Kaspersky ICS CERT

kaspersky

ISaPWN – research on the security of ISaGRAF Runtime

Alexander Nochvay
Artem Zinenko
Evgeny Goncharov

23.05.2022 Version 1.0

Executive Summary

In early 2020, we notified the Rockwell Automation Product Security Incident Response Team (RA PSIRT) of several vulnerabilities we had identified in the ISaGRAF Runtime execution environment.

According to public sources of information, ISaGRAF Runtime is used as an automation framework in multiple products in various industries across the globe and its use is not limited to ICS. ISaGRAF Runtime are also used in transportation, power & energy, and other sectors.

This report includes an analysis of the ISaGRAF framework, its architecture, the IXL and SNCP protocols that are used to program and control ISaGRAF-based devices and to communicate with them.

Our research has uncovered multiple vulnerabilities in ISaGRAF Runtime. The following potential vectors of attacks on ISaGRAF-based devices have been identified:

- A remote unauthenticated attacker could execute privileged commands of the IXL service on devices with ISaGRAF Runtime versions released before 2010.
- A remote attacker could easily implement a password brute force attack in ISaGRAF Runtime.
- An attacker that can carry out a MitM attack will be able to overwrite tag statuses, the program being downloaded to the device, or authentication data. Since authentication data is encrypted with a preset symmetric key, the attacker could decrypt an intercepted target (device) password.
- An attacker could exploit the vulnerabilities identified to gain remote access to a device with ISaGRAF Runtime and execute arbitrary malicious code inside the ISaGRAF Runtime virtual machine.
- An attacker could exploit the vulnerabilities to escape the ISaGRAF Runtime sandbox, ensure the malicious code's persistence on the device, and hide it from future detection.

Detailed descriptions of the vulnerabilities identified are provided, along with an analysis of the impact that their potential abuse could have and recommendations on additional risk mitigation measures.

By the end of 2021, all of the vulnerabilities identified had been fixed by the technology vendor, or mitigations were suggested by the vendor, CISA, or Kaspersky ICS CERT.

As of March 2022, the following vendors had reported ISaGRAF Runtime vulnerabilities in their products: Rockwell Automation, Schneider Electric, Xylem, GE, and Moxa.

For more information, please contact ics-cert@kaspersky.com.

ISaGRAF technology

ISaGRAF is a programming technology and execution environment for programmable logic controller (PLC) programs. It includes three components:

- **ISaGRAF Runtime**, an adaptable resource execution environment. The environment ported and customized for a specific PLC is called an ISaGRAF target.
- **ISaGRAF Workbench**, an application development environment for ISaGRAF Runtime. It provides the development, compilation, and debugging of resources[1]; downloading resources to the controller (it includes all the controller management functions required for this).
- **ISaGRAF Runtime Toolkit**, a tool for developing drivers and adapting the ISaGRAF Runtime execution environment for the target software and hardware platform (OS and PLC).

Unlike CODESYS (see our three-part article: 1, 2, 3), where an application is compiled into the controller's machine code, ISaGRAF resources are compiled into TIC (target-independent code) and are executed by the ISaGRAF Runtime environment. ISaGRAF Runtime is essentially a virtual machine with TIC serving as virtual code.

The application development environment can also be customized and extended.

There are several different development environments. ACP (Automation Collaborative Platform) is the main development environment provided by Rockwell Automation (RA) to developers of controllers based on their framework. ISaGRAF Workbench is part of ACP.

RA allows controller manufacturers to provide it to its end users (e.g., integrators and industrial enterprises) free of charge. Additionally, there are dedicated development environments for specific device families, such as AADvance Workbench for the AADvance family and SCADAPack Workbench for the SCADAPack family.

The Windows version of ISaGRAF Runtime is distributed as part of ACP.

[1] The ISaGRAF technology uses the term "resource" instead of "application".

ISaGRAF Runtime

There are currently three major lines of ISaGRAF Runtime: v1, v3, and v5. Each of these versions has a dedicated version of ISaGRAF Workbench. The latest version, ISaGRAF Workbench v6, is distributed with support for ISaGRAF Free Runtime v3 and v5 used as emulators of ISaGRAF Runtime.

Object of research

We used files obtained after installing ACP and a 2016 demo version of ISaGRAF Runtime 5.70.32 for Linux и Windows as the object of research.

Additionally, we analyzed the following versions of ISaGRAF Runtime:

- ISaGRAF Runtime in the firmware of IoPAC 8500 devices – 5.40.148_(09/11/2013);
- ISaGRAF Free Runtime in AADvance Workbench – (Build 2009.1.10.501_(090416));
- ISaGRAF Free Runtime in ISaGRAF Open – Build 2006.5.10.000_(060301);
- ISaGRAF Free Runtime в ICS Triplex ISaGRAF – Build 2009.1.10.501_(090416).

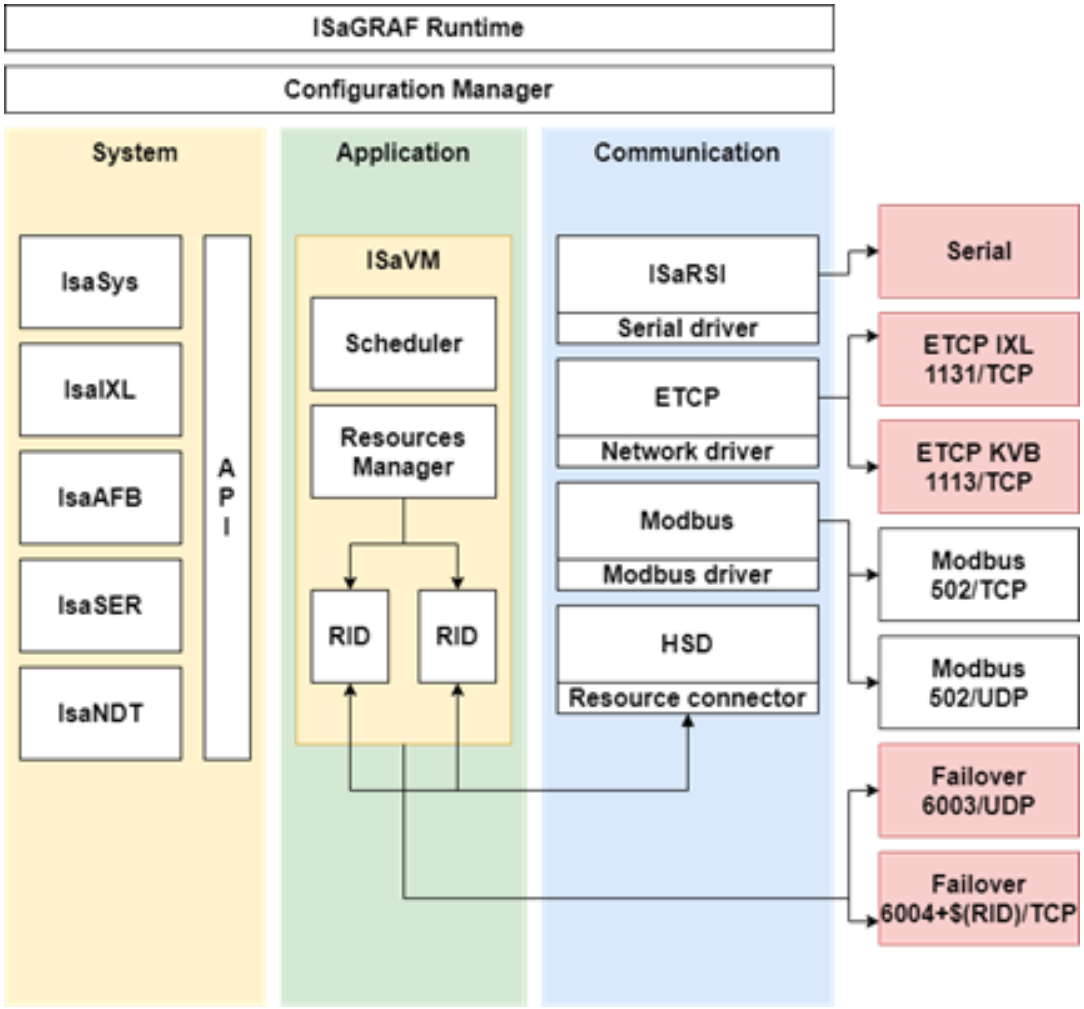
Components of ISaGRAF Runtime

Each component of ISaGRAF Runtime is either a dynamic library or an executable file. Each executable object is configured via a configuration file that has the same name or command-line argument.

Main components and component groups:

- Configuration Manager – the main component that launches other executable components;
- System Components – a group of libraries that provide APIs for working with configuration files, the memory, strings, files, etc.;
- Application – the ISaGRAF virtual machine component;
- Communication components – executable components that implement the communication with ISaGRAF Runtime over Ethernet or RS232/RS485 connections.
  It is important to note here that communication with the development environment is implemented by the Configuration Manager component.

Components communicate with each other via shared memory. A proprietary protocol is used to transfer messages.



ISaGRAF Runtime components

Configuration Manager

As mentioned above, the Configuration Manager (**hereinafter CM**) is the main component that initializes all other components. CM is launched by the `ISaGRAF` executable file (`ISaGRAF.exe`). CM is controlled via the `ISaGRAF.ini` configuration file and command-line arguments.

Main tasks performed by CM:

- Initialize shared memory for all other components;
- Initialize communication drivers (HSD, ETCP, and ISaRSI);
- Launch communication components (ISaRSI, ETCP);
- Launch virtual machines (ISaVM) and specify the resources to be loaded by them;
- Initialize the IXL network service on port 1131/TCP.

Compiled program and virtual machine

The format of compiled applications is comparable to the Windows Portable Executable file format in terms of complexity: a compiled application also has constant initialization, default data initialization, variable, code, and resource sections and a description of requirements for virtual machines that can execute the compiled file.

CM can launch several virtual machines simultaneously to run several applications – one virtual machine per application. If necessary, virtual machines can also communicate with each other via shared memory.

IXL protocol

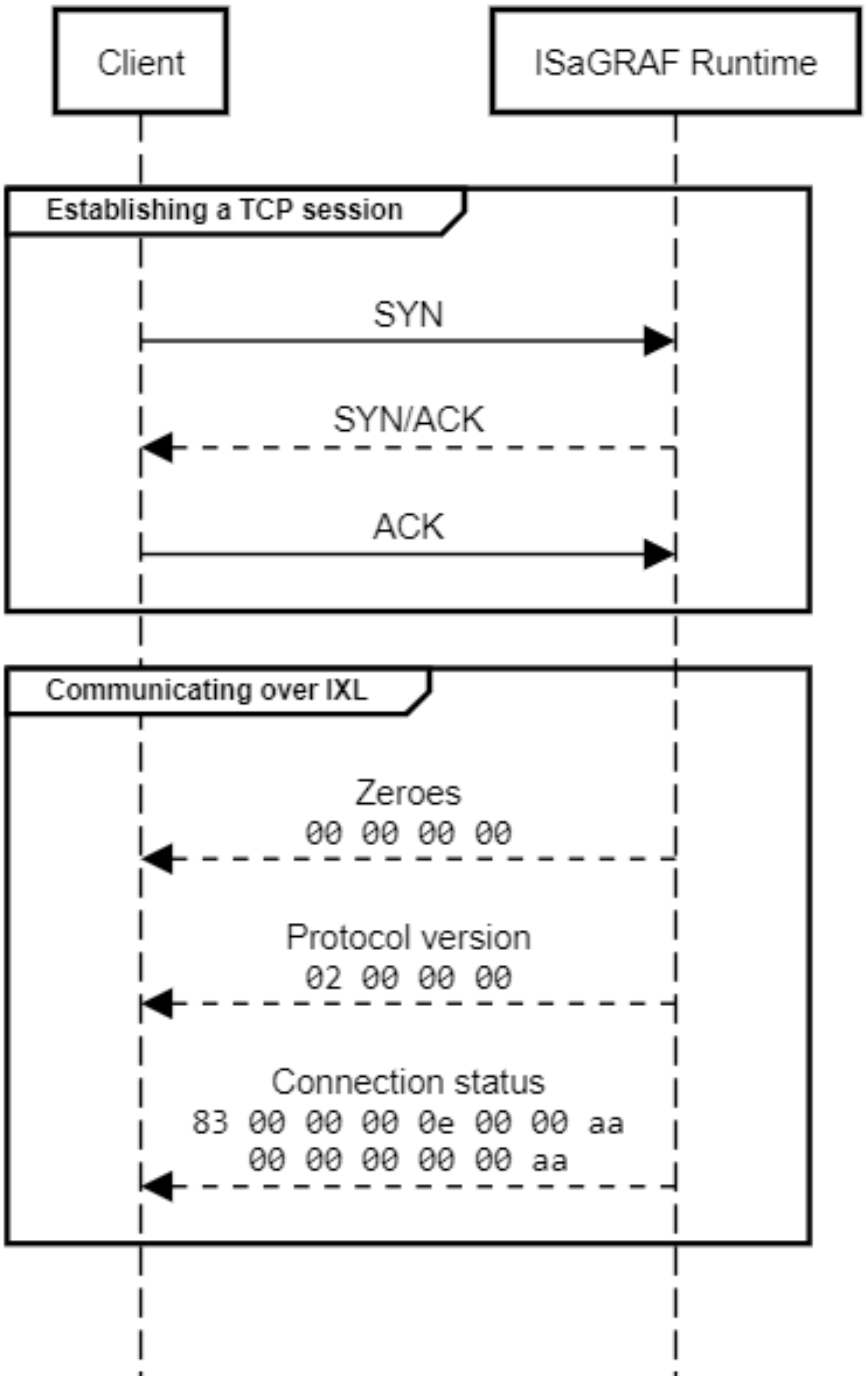
A device with ISaGRAF Runtime is programmed over Ethernet or a serial interface. Communication over a serial interface is provided by the ISaRSI[2] component, communication over Ethernet – by CM.

The development environment remotely performs basic operations with the PLC, such as programming the device, loading an application, debugging, and changing the configuration. To perform these basic operations over Ethernet, ACP communicates with ISaGRAF Runtime on port 1131/TCP. The communication is carried out over IXL (ISaGRAF eXchange Layer), a proprietary protocol designed for the exchange of data between the development environment and CM.

To establish a connection, the server node sends several service messages to the IXL client. The first service message contains four zero values (`00 00 00 00`). The second message contains the IXL protocol's version.

ISaGRAF Free Runtime reports version two of the protocol (`02 00 00 00`), the AADvance T900 emulator, version one (`01 00 00 00`), and the SCADAPack emulator, version zero (`00 00 00 00`).

[2] We believe that the acronym ISaRSI may stand for ISaGRAF Remote Serial Interface.



Communication between ACP and ISaGRAF Free Runtime over the IXL protocol

The format of IXL messages is very simple. Each command includes two fields – header and body. The constant `0xAA` is placed at the end of each body or header field. No checksum is used.

Format of the header field

A header contains the following data: command identifier, message type, error flag, message size, and resource identifier. A header has the following format:

Field	Type	Description
Direction	1 bit	Specifies message direction: 0 corresponds to a request, 1 to a response
Status	1 bit	Flag indicating whether an error occurred when handling the request
Command	6 bit	Command identifier
Packet size	4 byte	Size of the entire packet
Resource ID	2 byte	Resource identifier
Ending	1 byte	Constant `0xAA` indicating the header's ending

The Resource ID field defines the resource to which the command will be applied. CM uses the value to determine to which virtual machine the request is to be passed. A request is passed to a virtual machine via the ISaGRAF HSD driver, which uses shared memory to pass the data.

The Command field contains the command identifier. The IXL protocol defines the following commands:

- `READ` (`0x1`) – get variable values
- `WRITE` (`0x2`) – change variable values
- `CONNECT` (`0x3`) – complete establishing a connection, only sent as a response
- `DISCONNECT` (`0x4`) – terminate the connection
- `SUBCMD` (`0x6`) – execute a command from CM
- `START_DIALOG` (`0x7`) – start a dialog to execute system commands from CM
- `STOP_DIALOG` (`0x8`) – stop a dialog started to execute system commands from CM

An example of parsing data for the `CONNECT` command sent from ISaGRAF Runtime to the client side is provided below:

```
tcp.payload:
0000  83 00 00 00 0e 00 00 aa 00 00 00 00 00 aa

header:
0000: 83 = 0b100000011
             1.............. - Direction response
             .0.............. - Status OK
             ..............11 - Command CONNECT
0001: 00 00 00 0e = 14 - packet size
0005: 00 00 = 0 - resource id
0007: aa - mask ending

body:
0008: 00 00 00 00 00 aa
```

The receipt of such a packet means that an IXL connection has been successfully established.

Format of the body field

The format of the body field depends on the Command value and the Direction flag.

Example of the body field for a CONNECT response

The body value in the example from the Format of the header field section (see above) can be parsed as follows:

```
tcp.payload:
0008: 00 00 00 00 00 aa

body:
0008: 00 00 00 00 - error number
0012: 00 - Internal ID status
0013: aa - mask ending
```

The body field of the `CONNECT` command contains two error identifiers at the same time and the end-of-field constant.

Example of the body field's format for the READ command

The body field for the `READ` command contains the number of elements whose values are requested. A request example is shown below:

```
body data:
0008: 00 02 00 00 01 24 02 00 00 00 01 00 00 00 04 00 00 15 c8 03 00 00 00 01 00 00 00 04 aa

body:
0008: 00 02 - number of variables is 2

first item:
0010: 00 00 01 24 02 00 00 00 01 00 00 00 04

    parsed:
    0010: 00 00 01 24 - id of item
    0014: 02 - type of item is signed integer
    0015: 00 00 00 01 - default value
    0019: 00 00 00 04 - size of item

second item:
0023: 00 00 15 c8 03 00 00 00 01 00 00 00 04

    parsed:
    0023: 00 00 15 c8 - id of item
    0027: 03 - type of item is integer
    0028: 00 00 00 01 - default value
    0032: 00 00 00 04 - size of item

0033: aa - mask ending
```

The first two bytes define the number of variables requested. The variable element has the following structure: variable identifier, variable type, default value, and variable size. In the response, the default value will be replaced with the variable's current value.

Example of the body field's format for the SUBCMD command

The body field for the `SUBCMD` command defines only the subcommand value, on which the format of further data depends. At the same time, the purpose of a subcommand depends on the Resource ID field. If the client needs to execute a CM system command, the constant `0x0FFF` is used as the Resource ID value. Examples of system commands include getting the number of running resources or loading a configuration file. If a resource needs to be loaded, cleared, or removed, the constant `0xFFE` is used. Finally, if it is necessary to pass authentication or set a password, the Resource ID value is ignored.

In some cases, the body field may be absent altogether, such as in the case of the `SUBCMD RESOURCE_GET_STATE_EXTENDED` (`0x1E`) command:

```
body data:
0008: 1e aa

body:
0008: 1e - SUBCMD id value RESOURCE_GET_STATE_EXTENDED
0009: aa - mask ending
```

On the other hand, for the `SUBCMD RESOURCE_LAUNCH (0x50)` command, additional fields must be included:

```
body data:
0008: 50 00 00 01 00 00 aa

body:
0008: 50 - SUBCMD id value RESOURCE_LAUNCH
0009: 00 - run mode
0010: 00 01 - resource id to launch
0012: 00 - length of argument
0013: aa - mask ending
```

Additional fields for the `RESOURCE_LAUNCH` command contain the identifier of the resource to be launched, argument sizes, the arguments themselves (not present in the above example, since their length is equal to 0), and the mode in which the resources are to be launched.

We were able to find about 65 command handlers for ISaGRAF Free Runtime. The list of handlers may be different for other ISaGRAF targets. All commands can be placed in the following groups:

- Application debugging commands: performing one instruction, getting a list of breakpoints, getting the call stack, removing or adding breakpoints;
- File system commands: uploading a configuration file, uploading or removing an arbitrary file, uploading a file module, and uploading an application via a file;
- Commands for authentication and changing the target password;
- Commands for working with a resource: checking its status, checking the checksum, stopping, launching, getting basic information, getting the virtual machine's capabilities.

SNCP protocol

It should be noted that not all devices with ISaGRAF Runtime use port 1131/TCP to communicate over IXL. For example, according to documentation for the AADvance Controller device, network access to port 1132/TCP is "always" required for ISaGRAF to work properly:

AADvance Communication Ports

Protocol	Port Number	Port Open	Purpose	Port Open When
TCP	502	When configured	Modbus TCP slave	Open if Controller is configured as a Modbus TCP slave.
TCP	1132	Always	ISaGRAF®, application downloads, debug, SoE etc.	N/A
TCP	2000	When configured	Modbus RTU slave	Open if controller is configured as a Modbus RTU slave. The default port, 2000, is given in this table.

Fragment of AADvance Controller documentation

In TCP and UDP Port Configuration – Quick Reference Guide, "SNCP" is specified as the name of the protocol that uses port 1132/TCP:

1132	TCP	SNCP	AADvance	Safety Network Control Protocol, used by OPC, workbench debugger and binding networks

Fragment of TCP and UDP Port Configuration – Quick Reference Guide

SNCP encapsulates the IXL protocol, adding fields for verifying the integrity of data and the message source. SCNP uses the following message format:

Field	Type	Description
Magic	2 byte	Magic constant `0xCCCC` – identifier of an SCNP message
ixl_crc_meta	2 byte	Information on the type of the checksum used in the `ixl_crc` field – the value `0x0201` indicates the use of CRC64, otherwise CRC32 is used
message_id	2 byte	The packet's message identifier
ixl_body_size	2 byte	Size of `ixl_body`
ip_src	4 byte	Message source IPv4 address
ip_dst	4 byte	Message destination IPv4 address
ixl_crc	4/8 byte	Checksum of `ixl_body`
crc32	4 byte	Checksum of the SNCP header
ixl_body	ixl_body_size	Message in the IXL format

Vulnerabilities identified and possible attack vectors

In total, we have notified Rockwell Automation PSIRT of 9 vulnerabilities that can be exploited by a remote or local attacker.

Below we discuss attack scenarios for an ISaGRAF Runtime device being hijacked by a remote unauthenticated attacker by exploiting the vulnerabilities identified. The attacker's ultimate goal is to escape the restricted environment of ISaGRAF Runtime and take control of the device.

Gaining administrative access to CM

The IXL service defines some commands as privileged. Authentication with the target password is required to execute these commands. Privileged commands interact with the file system, the application, the ISaGRAF Runtime configuration, and control of the device's operating mode.

A successful attack on authentication with the IXL service enables the attacker to hijack the device and execute arbitrary code inside the ISaGRAF Runtime virtual machine.

Lack of authentication in ISaGRAF Runtime versions prior to 2010

The first vulnerability is the lack of authentication in ISaGRAF Runtime versions prior to 2010.

It is hard to identify the specific versions of ISaGRAF Runtime that are vulnerable because many vendors use their own version systems. Rockwell Automation does this, for example, for AADvance and ICS Triplex products. However, the version name of each ISaGRAF Runtime copy that we have analyzed includes the year. Therefore, we can provisionally refer to vulnerable ISaGRAF Runtime versions as versions prior to 2010.

We have confirmed that this vulnerability exists in ISaGRAF Free Runtime, which is part of different development environments. In all versions created prior to 2010, there was a stub instead of handlers for the authentication command, `PASSWORD_CHECK`.

The example below shows pseudocode for the PASSWORD_CHECK command in AADvance Workbench and ACP:

Configuration Manager – Build 2009.1.10.501_(090416)

```
01: subcmd_id = *body_data;
02: if (subcmd_id == PASSWORD_CHECK) {
03:   response->command = PASSWORD_CHECK;
04:   response->error = ISA_RC_SUCCESSFUL;
05:   ixsMsgProc(param_1->field_0x0,
param_2, response, 2, 0x10);
06: }
```

Configuration Manager – Build 5.72.00.142_(02/12/2021)

```
01: subcmd_id = *body_data;
02: if (subcmd_id == PASSWORD_CHECK) {
03:   dsysMemReset(recved_password, 9);
04:   dsysMemCpy(recved_password,body_data + 1, 8);
05:   dsysDecryptPassword8(recved_password, recved_password);
06:   if (IXDINFO.password[0] == '\0') {
07:     param_1->authorized = true;
08:   }
09:   else {
10:     compared = dsysStrCmp(recved_password, IXDINFO.password);
11:     param_1->authorized = compared == 0;
12:   }
13:   if (param_1->authorized == false) {
14:     error = ISA_RC_PASSWDINVALID;
15:   }
16:   else {
17:     error = ISA_RC_SUCCESSFUL;
18:   }
19:   response->command = PASSWORD_CHECK;
20:   response->error = error;
21:   ixsMsgProc(param_1->field_0x0, param_2, response, 2, 0x10);
22: }
```

It can be seen in the pseudocode that there is a stub for the PASSWORD_CHECK command in the 2009 version, which always returns the status ISA_RC_SUCCESSFUL (i.e., authentication completed). In the 2021 version, CM extracts the password from the body (line 04), decrypts it (line 05), and then compares it to the value in memory (line 10). If the values match, the handler returns ISA_RC_SUCCESSFUL (line 17), otherwise ISA_RC_PASSWDINVALID (line 14).

Thus, a remote attacker can execute privileged commands of the IXL service without passing authentication on devices that have ISaGRAF Runtime versions released before 2010.

Bruteforcing the target password

The target password is a case-sensitive alphanumeric string that is only up to 8 characters long. No protection against password bruteforcing is implemented in ISaGRAF Runtime, so the threat of a password being bruteforced on devices with ISaGRAF Runtime is quite relevant.

In the process of executing the PASSWORD_CHECK command, the encrypted target password is passed to the body. The target password is encrypted using the Tiny Encryption Algorithm (TEA) with preset keys.

The password encryption function in Python is shown below:

```
from ctypes import c_uint32
from construct import Int32ul, Int32ub, PaddedString

K1 = <REDACTED>
K2 = <REDACTED>
KEYS = [K1, K2, -K1, -K2]
DELTA = 0x9e3779b9

def encipher(v, k):
    y = c_uint32(v[0])
    z = c_uint32(v[1])
    sum_value = c_uint32(0)
    n = 32
    w = [0, 0]

    while (n > 0):
        sum_value.value += DELTA
        y.value += (z.value << 4) + k[0] ^ z.value + \
            sum_value.value ^ (z.value >> 5) + k[1]
        z.value += (y.value << 4) + k[2] ^ y.value + \
            sum_value.value ^ (y.value >> 5) + k[3]
        n -= 1

    w[0] = y.value
    w[1] = z.value
    return w

def encrypt_password(password: str):
    if len(password) == 0:
        raise BaseException
    _ = PaddedString(8, "ascii").build(password)
    encrypted = encipher([Int32ul.parse(_[:4]), Int32ul.parse(_[4:])], k=KEYS)
    return Int32ub.build(encrypted[0]) + Int32ub.build(encrypted[1])
```

So all that remains to carry out an attack is to implement establishing a connection with the CM network service over the IXL protocol and executing the `PASSWORD_CHECK` command with a possible password.

Encrypting the password with a symmetric algorithm using a preset key and MiTM

Data sent over the IXL protocol is not encrypted and ISaGRAF products have no way of protecting data by encrypting data transfers. If no other security tools are used to protect communication and an attacker can carry out a MitM attack, the attacker will be able to overwrite tag statuses, the program that is being downloaded to the device, or authentication data. Since authentication data is encrypted with a preset symmetric key (CVE-2020-25180), the attacker can decrypt an intercepted target password.

A decryption function written in Python is as follows:

```
from ctypes import c_uint32
from construct import Int32ul, Int32ub, PaddedString

K1 = <REDACTED>
K2 = <REDACTED>
KEYS = [K1, K2, -K1, -K2]
DELTA = 0x9e3779b9

def decipher(v, k):
    y = c_uint32(v[0])
    z = c_uint32(v[1])
    sum_value = c_uint32(0xc6ef3720)
    n = 32
    w = [0, 0]

    while (n > 0):
        z.value -= (y.value << 4) + k[2] ^ y.value + \
            sum_value.value ^ (y.value >> 5) + k[3]
        y.value -= (z.value << 4) + k[0] ^ z.value + \
            sum_value.value ^ (z.value >> 5) + k[1]
        sum_value.value -= DELTA
        n -= 1

    w[0] = y.value
    w[1] = z.value
    return w

def decrypt_password(encrypted):
    if len(encrypted) == 0:
        raise BaseException
    c = decipher([Int32ub.parse(encrypted[:4]),
                  Int32ub.parse(encrypted[4:])], k=KEYS)
    return PaddedString(8, "ascii").parse(Int32ul.build(c[0]) + Int32ul.build(c[1]))
```

Mitigation measures suggested by Rockwell Automation include properly configuring the firewall and segmenting the network. It should be added that the protocol does not require the session value to be sent, and after successful authentication, the service remembers only the client's IP address.

Sandbox escape

ISaGRAF Runtime developers have implemented a command for downloading an arbitrary file via the IXL service. However, the name of the file is not checked against lists of allowed filenames or for the presence of special characters. This leads to a path traversal vulnerability (CVE-2020-25176) that can be exploited to achieve RCE.

For example, an attacker can download any file to the device via the `DOWNLOAD_FILE_PACKET` (`0x3b`) command by manipulating its name and using special characters in it.

Given that CM looks for some executable files using filename masks (CVE-2020-25182), all an attacker needs to do to execute arbitrary code is to download an executable file with the right name, such as downloading a file with the name "`IVMdead`" and sending a command for the resource with identifier 57005 to be executed. This will result in CM launching the `IVMdead` executable file.

This vulnerability can be exploited to escape from the restricted environment of ISaGRAF Runtime and achieve persistence on the device. Possible attackers can exploit this vulnerability to mask their presence on an ISaGRAF Runtime device.

To fix this vulnerability, Rockwell Automation has added checking filenames for special characters.

Conclusions

Our research demonstrates once again how serious the issue of vulnerabilities in third-party components is. Although it was not difficult to identify the vulnerabilities described above, getting these vulnerabilities fixed in end products takes a very long time.

In the case of ISaGRAF, to find out that a product is vulnerable, its user needs to wait for Rockwell Automation to fix the vulnerabilities and release an advisory and then for the product's vendor to do the same. In some cases, the ISaGRAF supply chain is even longer.

As of March 2022, the following vendors had reported vulnerabilities in their products:

- Rockwell Automation
- Schneider Electric
- Xylem
- GE
- Moxa

Therefore, we recommend using the defense-in-depth approach and, in addition to installing updates, making a special emphasis on preventing unauthorized network access on ports 1131/TCP and 1132/TCP.

Please write to ics-cert@kaspersky.com on any issues related to ensuring the security of endpoint devices built on the ISaGRAF technology, as well as on any issues related to this research.

Kaspersky Industrial Control Systems Cyber Emergency Response Team (Kaspersky ICS CERT) is a global project of Kaspersky aimed at coordinating the efforts of automation system vendors, industrial facility owners and operators, and IT security researchers to protect industrial enterprises from cyberattacks. Kaspersky ICS CERT devotes its efforts primarily to identifying potential and existing threats that target industrial automation systems and the industrial internet of things.

Kaspersky ICS CERT ics-cert@kaspersky.com