1.4 – Impact of security on safety

Practical guidance – cobots (collaborative robots)

Author: Panayiotis Karachristou and John Clark, University of Sheffield

Threat modelling

The process of identifying security threats is known as threat modelling. This is accomplished through the use of abstractions. This operation's output is commonly referred to as a threat model.

Threat modelling in robotics defines risks/threats associated with the robot. This can occur in both software and hardware components. Furthermore, threat modelling provides a method for resolving or mitigating the previously identified threats [1].

Threat modelling is the process of determining what might go wrong in terms of security with the system in question. Why is this essential? Because a good threat model allows you to address classes of attacks rather than individual attacks, resulting in the delivery of a much more secure product [2].

Cobot threat modelling lifecycle

In order to assist us with the threat modelling of a cobotics system, we introduce a new novel procedure, the Cobot Threat Modelling Lifecycle.

Figure 1 - Cobot Threat Modelling Lifecycle

The cobot threat modelling lifecycle as depicted in Figure 1 is composed of six different steps. These steps are:

1. Creating a diagram of your cobot system. In this example a high level diagram was used, but in reality any kind of diagram can be used, the right one is the one more helpful in better understanding how the system works.
2. Define the security requirements, or in other words what are you trying to achieve? What is your aim?
3. Identify: actors, adversaries, assets and entry points. This step is needed for the same reason step 2 is needed, which is to help better understand how the system works.
4. Identify threats. Identify all threats in the system using the STRIDE mnemonic.
5. Mitigate threats. Find mitigations to the threats previously identified.
6. Validate that all the threats have been mitigated. The mitigation techniques previously identified may identify new threats. For example, introducing a camera in the system for face recognition to authenticate users, introduces a new entry point to the system which in turn introduces new threats. If all threats are validated you can then go in maintenance mode. If on the other hand, you find that any new entry points, assets, threats etc. during this process have been added to the system then you iterate again.

Maintenance - decommissioning

When the system successfully finishes the validation phase it then should enter the maintenance phase. In this phase updates can be made to the system in a secure manner. Moreover, it should be noted that things change and items are sometimes introduced or decommissioned in the system. When this happens the threat modelling lifecycle should be applied to the system again and when an item is decommissioned form the system it should be done in a secure manner making sure no sensitive information is stored on it.

1. Step 1 - example system diagram

The system which is depicted below is based on the teconnex cobot cell. The Arc-welding system is hosted by a specialised manufacturer which uses cobots in order to weld metallic parts which it produces. Figure 2, depicts a high-level diagram of the cobot system in question.

Copyright © 2021 University of York

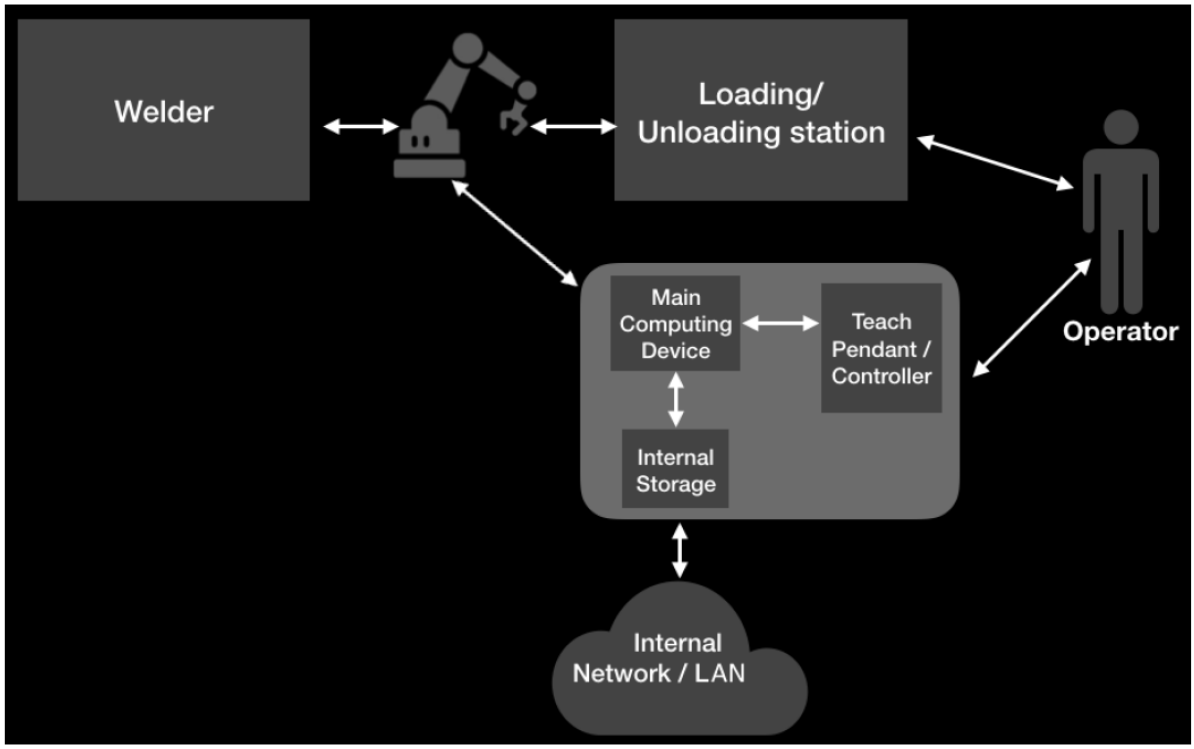

Figure 2 - high level diagram

As we can see the system in question consists of the cobot itself, a welder, the station where the cobot picks up and returns parts, the main computing device which is the device that sends commands to the robot which also has internal storage and it is controlled by the cobot operator through the teach pendant. Lastly, we can see that this system is connected to the internal network/LAN.

2. Step 2 - define security requirements

This section defines the requirements the cobot system depicted in Figure 2 needs to meet. The requirements are all high-level security objectives. A security requirement is a description of the required security functionality that ensures that one of the many different security features of the software is met. The security requirements can be found by asking two questions:

1. What are you trying to achieve?
2. What is your aim?

The requirements are identified using the Arc-welding example which is explained in section are:

- **Access**. The system should only be accessible by its operator / authorised user including any hardware, software and ports.
- **Authentication**. It should be able to maintain user authentication.
- **Accuracy**. The cobot needs to read accurate sensor measurements and issue correct and accurate controls on actuators so that movements are made within acceptable

Copyright © 2021 University of York

margins of error. The security measures taken should not compromise the speed, reliability and reaction times of the system.

- **Resilience**. The software of the cobot system must remain resilient in the face of attacks.
- **Availability**. The system must be available and behave reliably even under DOS attacks.
- **Predictability**. The behaviour of the system needs to be correct and predictable.
- **Confidentiality**. The system must ensure the privacy of the stored information.
- **Secure updates**. System should be securely updated.
- **Safety**. Adequate and correct information should be provided by the cobot to allow operators to take safe and informed decisions. Operators should be able to execute emergency procedures fast and safely.
- **Integrity**. The cobot controller should minimise the possibility of the cobot being damaged by badly written control logic.

3. Step 3 - identify actors, adversaries, assets and entry points.

Actors

Actors are people or systems that come into contact with the cobot. Acknowledging them helps determine how the system can be compromised, by considering the actors who interact with it. For example, actors could issue commands which would abuse or attack the system. Actors can be divided into several groups, depending on whether or not they are physically present next to the cobot [3].

In figure 2 it can be seen that in the context of the Arc-welding cell, there is a singe actor, the operator. His job is to assemble a part and place it on the loading station. Once placed the cobot then picks up the part and takes is to the welder to weld. Once the welding is fiished it is placed by the cobot back on the station for the operator to receive. No other actor to our knowledge exists in the teconnex environment. In order to make this project more general it will be assumed that there is an extra actor. Therefore, the actors will be divided into two categories, cobot operator and cobot developer. The developer is the user who, updates and reprograms the cobot, therefore he will have all the privileges the operator has in addition to administrative privileges.

Assets

Assets of a system represent anything of value that needs to be secured against attacks. These include any device, user, resource or property of said system. Assets can be classified using the CIA Triad model of Confidentiality, integrity and availability.

- **Confidentiality**, essentially, is concerned the privacy of the system. It is the process of maintaining information that should be private and confidential, private and confidential. The data of the system should not in any way be accessible by attackers. The cobot systems can have two types of data - sensor data and robot data (software, logs, etc.). None of these data should be accessible by unauthorised users.

Copyright © 2021 University of York

- **Integrity** assets are the assets that deal with the behaviour of cobot, no attacker should be able to modify it. Integrity assets, can be further broken down into five categories. Physical safety, which means the robotic systems should not in under any circumstance damage their users or their surroundings. Secondly, robot integrity, meaning the system should not be able to damage itself. Robot Actuators Command Integrity, on the other hand demands that unhallowed actors should not have the ability to control the actuators of the cobot. Another category is Robot Behaviour Integrity assets, this category states that the robotic system shall not allow unauthorised actors to interrupt its tasks. Lastly, we have Robot Data Stores Integrity which is concerned with robot data, the robot data should not in any case be modified by an attacker.

- **Availability** makes sure that all systems and data that should be available remain available. The cobot should be able to operate even under attack. The availability of the cobot can be broken down into three categories - Compute Capabilities, Robot Availability and Sensor Availability. The cobot should be able to maintain its Compute Capabilities, it should not stop from functioning correctly if its starved out from its computer resources. Robot Availability means the robot should be always be available and respond in a reasonable time in any command. Lastly, we need to have sensor availability which means that sensor data should always be available shortly after production to any authorised users.

These assets are represented in more detail and in the context of the Arc-welding cell in table 1.

Category	Asset	Description
Confidentiality	Robot Data Privacy	Data should not be accessible by unauthorized users. e.g. Logs, Software
Integrity	Physical Safety	Cobot must not harm the user or its environment (surroundings)
	Robot Integrity	Cobot must not damage itself
	Robot Actuators Command Integrity	Unauthorized users should not be able to control the cobot actuators
	Robot Behaviour Integrity	Cobot tasks should not be disrupted by attackers
	Robot Data Integrity	Unauthorized users should not be able to change robot data
Availability	Compute Capabilities	Starving cobot from its resources should not disrupt its operation
	Robot Availability	Cobot must answer commands in a reasonable time.

Table 1: Assets in the Context of the Arc-welding Cell

Source adapted from: [3]

Entry Points

Entry points portray how the device communicates with the surrounding environment, for example, channels of communication, ports, sensors etc. An easy way of finding the entry points of your system is by asking the question, "how do actors interact with the system?". Table 2, describes the entry points identified in the teconnex cell environment.

Copyright © 2021 University of York

Entry Point	Description
Robot Administration Tools	Tools allowing local users to connect to the robot computers directly (e.g. SSH, VNC)
Teach Pendant	The cobot interface device
Robot Code Deployment Infrastructure	Deployment infrastructure for binaries or configuration files are granted read/write access to the robot computers filesystems
Sensors	Sensors are capturing data which usually end up being injected into the robot middleware communication channels.
Embedded Computer Physical Access	External (HDMI, USB...) and internal (PCI Express, SATA...) ports

Table 2: System Entry Points for Arc-welding cell

Source adapted from: [3]

Adversarial models

Adversary modelling is a very important concept in cybersecurity. It the process of evaluating and formalising the potential attackers to your system. The DolevYao Model is one of the first and most commonly used adversary models [4].

Adversaries or actors are people or systems that come in contact with the system in question in one way or another. Adversary modelling aids the process of securing the system by knowing who you're defending against.

An adversarial model provides the attacking context in which our threat modelling and subsequent identification and deployment of countermeasures are carried out. We distinguish various levels of what an attacker can do and why.

We can categorise an attacker's context along various axes:

- **The type of attacker:**
    - Active employees. This is the hardest type to account for since they have physical access and good understanding of the system.
    - Former employees. This type of attacker is a high threat due to their high understanding of the system.
    - Competitors. A company wanting to get ahead of its competitors.
    - Dissident. A person or group of people wanting to attack the company expressing their disagreement with company policies.
- **The attacker's environment:**
    - Direct physical access to the cobot. Typically, this sort of adversary is always difficult to account for, and can pose as a disgruntled employee.
    - Passive RF capability. The attacker closer than 15 meters.
    - Active RF modulation capability. The attacker is in a 150-metre radius

Copyright © 2021 University of York

- Network attacker. Cobots are typically connected to a local area network and not the internet. In the limited cases they are this type of attacker has to be taken into account.

- **The resources the attacker has:**
  - Access privileges. The kind of privileges the attacker is able to gain.
  - Equipment/devices. The kind of tech equipment the attacker has in his possession.
  - Knowledge. How knowledgeable the adversary is.
  - Workforce. The number of people executing the attack.
- **The attacker's goals:**
  - Reason. Why is the adversary attacking the system?
  - Overall goal. What is the attacker wishing to obtain? e.g. data, money.
- **The adversary's capabilities:**
  - Send. Does the attacker have the ability to send data to the system?
  - Reveal. Is the adversary able to see data/information about the system?
  - Execute. Is the attacker able to execute any commands on the system?
  - Corrupt. Does the attacker have the capability to corrupt any data?

Axis	Attacker Context
Type	Active employees
	Former employees
	Competitors
	Dissident
Environment	Direct physical access to the cobot
	Passive RF capability (Closer than 15 Meters)
	Active RF modulation capability (150 meter radius)
	Network attacker
Resources	Access Privileges
	Equipment/Devices
	Knowledge
	Workforce. (Number of people)
Goals	Reason (Why is the adversary attacking?)
	Overall Goal (What are they after?)
Capabilities	Send data
	Reveal data
	Execute commands
	Corrupt data

Table 3: The attacker's context along various axes

Copyright © 2021 University of York

4. Step 4 - identify threats

STRIDE

The acronym STRIDE which stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege, was first introduced as a threat modelling approach by Loren Kohnfekder and Praerit Garg in 1999 [5]. This was developed to allow people who create software to recognise the types of attacks that their software might encounter. The sole goal of using STRIDE is to aid you into finding potential attacks to your system. It helps to categorise them and makes it easier to find mitigations to those threats.

Many of the properties you wish your program or system to possess are the opposite of STRIDE threats. These properties are authenticity, integrity, non-repudiation, confidentiality, availability, and authorisation. Table 4 explains into more detail the categories of STRIDE threats.

Threat	Property Violated	Threat Definition	Typical Victims
Spoofing	Authentication	Pretending to be something or someone other than yourself	Processes, external entities, people
Tampering	Integrity	Modifying something on disk, on a network, or in memory	Data stores, data flows, processes
Repudiation	Non- Repudiation	Claiming that you didn't do something, or were not responsible. Repudiation can be honest or false, and the key question for system designers is, what evidence do you have?	Process
Information Disclosure	Confidentiality	Providing information to someone not authorized to see it	Processes, data stores, data flows
Denial of Service	Availability	Absorbing resources needed to provide service	Processes, data stores, data flows
Elevation of Privilege	Authorization	Allowing someone to do something they're not authorized to do	Process

Table 4: The STRIDE threats

Source adapted from: [2]

STRIDE has certain desirable properties, these are properties which if your system possesses, it is safe from threats. These properties are seen in the second column of Table 4. Namely, they are, Authentication, Integrity, Non-Repudiation, Confidentiality, Availability and Authorisation.

- **Authentication** is the processes or behaviour that increases confidence that something is legitimate. Each person behind the keyboard is really allowed to use the program by entering a password. For instance, in most it is required for someone to provide his credentials (username and password) in order to operate a system. Different types of systems demand various authentication levels.

Copyright © 2021 University of York

- **Integrity** is explained in detail in the context of robot/cobot assets in section 3.2.2. In general integrity means that no-one should be able to interfere with the system to create unauthorised modifications this applies to every aspect of the system hardware, software, network or data.
- **Non-repudiation** is a scenario in which the subject cannot prove the authorship or legitimacy of the relevant claim successfully.
- **Confidentiality** refers to protecting the data of the system against unauthorised access. Only people with the correct credentials should be able to access the relevant information.
- **Availability** is explained in detail in the context of robot/cobot assets (see step 3). In short, the system should be always available no matter what, even if under attack and starved from its resources.
- **Authorisation** is the procedure used to decide the user/client privileges or access levels related to system resources, this includes any programs, files, services, data or application features. Usually after authorisation, authentication is carried out to verify the user identity.

Threats identified for example system

Threats were identified by using the STRIDE methodology described above.

Spoofing

1. Identity spoofing - an attacker spoofs his identity and poses as a user or administrator of the system, this could be achieved either by accessing the information needed illegally e.g. social engineering, Keylogger attack
2. An adversary spoofs a robot sensor (e.g. by replacing the sensor itself, or virtually by spoofing the signature of the device).
3. An attacker spoofs a robot actuator. This can be done the same way as the actuator, by replacing physically the actuator or virtually spoofing the actuators signature.
4. Attacker spoofs the robot trigger commands. e.g. spoofing the device that sends these commands, in the case of cobots this would most probably be the teach pendant.
5. Many vulnerabilities can be exploited with network and USB/micro SD ports having no validation. Robot joints can be controlled over these ports, robot actions updated/changed or configurations modified. Connecting a special USB device, that act as a keyboard, can type malicious commands to the robot or change settings.

Tampering

1. Tampering with robot state. The adversary alters the true robot status causing the true operator to lose control or get injured.
2. Tampering with files. An attacker tampers with the robot file-system, this could be achieved in a number of ways both physically and virtually, for example the attacker removes the internal storage.
3. Tampering with permissions. The attacker manipulates the system permissions. A most probable scenario would be giving himself administrative access and full control of the system, or denying someone access to it.
4. Tampering with the hardware. This is a more physical type of attack.
5. A user can accidentally misconfigure the robot.
6. An intruder modifies the robot file system by physically accessing it.

Copyright © 2021 University of York

7. Unprotected BIOS can allow altering of boot order enabling live boots.
8. Non-existence of lockout thresholds e.g. limit the number of failed login attempts, can allow an attacker to brute force attempts and gain access.
9. Invalid input parameter validation. This allows an attacker to perform command injection attacks.
10. If the Boot image that is downloaded isn't signed it opens a vulnerability for attackers to exploit and replace it with their own malicious file.
11. An attacker can maliciously trigger the safety features to perform a DOS attack.

Repudiation

1. Lack of error messages. No error message is returned when a certain failure happens.
2. Deletion of logs/data. An attacker manages to delete data or log files from the system.
3. An attacker can silently change the way a component is running on the robot from executing normally by making small unnoticeable changes.
4. Alarm suppression. Adversaries may try and attack any alarm functions on the cobot system and cause it to not work resulting in safety hazards.
5. In order to cover their tracks, adversaries may attempt to remove indicators of their presence on a system. When an adversary suspects that they are about to be discovered, they may attempt to overwrite, delete, or conceal changes they have made to the device.

Information disclosure

1. Sniffing the data. A way this attack could be achieved is by adding a malicious module on the main computer or even adding a sniffing device in the building that has the cobot.
2. Eavesdropping on the robot file system by physically accessing it.
3. An attacker steals log data. Usually log data is not encrypted therefore it makes for an easy source of information to an attacker.
4. An assailant intercepts the robot actuators command (can recompute localisation).
5. No mechanisms to prevent/control live boots can lead to leads to data exfiltration and elevation of privileges.
6. An attacker can easily access any unencrypted intellectual property.
7. An attacker could exploit a directory traversal vulnerability to access data. Clear text transmission of sensitive information can lead to easy information leakage by an attacker.

Denial of service

1. Flooding commands. An attacker repeatedly sends a lot of commands to the cobot.
2. Preventing robot communications (Jamming) An attacker uses a jamming device in order to prevent the transmitted data from reaching its destination.
3. Attacker cuts power to robot.
4. An attacker trigger the E-Stop. This can be achieved by sending a malicious command to the actuators to trigger the E-Stop.
5. An attacker drains the battery of the cobot. If the cobot is battery powered, an attacker can compromise the cobot software to send malicious commands which will drain the robot battery.

Copyright © 2021 University of York

6. An adversary modifies the command sent to the robot actuators. This can be done by intercepting and retransmitting the command.
7. An assailant overwhelms the robot disk with data. This could be achieved in a number of ways. A common scenario would be the attacker can send commands that require high computational power overwhelming the robot with data in the short-term memory and causing it to stop functioning properly.
8. Attempting to corrupt a device with the injection of a script or malicious application This application can be installed by an unknowingly by a different party or by the assailant themselves.
9. An attacker could exploit a software vulnerability (e.g race condition) to kill processes with root privileges.
10. An attacker could exploit a buffer overflow vulnerability to cause DoS.
11. An attacker could exploit a side channel attack vulnerability to cause DoS.
12. Attackers can cause a denial of service by causing an infinite loop on the system that can use up all the resources. A known example of this vulnerability is: “UnZip 6.0 allows remote attackers to cause a denial of service (infinite loop) via empty bzip2 data in a ZIP archive.”
13. Uncontrolled resource consumption If the resource consumption is not controlled in any way it allows attackers to perform DoS attacks.

Elevation of privilege

1. An assailant modifes the user permissions.
2. An attacker executes arbitrary code in the system. A way can be done by finding vulnerabilities in the code of the system.
3. An attacker gains access to the robot OS though its administration interface.
4. An unauthorised user connects to an exposed port.
5. An unauthorised user modifies configuration values. This can be achieved by finding a vulnerability in the robot operating system.
6. Running programs with elevated privileges, for example in Universal Robots UR-Caps (zip files containing Java powered apps) are executed with unbounded privileges.
7. Unauthenticated reading of robot data and unauthenticated writing of registers and outputs through RTDE interface.
8. Mismanaged permission implementation leads to privilege escalation, exfiltration of data and DoS.
9. Hard coded credentials also often referred to as Embedded Credentials, are plain text passwords or other secrets in source code, this is not a secure practice as attackers can easily gain access to this information and use it maliciously.
10. Bypassing user authentication is a venerability that can allow unauthorised access to a malicious actor in the system. An example of how this can be caused when an attacker uploads a malicious language _le to the system. This is actually a known vulnerability present on ABB robots running eSOMS version 6.0.2.
11. Lack of authentication parameters and validation allows attackers to do multiple attacks from damaging the robot with the removal of hardware limitation and the ability to gain access to hardcoded credentials.
12. Attackers can use built in system functionalities to their advantage. Example functionalities that may be used as such quoted from the UR 10 robot are:

Copyright © 2021 University of York

- A SubProgram command, a call to a sub program will run the program lines a sub program first, and then return to the following program line. This can be used to hide and run malicious code.
- Thread command. A thread is a process that runs in parallel with the robot program. A thread can be used to control an external machine that is not controlled by the robot arm. Variables and output signals can be used by a thread to communicate with the robot program. This can be exploited maliciously to run something simultaneously to the existing program.
- Commands can be looped indefinitely.

5 & 6. Steps 5 and 6 - mitigate threats and validate

Mitigation techniques

Mitigation techniques are policies or measures that need to be taken in order to prevent any breach of security. In order to produce these mitigation techniques, the threats found using STRIDE were taken into consideration.

- Continuous authentication. The continuous authentication of active physical users should be carried out to the operators of the cobot. This could be achieved by using NFC Tags or smart wearable devices the users.
- Teach Pendant should be password encrypted. This would be in addition to the continuous authentication as an extra layer of production, as the teach pendant is the main way a user interacts with the system.
- Each component should be authenticated. In the cobot system each component (main computing device, teach pendant, internal storage) should at some stage be authenticated this should be done in such a manner that the speed of the system is not compromised, for example it can be done only at boot and at certain random intervals. This could prevent the system from talking to any malicious devices.
- Communications should be encrypted (as much as possible). In addition to authenticating the components between them a good practice would be to encrypt the packets sent between these components. This should be done in order to prevent any attacks such as man in the middle. This should be done in ways that would not limit the systems abilities as this can be computationally expansive and cause delays to the system.
- Limit access to internal communications and buses as much as possible.
- File system should be encrypted (as much as possible). Just like the communications the anything that is stored in the file system should be encrypted so if it is stolen the thief cannot understand the information.
- Logs should not contain private data. Any logs that the system keeps should not be of value to an external entity of the system. If this isn't possible, any data that is logged and id of any importance should be encrypted.
- Joint commands should have limits. Limits should be set to the commands sent to the joints of the robots this is done in case an unauthorised user tries to manipulate the cobot. This measure is a fail safe to protect the system from harming itself and its surroundings
- Timely system updates. The system should be updated in a timely manner to make sure any new vulnerabilities found are patched.

Copyright © 2021 University of York

- Secure system updates. The updates of the cobot system should be carried out in a secure manner. Example ways this could be achieved are, signature on the packages of the updates, encrypted packages and authentication of the user making the updates.
- Effective logging of all security events. Any events that have to do with the security of the system, these logs should not contain any sensitive information and if the processing power of the system permits they should be encrypted.
- Ports should be accessible only by authorised users. This includes any ports, either external or internal. This could be ensured by the use of continuous authentication, if the user is authenticated and only then then the port should pass any data or power through it.

Threats for cobots outside the Arc-welding system scope

It should be noted that the system that the cobot threat modelling lifecycle is applied to in this paper is not connected to the internet, meaning does not include certain threats to the system that would otherwise be present. This section will go over some example threats and mitigations using the methodologies previously identified.

Spoofing

- An attacker spoofs a component identity. For example, the attacker spoofs the identity of a node and use it to cause the robot to stop.
- An attacker spoofs a robot sensor. An example scenario is, the robot system uses a poorly configured camera to decide its movements, an attacker can intercept and change the camera feed going to the robot.

Tampering

- The lack of integrity checks. Integrity checkers examine stored files or network packets to see if they have been altered or changed. For example, it was found that Universal robots have no integrity checks on UR+ artifacts when installed on the robot. This means an attacker can change an installation file maliciously and install it on the cobot.

- An attacker intercepts and changes/alters a message through the network, this can be especially easy when there are no integrity/signature checks on the messages.

- An attacker modifies or deletes robot data that is stored on the cloud.

- Attackers can use external remote services as an entry point in to the local network. External remote services allow a user to connect to an internal network resources remotely. Examples include VPN's, Citrix, proxies etc.

Repudiation

- No authentication to control for accessing a robot from within local network. For example, no authentication required to exert manual control over robot (UFactory). A network attacker can take advantage of this by spoofing his location (making the robot think he is in the LAN) and gain access to the robot.
- An attacker listens to a communication channel without authorisation. These types of attacks are usually done silently without causing suspicion, that way this can be done for a long time without anyone knowing.

Copyright © 2021 University of York

Information disclosure

- Attackers can obtain information remotely through a number of software vulnerabilities. For example, the Real-Time Data Exchange (RTDE) interface in universal robots allows unauthenticated reading of robot data and unauthenticated writing of registers and outputs.
- Unencrypted intellectual property can make an easy target for attackers, they can find ways to access it through the network. For example, it was found that Unprotected/unencrypted intellectual properties exist in universal robots controller cb 3.1.

Denial of service

- WiFi deauthentication attack. If the robot is connected via wifi an attacker can perform a WiFi de-authentication attack using a jammer device. This does require the device to be in close proximity.
- An attacker can flood the cobot with network packets to cause a denial of service attack. Researchers have found some controllers are susceptible to this type of attack (ABB, Phoenix Contact, Schneider Electric, Siemens, WAGO - Programmable Logic Controllers, multiple versions)
- Uncontrolled resource consumption can allow an attacker to cause a denial of service attack by using all the cobot resources.
- An attacker can cause a communication channel to become unusable. This can be achieved with the spamming of packets.
- An attacker could launch a MITM (man-in-the-middle) attack, for example in UR robot this could be done against MODBUS from a device with a spoofed IP address. This can lead to DOS, if the attacker intercepts the packets modifies them and forwards them to another component in the system.

Elevation of privilege

- Attackers can control a robot remotely through software vulnerabilities. For example, the UR dashboard server enables unauthenticated remote control of core robot functions.
- In some instances there exists no authentication to control robot from within network, for example no authentication required to exert manual control over a UFactory robot.
- Attackers can exploit an XSS like attack to get authenticated privileges remotely which can also lead to remote code execution.
- User enumeration attack. When a malicious actor uses brute-force techniques to guess or confirm valid users in a system, this is known as user enumeration. For example, a vulnerability was discovered in the Universal Robots Controllers' file system based in Debian, which is subject to CVE-2016-6210 and allows to perform user enumeration by leveraging a aw in the methodology of hashing on a static password when the username does not exist. It allows remote attackers to enumerate users by leveraging the timing difference between responses when a large password is provided.
- Remote Robot Control, an attacker might be able to gain control of the cobot through the network, for example the Universal Robots dashboard server enables unauthenticated remote control of core robot functions.

Copyright © 2021 University of York

- No lockout threshold, for example ABB IRC5 FTP daemon in VxWorks does not close the TCP after a number of failed log in attempts which allows brute force attempts.
- Hard-coded credentials. For example, Universal Robots Robot Controllers Version CB 3.1, SW Version 3.4.5-100 utilises hard-coded credentials that may allow an attacker to reset passwords for the controller.

References

- [1] V. M. Vilches, “Threat modeling a ros 2 robot,” Oct 2019. [Online]. Available: https://news.aliasrobotics.com/threat-modeling-a-ros-2-robot/
- [2] A. Shostack, Threat modeling: Designing for security. John Wiley & Sons, 2014.
- [3] “Ros 2 robotic systems threat model.” [Online]. Available: https://design.ros2.org/articles/ros2 threat model.html
- [4] D. Dolev and A. Yao, “On the security of public key protocols,” IEEE Transactions on information theory, vol. 29, no. 2, pp. 198{208, 1983.
- [5] L. Kohnfelder and P. Garg, “The threats to our products,” Microsoft Interface, Microsoft Corporation, vol. 33, 1999.

Copyright © 2021 University of York