider Electric.

You should pay special attention to features and cyber security devices that help to restrict access to authorized users only. This includes examples such as intrusion detection systems, network firewalls, secure remote access, device authentication, device firewall, disabling/filtering unsecure or programming protocols.

Omron

According to public vulnerability notices, Omron has previously identified other vulnerabilities that use the same or similar FIN ports that are used by OMSHELL. Omron's guidance for unpatched vulnerabilities, as noted in their security brief, indicates that external firewall filtering of identified FIN ports can be used as a mitigation. Mandiant believes that the recommended methodology may be a viable mitigation, though this mechanism has not been tested with INCONTROLLER. Additional guidance related to Omron's previous recommendations can be found in the related ICS Advisory for that older vulnerability.

Discovery Methods

- Identify and investigate the creation, transfer, and/or execution of unauthorized Python scripts or Python-compiled executables (e.g., PyInstaller or Py2Exe) on OT systems.

TAGRUN

- Search for and investigate irregular connections to OPC UA endpoints and enable robust audit logging for OPC UA applications. Aggregate OPC UA logs and audit records to a central location where applicable.
- Review OPC UA audit records for evidence of credential bruteforcing, nefarious certificate usage, irregular connection attempts, configuration changes, and changes to OPC tags.
- Search for and investigate TAGRUN ping command execution:
  - Windows OS: ping -n 1 -w 2 <IP>
  - Non-Windows OS: ping -c1 -w2 <IP>
- Review OT network traffic for evidence of pingsweep activity.

CODECALL

- Search for and investigate evidence of the creation/existence of CODECALL-related host-based indicators on systems with access to OT resources:
  - <user_specified_name>.mac (macros created by the framework use the ".mac" extension)

- Enable robust logging for Schneider Electric PLC devices and aggregate logs to a central location where applicable.
- Review Schneider Electric device logs for evidence of the following activity:
  - Credential bruteforcing
  - Authentication attempts using the hard-coded "m258" user name
  - Error codes associated with abnormal device crashes/reboots
  - Files uploaded or downloaded
  - File deletion
  - Unauthorized changes in device configuration and execution of commands
  - Connections to devices outside of documented norms for the device and environment
- Search for and investigate evidence of ARP scanning followed by abnormal Modbus/Codesys traffic differing from environment baselines.
- Search for abnormal Modbus and Codesys traffic flows compared to environment baselines.

OMSHELL

- Search for and investigate evidence of the creation/existence of OMSHELL-related host-based indicators on systems with access to OT resources and connectivity (e.g., packet captures).
- Enable robust logging for Omron PLC devices and aggregate logs to a central location where applicable.
- Review Omron device logs for evidence of the following activity:
  - Activation of Telnet daemon on the device.
  - Unauthorized Telnet connection attempts including the use of default credentials, such as "_root/omron."
  - Wiping of PROGRAM memory and device resets.
  - Unauthorized changes in device configuration and execution of commands.
  - Connections to devices outside of documented norms for the device and environment.
  - Files uploaded or downloaded.
- Identify and investigate nefarious pingsweep scanning activity, telnet traffic, and HTTP traffic on systems with access and connectivity to OT resources/devices:
  - udp://<target_ip>:9600
  - http://<target_ip>:80
  - tcp://<target_ip>:23
  - tcp://<target_ip>:7777
- Search for and investigate evidence of Omron FINS traffic outside of standard norms and environment baselines.
- Collect, identify, and investigate nefarious HTTP POST data to Omron devices containing Omron API commands.

Appendix A: MITRE ATT&CK for ICS Mapping

Module	Tactic	Technique
TAGRUN	Execution	T0807: Command-Line Interface
TAGRUN	Execution	T0853: Scripting
TAGRUN	Lateral Movement	T0859: Valid Accounts
TAGRUN	Discovery	T0888: Remote System Information Discovery
TAGRUN	Persistence	T0859: Valid Accounts
TAGRUN	Collection	T0801: Monitor Process State
TAGRUN	Collection	T0861: Point & Tag Identification
TAGRUN	Command and Control	T0885: Commonly Used Port
TAGRUN	Command and Control	T0869: Standard Application Layer Protocol
TAGRUN	Impact	T0832: Manipulation of View
TAGRUN	Impact	T0882: Theft of Operational Information
TAGRUN	Discovery	T0846: Remote System Discovery

Table 2: MITRE ATT&CK for ICS mapping for TAGRUN

Module	Tactic	Technique
CODECALL	Execution	T0807: Command-Line Interface
CODECALL	Execution	T0853: Scripting
CODECALL	Persistence	T0859: Valid Accounts
CODECALL	Persistence	T0857: System Firmware
CODECALL	Persistence	T0889: Modify Program
CODECALL	Discovery	T0846: Remote System Discovery
CODECALL	Discovery	T0888: Remote System Information Discovery
CODECALL	Lateral Movement	T0812: Default Credentials
CODECALL	Lateral Movement	T0843: Program Download
CODECALL	Lateral Movement	T0859: Valid Accounts
CODECALL	Collection	T0801: Monitor Process State
CODECALL	Collection	T0845: Program Upload
CODECALL	Collection	T0801: Monitor Process State
CODECALL	Command and Control	T0885: Commonly Used Port
CODECALL	Command and	T0869: Standard Application Layer

	Control	Protocol
CODECALL	Inhibit Response Function	T0804: Block Reporting Message
CODECALL	Inhibit Response Function	T0803: Block Command Message
CODECALL	Inhibit Response Function	T0814: Denial of Service
CODECALL	Inhibit Response Function	T0809: Data Destruction
CODECALL	Inhibit Response Function	T0816: Device Restart/Shutdown
CODECALL	Inhibit Response Function	T0857: System Firmware
CODECALL	Impair Process Control	T0836: Modify Parameter
CODECALL	Impair Process Control	T0855: Unauthorized Command Message
CODECALL	Impact	T0813: Denial of Control
CODECALL	Impact	T0815: Denial of View
CODECALL	Impact	T0826: Loss of Availability
CODECALL	Impact	T0827: Loss of Control
CODECALL	Impact	T0828: Loss of Productivity and Revenue
CODECALL	Impact	T0831: Manipulation of Control
CODECALL	Impact	T0882: Theft of Operational Information

Table 3: MITRE ATT&CK for ICS mapping for CODECALL

Module	Tactic	Technique
OMSHELL	Initial Access	T0886: Remote Services
OMSHELL	Execution	T0807: Command-Line Interface
OMSHELL	Execution	T0853: Scripting
OMSHELL	Execution	T0858: Change Operating Mode
OMSHELL	Execution	T0821: Modify Controller Tasking
OMSHELL	Execution	T0834: Native API
OMSHELL	Persistence	T0889: Modify Program
OMSHELL	Persistence	T0859: Valid Accounts
OMSHELL	Evasion	T0858: Change Operating Mode
OMSHELL	Discovery	T0842: Network Sniffing
OMSHELL	Discovery	T0846: Remote System Discovery
OMSHELL	Discovery	T0888: Remote System Information Discovery
OMSHELL	Lateral Movement	T0812: Default Credentials
OMSHELL	Lateral Movement	T0867: Lateral Tool Transfer
OMSHELL	Lateral Movement	T0843: Program Download
OMSHELL	Lateral Movement	T0886: Remote Services
OMSHELL	Lateral Movement	T0859: Valid Accounts
OMSHELL	Collection	T0868: Detect Operating Mode
OMSHELL	Collection	T0801: Monitor Process State
OMSHELL	Collection	T0845: Program Upload
OMSHELL	Command and Control	T0885: Commonly Used Port
OMSHELL	Command and Control	T0869: Standard Application Layer Protocol
OMSHELL	Inhibit Response Function	T0881: Service Stop
OMSHELL	Impair Process Control	T0836: Modify Parameter
OMSHELL	Impair Process Control	T0855: Unauthorized Command Message
OMSHELL	Impact	T0879: Damage to Property
OMSHELL	Impact	T0837: Loss of Safety
OMSHELL	Impact	T0831: Manipulation of Control
OMSHELL	Impact	T0882: Theft of Operational Information

Table 4: MITRE ATT&CK for ICS mapping for OMSHELL

Appendix B: TAGRUN Technical Analysis

TAGRUN is a scanner, reader, and writer tool for OPC UA servers. The tool is written in Python and takes in a list of IP addresses and ports as arguments and scans them for the presence of OPC UA endpoints. TAGRUN also supports login via credentials or authentication certificates to be able to retrieve OPC endpoints from an online server and fetch a server's structure to a specified depth. TAGRUN can also read/write values to nodes within a server. Additionally,

TAGRUN has the capability of brute forcing credentials to get access to an OPC UA server.

The primary scanning functionality of the tool is performed by first pinging the specified range of IPs/ports to check if the hosts are online.

TAGRUN then uses the OPC UA Python library to attempt to connect to online hosts and retrieve available endpoints. The same library is used to attempt to scan the server structure and read/write to specific values within the structure. The depth of server scanning and an endpoint URI can be optionally specified.

The number of maximum threads for brute forcing OPC UA server credentials can be set as well. After finding alive hosts from the ping/ICMP messages, the OPC UA connection request and response look as follows over the network:

```
48 45 4c 46 3e 00 00 00  00 00 00 00 ff ff ff 7f   HELF>... ........
ff ff ff 7f 00 00 00 00  00 00 00 00 1e 00 00 00   ........ ........
6f 70 63 2e 74 63 70 3a  2f 2f 31 37 32 2e 31 36   opc.tcp: //172.16
2e 32 31 38 2e 32 31 31  3a 36 32 36 34 30         .218.211 :62640

41 43 4b 46 1c 00 00 00  00 00 00 00 ff ff 00 00   ACKF.... ........
ff ff 00 00 00 00 40 00  00 00 00 00               ......@. ....

4f 50 4e 46 84 00 00 00  00 00 00 00 2f 00 00 00   OPNF.... ..../...
68 74 74 70 3a 2f 2f 6f  70 63 66 6f 75 6e 64 61   http://o pcfounda
74 69 6f 6e 2e 6f 72 67  2f 55 41 2f 53 65 63 75   tion.org /UA/Secu
72 69 74 79 50 6f 6c 69  63 79 23 4e 6f 6e 65 ff   rityPoli cy#None.
ff ff ff ff ff ff ff 01  00 00 00 01 00 00 00 01   ........ ........
00 be 01 00 00 de d6 75  f9 10 23 d8 01 01 00 00   .......u ..#.....
00 00 00 00 00 ff ff ff  ff e8 03 00 00 00 00 00   ........ ........
00 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00   ........ ........
80 ee 36 00                                        ..6.

4f 50 4e 46 87 00 00 00  04 00 00 00 2f 00 00 00   OPNF.... ..../...
68 74 74 70 3a 2f 2f 6f  70 63 66 6f 75 6e 64 61   http://o pcfounda
74 69 6f 6e 2e 6f 72 67  2f 55 41 2f 53 65 63 75   tion.org /UA/Secu
72 69 74 79 50 6f 6c 69  63 79 23 4e 6f 6e 65 ff   rityPoli cy#None.
ff ff ff ff ff ff ff 01  00 00 00 01 00 00 00 01   ........ ........
00 c1 01 dd b0 b6 f9 10  23 d8 01 01 00 00 00 00   ........ #.......
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 04   ........ ........
00 00 00 01 00 00 00 dd  b0 b6 f9 10 23 d8 01 80   ........ ....#...
ee 36 00 ff ff ff ff                               .6.....

4d 53 47 46 63 00 00 00  04 00 00 00 01 00 00 00   MSGFc... ........
02 00 00 00 02 00 00 00  01 00 ac 01 00 00 fe 24   ........ .......$
76 f9 10 23 d8 01 02 00  00 00 00 00 00 00 ff ff   v..#.... ........
ff ff e8 03 00 00 00 00  00 1e 00 00 00 6f 70 63   ........ .....opc
2e 74 63 70 3a 2f 2f 31  37 32 2e 31 36 2e 32 31   .tcp://1 72.16.21
38 2e 32 31 31 3a 36 32  36 34 30 00 00 00 00 00   8.211:62 640.....
00 00 00
```

Figure 4: TAGRUN network communications

The traffic follows the OPC UA specification (IEC 62541) using headers from the python-opcua code from GitHub. These connection requests are generated via the "connect_and_get_server_endpoints" API of the OPCUA library and do not contain any unique network indicators as such, except for the potentially repeated nature of the scan itself.

Appendix C: CODECALL Technical Analysis

CODECALL is a Python-based framework capable of interacting with devices that support the Modbus protocol. The framework is extensible with attack modules for specific devices and with any PLC devices that understand Modbus. Additionally, it contains scanning functionality to look for Modbus-enabled devices in a range of IPs. CODECALL capabilities include the ability to connect to devices, load a set of commands that can be executed in one go and dumped/loaded to/from an external file, upload files to devices, execute device specific commands provided by additional modules, connect to devices using the Modbus protocol, and read/write to device registers over Modbus.

The framework is extensible with attack modules for specific devices. The framework also supports Modbus protocol and can communicate with any PLC devices that understand Modbus. Additionally, it also contains a scan function to look for Modbus enabled devices in a range of IPs. The framework has support for "macros," which are essentially a set of commands that can be executed in one go and dumped/loaded to/from an external file.

The framework leverages a helper library for Schneider devices. This library is used within the framework at multiple locations for purposes such as to get a list of available Schneider devices. Additionally, it contains code that implements the network communications of the framework including the "Machine Expert" protocol and the Modbus protocol.

TM251 Module

This tool includes functionality for PLC device attack for interacting with, scanning, and attacking Schneider Electric's PLC devices, likely to include at least TM251 models. The following is the list of capabilities that it supports.

- Download/upload to the PLC device.
- Get file/directory listing.
- Delete file.
- Login/logout of the device after a connection has been made.
- Brute-force credentials, provided a dictionary file.
- Attempt a DDoS attack.
- Disconnect from the PLC device.
- Crash the device.
- Adds a route if the device gateway IP exists on a different interface.

- Send custom raw packet.
- Check the current login status.
- Attempts to login with hard-coded user name "m258" and a specified password hash.

Macros

CODECALL can use "macros" files as a method to load and execute commands.

Network Traffic

The framework operates over two protocols:
- Modbus over TCP on port 502
- "Machine Expert" protocol over UDP ports 1740, 1741, 1742 and 1743
    - According to Schneider's documentation, TCP port 11740 is also supported

The following is an example communication over Modbus. This includes the initial connection and register read/write. This traffic was generated using the framework with the Modbus simulation server ModRSsim2.

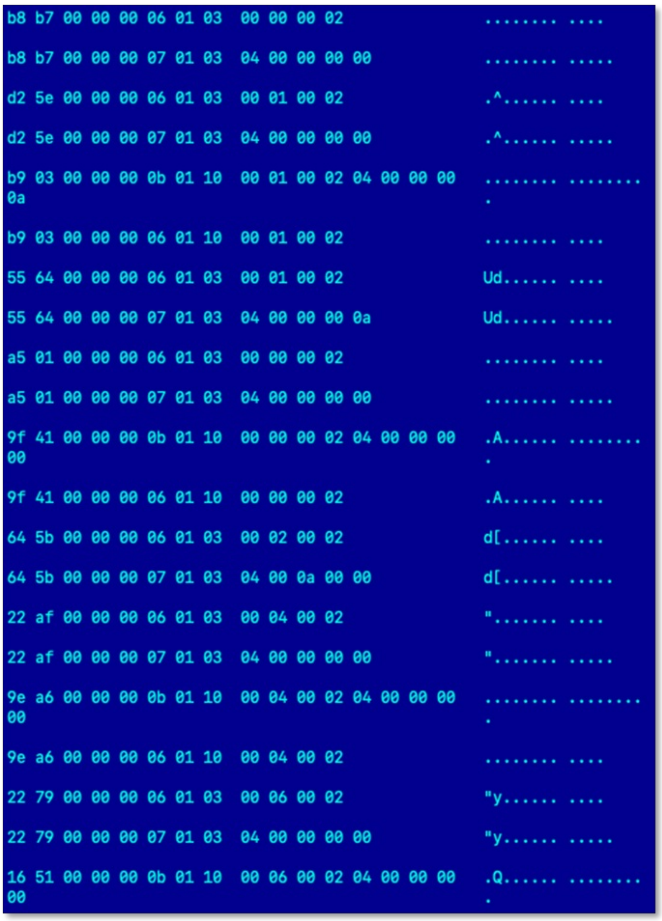

Figure 5: Modbus network traffic

The traffic itself is generic Modbus traffic and does not contain any unique indicators. Modbus scan PCAP contains ARP packets to the IP range being scanned, with finally a Modbus packet when a Modbus enabled device is hit. The framework by default attempts device identification upon connection to a Modbus device, however the Modbus simulator used did not support that command and thus a successful response to the identification request is not observed.

There are no "Machine Expert" protocol simulators available. Schneider's PLC IDE (EcoStruxure Machine Expert Software) allows programming multiple PLC devices including TM251MESE and simulating them to interact with each other. However, from a brief inspection, it seems that it does not expose those simulated devices to the outside network to be interacted with, with the likes of this framework. Thus, no PLC interaction over "Machine Expert" interaction traffic was generated. However, following are the initial packets that are sent out to the PLC device over UDP when a connection is attempted by the framework.

Figure 6: Machine expert network traffic

Appendix D: OMSHELL Technical Analysis

OMSHELL is a Python-based framework that is targeted towards Omron's PLC devices. OMSHELL has the capability to scan the network to identify Omron devices as well as any other open ports on an online device, connect to the device,

get/set configuration information, upload/download files, backup/restore device's configuration and data, activate and connect to the telnet module of the device, deploy a backdoor that comes bundled with the framework and terminate arbitrary processes. OMSHELL primarily operates using the HTTP protocol, however, it does utilize Omron's proprietary FINS over UDP protocol during the scanning and device identification phase. Additionally, OMSHELL appears to be capable of communicating with Omron's Servo drives as well.

Network Indicators

Connections
udp://<target_ip>:9600
http://<target_ip>:80
tcp://<target_ip>:23
tcp://<target_ip>:7777 (when default settings are used with the ExploitServerModule)

Table 5: Network connections

Method	URI	Command	Module	Description
POST	/cgi-bin/cpu.fcgi	Backup_getCommandFile Backup_beginBackup Backup_beginRestore Backup_createBackupFile Backup_precheckFlushRestore Backup_endRestore/Backup_endBackup Backup_getBackupArchive Backup_getBackupStatus Backup_getListOfFiles UserProgram_tool Backup_getRestoreResult Backup_notifyBackupArchiveUploadCompletion Succeeded Backup_prepareRestore Backup_putBackupArchive Backup_setBackupParameter Backup_setBackupParameter	Omron Client	Device backup related commands.
POST	/cgi-bin/cpu.fcgi	Session_resetExpires	Omron Client	Is called right after Backup_beginBackup.
POST	/cgi-bin/cpu.fcgi	CPU_getMode CPU_getModel CPU_getPlcName CPU_loadTcpIpSetting CPU_reset CPU_setMode mode=PROGRAM/CPU_setMode mode=RUN CPU_setPlcName CPU_ClearAllMemory	Omron Client	CPU/config/mode related commands. Command names are descriptive.
POST	/cgi-bin/cpu.fcgi	File_beginDownload File_download File_endDownload FileList_get FileList_put File_precheckFlushDownload File_prepareDownload File_upload	Omron Client	File transfer related commands. Command names are descriptive.
POST	/cgi-bin/ecat.fcgi	Sync_lock /Sync_unlock	Omron Client	Synchronization related commands.
POST	/cgi-bin/ecat.fcgi	CoESDO_read	Omron Client	Reads a value from a CoE* object of a specified subordinate on an EtherCAT network. (ref.).
POST	/cgi-bin/ecat.fcgi	CoESDO_writeBin	Omron Client	Writes a value to a CoE* object of a specified subordinate on an EtherCAT network. (ref.) In the framework's context, CoESDO_read/CoESDO_writeBin appear to be used to communicate with Servo drives.
POST	/cgi-bin/cpu.fcgi	File_flushDownload	File List	Flushing downloaded items to device memory.
POST	/cgi-bin/cpu.fcgi	Console mount -o remount,rw /dev/hda0t79	Serve rExploit	Remounts device in read/write mode.
POST	/cgi-bin/cpu.fcgi	Console telnetd -debug 23	Telnet	Executes Telnet daemon on the

			Activator	device.
GET	/utif/UserInfo.xml	//	Omron Client	Fetches user information from device.
GET	/utif/CpuUnit.xml	//	Omron Scanner	Fetches CPU information from device.
GET	/Systems.xml	//	Omron Scanner	System configuration information.

Table 6: Network indicators

The framework contains code for shell wrappers of framework modules, implementation of framework modules, a telnetlib wrapper, file read/write helper functions, compiled x86 and ARM versions of the "Server" backdoor binary, tcpdump and dd utility, MAC to company and port to description mappings, and File IO over HTTP related implementation that ***might*** be standard to Omron's Sysmac Automation Platform and EtherCAT.

The primary protocol used by the framework is HTTP. The framework does use Omron's proprietary FINS over UDP protocol, but only during the device identification process. There is very little documentation available that hints or details the HTTP protocol/API usage for Omron PLCs. The only Omron PLC devices that seem to support HTTP for certain belong to the NX/NJ series. The documentation for this series refers to port 80 being used as an HTTP server on page 39. However, no specifics of the commands available over HTTP are provided. The code contains references to the following models:

- NX1P2 (https://www.ia.omron.com/products/family/3650/)
- NJ501 (https://www.ia.omron.com/products/family/3075/)
- R88D-1SN10F-ECT (https://industrial.omron.eu/en/products/R88D-1SN10F-ECT)

Where before deploying the "tcpdump" binary to the device, it checks for the device model and selects the architecture. The implementation also suggests that the processor in nx1p2 is x86 based, while that of nj501 is ARM based. Following is a list of all the commands being used, most of which are transmitted as POST data.

```
POST /cgi-bin/cpu.fcgi HTTP/1.1
Host: 172.16.218.203
User-Agent: python-requests/2.25.1
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Length: 12

CPU_getModel
```

Figure 7: OMSHELL HTTP command execution traffic

Identification

This component first does an ICMP sweep on the specified range of IP addresses to find online hosts, then it scans them on the specified ports to identify running services and to find Omron PLC devices. Open ports are mapped to service descriptions using "nmap-mac-prefixes." Following three techniques are used to identify these devices:

1. Checks the MAC address of the device. After mapping the MAC address to the company, the framework checks if the company name contains "omron." Following Omron related MACs are present in the file.

MAC Addresses
B0495F Omron Healthcare
00000A Omron Tateisi Electronics
002209 Omron Healthcare

Table 7: MAC address prefixes

2. Attempts to connect to port 80, and if a connection is made issues the GET request "http://<ip>/Systems.xml" to check whether the device is an Omron PLC.
3. Attempts to identify the device using Omron's proprietary FINS over UDP protocol over port 9600.

Set

Sets and overwrites the default IP, user name, and password

Clear Memory

Wipes the PROGRAM memory, optionally resets the PLC, and sets the mode to RUN again.

Backup

Loads backup configuration and can backup data from or restore data to a PLC. The path to backup/restore locally is supplied by the user.
The backups are XOR encrypted

When a backup is fetched and unpacked, the decrypted files are stored with the extension ".dec."

Telnet Activation

The telnet activation module starts the telnet service on the PLC.

Exploit Telnet

This module activates the telnet daemon and then connects to the device's telnet port and uploads and optionally executes an arbitrary payload that has been specified.

Exploit Server

This module is somewhat like the Telnet module, except that instead of to the telnet server, it expects to connect to the "Server" present within the framework. These x86/ARM native "Server" binaries are essentially simple backdoors that listen on specified ports and provide arbitrary command execution capabilities. The default port for connection to the server is 7777. This exploit module also provides the ability to upload a tcpdump binary to the device and start a network capture which is dumped to a user specified path. The capability also provides an ability to kill arbitrary processes.

Transfer

This module implements file transfer capabilities directly over HTTP. The data transferred is XOR encoded and gzip compressed.

Servo

This module allows communicating with Servo drives using the following commands: "CoESDO_read" and "CoESDO_writeBin." The data relayed with these commands is serialized/deserialized using protobuf.

Appendix E: YARA Signatures

Note: The YARA rules included are meant for hunting purposes and have not been tested for use in production environments.

```
rule MTI_Hunting_AsRockDriver_Exploit_PDB
{
	meta:
		author = "Mandiant"
		date = "03-23-2022"
		description = "Searching for executables containing strings associated with AsRock driver Exploit."

	strings:
		$dos_stub = "This program cannot be run in DOS mode"
		$pdb_bad = "dev projects\\SignSploit1\\x64\\Release\\AsrDrv_exploit.pdb"
		$pdb_good =
"c:\\asrock\\work\\asrocksdk_v0.0.69\\asrrw\\src\\driver\\src\\objfre_win7_amd64\\amd64\\AsrDrv103.pdb"

	condition:
		all of them and (@pdb_bad < @dos_stub[2]) and (#dos_stub == 2) and (@pdb_good > @dos_stub[2])
}

rule MTI_Hunting_AsRockDriver_Exploit_Generic
{
	meta:
		author = "Mandiant"
		date = "03-23-2022"
		description = "Searching for executables containing strings associated with AsRock driver Exploit."

	strings:
		$dos_stub = "This program cannot be run in DOS mode"
		$pdb_good =
"c:\\asrock\\work\\asrocksdk_v0.0.69\\asrrw\\src\\driver\\src\\objfre_win7_amd64\\amd64\\AsrDrv103.pdb"

	condition:
		all of them and (#dos_stub == 2) and (@pdb_good > @dos_stub[2])
}

rule MTI_Hunting_ICECORE_integrityinitkey
{
	meta:
		author = "Mandiant"
		date = "03-23-2022"
		description = "Searching for executables containing strings associated with the ICECORE malware family."

	strings:
		$hmac_init_key = { C7 [2-6] C8 67 FA E8 C7 [2-6] 4C 7E 80 79 C7 [2-6] 7B 79 B6 A0 C7 [2-6] 5E 74 45 52
}
	condition:
```

```
		uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550 and all of them
}

rule MTI_Hunting_ICECORE_websocketmsgmask
{
	meta:
		author = "Mandiant"
		date = "03-23-2022"
		description = "Searching for executables containing strings associated with the ICECORE malware family."

	strings:
		$websocket_buildtextmsg_mask = { C7 [2-6] 01 02 03 04 }
		$websocket_buildtextmsg_recalc = { 8B 07 8B CA 83 E1 03 03 C2 42 [2-6] 30 4C 30 04 3B 13 }

	condition:
		uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550 and all of them
}

rule MTI_Hunting_ICECORE_configstatickey
{
	meta:
		author = "Mandiant"
		date = "03-23-2022"
		description = "Searching for executables containing strings associated with the ICECORE malware family."

	strings:
		$static_iv1 = { C7 [2-6] A2 F2 F8 28 }
		$static_iv2 = { C7 [2-6] BD E9 5A 44 }
		$static_iv3 = { C7 [2-6] FD A8 7B 62 }
		$static_iv4 = { C7 [2-6] 78 B4 37 9D }
		$static_key1 = { C7 [2-6] A6 C3 08 EB }
		$static_key2 = { C7 [2-6] 20 C5 89 CA }
		$static_key3 = { C7 [2-6] 93 03 24 11 }
		$static_key4 = { C7 [2-6] 50 C9 B2 AF }
		$static_key5 = { C7 [2-6] 0E C7 4E B3 }
		$static_key6 = { C7 [2-6] 97 73 4F 33 }
		$static_key7 = { C7 [2-6] 32 6C 92 6A }
		$static_key8 = { C7 [2-6] 0B 88 62 9A }

	condition:
		uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550 and all of them
}

rule MTI_Hunting_ICECORE_Strings
{
	meta:
		author = "Mandiant"
		date = "03-23-2022"
		description = "Searching for files containing strings associated with the ICECORE malware family."

	strings:
		$s1 = "HWKey::load"
		$s2 = "HWKey::create_com"
		$s3 = "CoCreateInstance CLSID_WbemLocator %08X"
		$s4 = "HWKey::read_macaddr"
		$s5 = "ExecQuery SELECT * FROM Win32_NetworkAdapter %08X"
		$s6 = "HWKey::read_baseboard"
		$s7 = "ExecQuery SELECT * FROM Win32_BaseBoard %08X"
		$s8 = "MD4IntegrityChecker::run"
		$s9 = "MD5IntegrityChecker::run"
		$s10 = "HMACIntegrityChecker::run"
		$s11 = "SHA1IntegrityChecker::run"
		$s12 = "SHA256IntegrityChecker::run"
		$s13 = "SHA224IntegrityChecker::run"
		$s14 = "SHA512IntegrityChecker::run"
		$s15 = "IntegrityControl::IntegrityControl"
		$s16 = "SHA384IntegrityChecker::run"
		$s17 = "IntegrityControl::watch_module"
		$s18 = "PEEncrypt::create_gamma"
		$s19 = "EVP_aes_256_cbc fail"
		$s20 = "PEEncrypt::module_hash"
		$s21 = "ModuleOwner::handle_entry_exception"
		$s22 = "ModuleOwner::load_new_module"
		$s23 = "module %s config readed"
		$s24 = "inconsistent module cfg '%s', error %s"
		$s25 = "module(%s) file not found - %S"
```

```
        $s26 = "ModuleOwner::begin_module_load"
        $s27 = "load required library fail (%s)"
        $s28 = "module hash is invalid, but we load this module (%s)"
        $s29 = "load library fail (%s)"
        $s30 = "ModuleOwner::run_module_loadlibarry"
        $s31 = "module can't be loaded (%s)"
        $s32 = "ModuleOwner::final_module_initialize"
        $s33 = "start load library (%s)"
        $s34 = "ModuleOwner::locked_load_library"
        $s35 = "ModuleOwner::add_new_module"
        $s36 = "ModuleOwner::module_message"
        $s37 = "ModuleOwner::unload_module"
        $s38 = "ModuleOwner::module_fully_initialized"
        $s39 = "NotificationMonitor::new_message"
        $s40 = "hardware info: %s"
        $s41 = "WMIEnum::next"
        $s42 = "system info: %s"
        $s43 = "system_info_kb"
        $s44 = "SELECT * FROM Win32_ComputerSystem returns more than 1 object"
        $s45 = "system_info_hardware"
        $s46 = "installed kb: %s"
        $s47 = "SELECT * FROM Win32_Processor returns more than 1 object"
        $s48 = "vm info: %s"
        $s49 = "SELECT * FROM Win32_BIOS returns more than 1 object"
        $s50 = "SystemScout::get_system_info_base"
        $s51 = "ServerIO::establish_connection"
        $s52 = "ConfigDB::hwkey_ready"
        $s59 = "ServerIO::check_proxy_bypass"
        $s60 = "ServerIO::retry_connection"
        $s61 = "ServerIO::read_proxy_status"
        $s62 = "ServerIO::handle_proxy"
        $s63 = "ServerIO::read_proxy_addr"
        $s64 = "ANTIDEBUG:Interrupt_0x2d"
        $s65 = "ANTIDEBUG:CheckRemoteDebuggerPresentAPI"
        $s66 = "ThreadMonitor::is_debugger_present"
        $s67 = "ANTIDEBUG:NtQueryInformationProcess_ProcessDebugFlags"
        $s68 = "ANTIDEBUG:NtQueryInformationProcess_ProcessDebugObject"
        $s69 = "ANTIDEBUG:Interrupt_3"
        $s70 = "ANTIDEBUG:IsDebuggerPresentAPI"
        $s71 = "ANTIDEBUG:NtSetInformationThread_ThreadHideFromDebugger"
        $s72 = "ANTIDEBUG:VirtualAlloc_WriteWatch_BufferOnly"
        $s73 = "ANTIDEBUG:NtQueryInformationProcess_ProcessDebugPort"
        $s74 = "ANTIDEBUG:NtQuerySystemInformation_SystemKernelDebuggerInformation"
        $s75 = "ANTIDEBUG:VirtualAlloc_WriteWatch_CodeWrite"
        $s76 = "ANTIDEBUG:SetHandleInformatiom_ProtectedHandle"
        $s77 = "ANTIDEBUG:VirtualAlloc_WriteWatch_APICalls"
        $s78 = "ANTIDEBUG:VirtualAlloc_WriteWatch_IsDebuggerPresent"
        $s79 = "ANTIDEBUG:HeapFlags"
        $s80 = "ANTIDEBUG:HeapForceFlags"
        $s81 = "ANTIDEBUG:MemoryBreakpoints_PageGuard"
        $s82 = "ANTIDEBUG:IsDebuggerPresentPEB"
        $s83 = "ANTIDEBUG:NtGlobalFlag"
        $s84 = "ANTIDEBUG:SharedUserData_KernelDebugger"
        $s85 = "WebsocketParser::read_header"
        $s86 = "ConfigDB::write_config_start_write"
        $s87 = "ConfigDB::check_server_public_key"
        $s88 = "ConfigDB::setup_ssl_credentials"
        $s89 = "ConfigDB::read_config_setup_config"
        $s90 = "ConfigDB::save_config"
        $s91 = "ConfigDB::write_config_make_buffer"
        $s92 = "ConfigDB::write_config_encrypt_buffer"

    condition:
        5 of them
}
```

Please rate this product by taking a short four question survey

Threat Intelligence Tags

Affected Industries

- Aerospace & Defense
- Automotive
- Chemicals & Materials

- Construction & Engineering
- Energy & Utilities
- Governments
- High Tech/Software/Hardware/Services
- Manufacturing
- Oil & Gas
- Pharmaceuticals
- Technology
- Telecommunications
- Transportation

Affected Systems

- Users/Application and Software
- Enterprise/Technologies Support Infrastructure
- Control Systems and Applications
- Equipment Under Control
- Operations Management
- Safety Protection
- Industrial Network Protocols

Intended Effects

- Disruption
- Degradation
- Destruction
- Interference with ICS
- Military Advantage
- Political Advantage

Motivations

- Military/Security/Diplomatic

Malware Families

- INCONTROLLER
  Aliases
  - INCONTROLLER
- OMSHELL
  Aliases
  - OMSHELL
- ICECORE
  Aliases
  - ICECORE
- CODECALL
  Aliases
  - CODECALL
- TAGRUN
  Aliases
  - TAGRUN

Tactics, Techniques And Procedures (TTPs)

- Exploit Development
- Malware Propagation and Deployment
- Malware Research and Development
- Network Reconnaissance
- Distributed Denial-of-Service (DDoS) Attack

Targeted Information

- IT Information

Version Information

Version:3.0, April 13, 2022 09:12:08 PM

Common Vulnerabilities and Exposures

CVE ID:	CVE-2020-15368(CVE Description)Mandiant Vulnerability Analysis