The Capability-Based Command Protocol

CBCP Project Team:

Jørn Guldberg
jfg@quasios.com

Patrick Jakobsen
paj@quasios.com

Jakob Kjær-Kammersgaard
jkk@quasios.com

Jacopo Mauro
mauro@imada.sdu.dk

Daniel Merkle
daniel@imada.sdu.dk

Kasper Døring
kdo@universal-robots.com

1 Introduction

IT security is a problem for Industry 4.0 and smart manufacturing. Exposing production lines and IoT-devices to the public internet presents new possibilities for remote operations and monitoring, but also opens up the attack surface for hackers.

With Industry 4.0 the demand for remote operation, monitoring and making autonomous networks of robots and IoT-devices has increased. However, most robots are still isolated from the internet because they are vulnerable to cyberattacks and this prevents the movement to Industry 4.0. Isolating robots from the internet does not either mean that the robots are cyber secure. If an attacker gains access to the local network, the attacker can do serious harm, and propagate wide in the local network as there often is no security solution internally. A solution for Industry 4.0 is needed which both can secure the communication from the public internet, the local network and where it is possible to send secure autonomous commands.

To increase the IT security, in this paper we present the Capability-Based Command Protocol (CBCP), i.e., a software solution that can be used to define and secure network communications both internally and also across the public internet. The CBCP was developed as part of the homonymous project financed by the Danish Cybersecurity Hub and the Department of Mathematics and Computer Science at the University of Southern Denmark.

2 Background

One of the most used solutions to remote operate a device is done through a secure Virtual Private Network connection (VPN) and a remote desktop service like Virtual Network Computing (VNC). These solutions ensure security from the public internet, but when a user is logged on the system, the user can have or gain administration privileges and can potentially do harm. Remote access tools (e.g., Secure Shell) can be used to access a remote device and execute commands. However, it has the same drawbacks as the VPN/VNC solution with the possibility to gain administration privileges. In addition, automation using SSH keys can poses its challenges since it makes difficult to be performed at a large scale [1].

CBCP has been designed with these challenges in mind: a protocol where devices can send secure commands to each other, an operator can send secure commands into the network, and a formal way to manage permissions and encryption keys.

CBCP is a capability based protocol, where tailored permissions are required to execute commands on other machines. A capability is a non-forgable, cryptographic object that represents the permissions, which the server can validate to ensure that the capability indeed represents a given permission. Capabilities in CBCP are based on extended password capabilities [2], which provides a

complete toolset for working with fine grained permissions.

3 CBCP

CBCP is an application layer protocol as depicted in Figure 1. It is intended for well-defined networks where the identities and possible interactions of connected hosts are known. In CBCP, an interaction implies a client issuing a command to a server, which sends a response in return. The server only accepts the command if the client has a capability that grants access to the command in question. CBCP ensures confidentiality, integrity, and authenticity by using concepts from both public and symmetric key cryptography. CBCP addresses availability by limiting the amount of work being done for malformed or malicious packets.

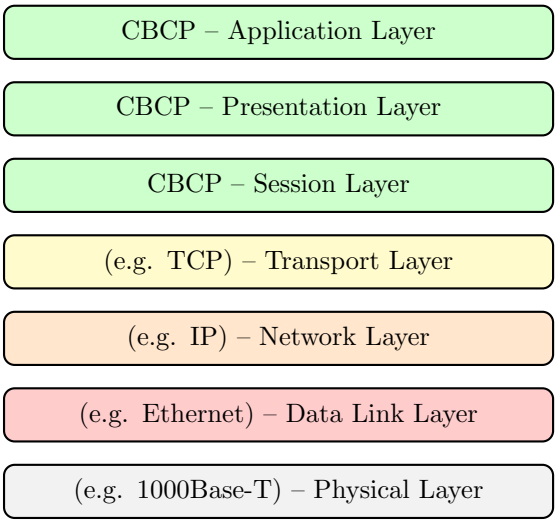

Figure 1: CBCP in the OSI Model.

CBCP is transport-layer agnostic and it is possible to use CBCP with any transport layer that provides reliable data transfer. If a transport layer without reliable data transfer is required, then CBCP may be used if a middle-layer that offers reliable data transfer on top of the transport layer is used. This supports usage across a wide range of existing systems.

The permission management system in CBCP is based on extended password capabilities[2] that encode a specific collection of rights. As an example, Figure 2 shows a robot controlled by two other network participants with separate responsibilities. In the figure, only the operator is allowed to start and stop the robot, and only the computer is allowed to change the task the robot is performing. The operator and computer have different, restrictive permissions. 'Start', 'Stop' are possible only for the operator while 'Change task' permissions are held by the the computer.

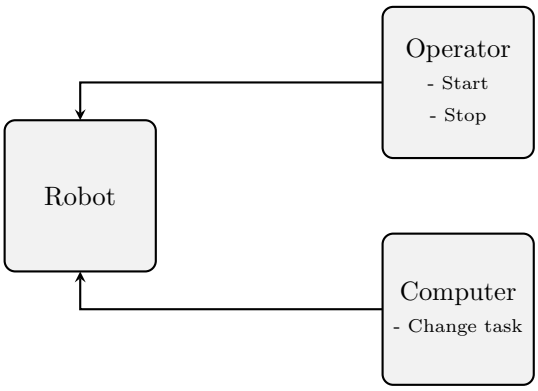

Figure 2: Permission in a simple Industry 4.0 scenario.

CBCP allows access rights to be defined with the finest granularity the application offers, which makes it possible to eliminate superfluous permissions that can be used in attacks.

Each host in the network holds a public/private key pair. The first step when a client host connects to a server host is for them to prove their identities to each other. They then establish a symmetric encryption key that is used to encrypt the communication between them. After the symmetric encryption key is established, the client can begin sending commands to the server. The symmetric key protects the entire communication: command number, interface, payload, etc. The only information that can be sniffed is an identifier for the client that the server requires to determine which party is sending the packet.

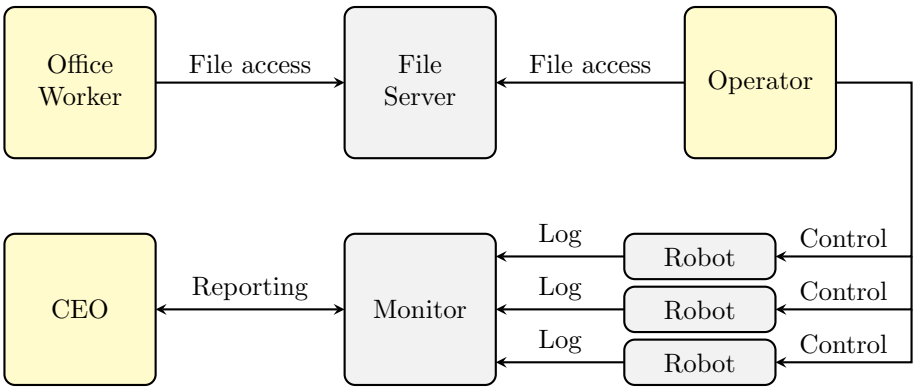

Figure 3: Example of network setup.

The deployment of applications using CBCP is done using a formal network description. This guarantees that the permitted communication in the network

is precisely the communication that is intended. The description can therefore be used for various security analyses, e.g. determining possible attack paths and high-risk components. As an example, Figure 3 visually represent a CBCP-network (we invite the interested readers to A for its textual representation). For this representation, it is possible to see how an attacker can move in the network. Assume for instance that Office Worker has been compromised and we want to know how the attacker can compromise the CEO. By following the arrows we can see that the attacker has to move through File Server, Operator, one of the Robots, the Monitor and finally reaches CEO. The move from the File Server to the Operator requires that the Operator sends a command to the File Server, which then must compromise the Operator through the response. Hopefully the network is defined to give the least privileges needed such that its gets harder for the attacker to compromise other hosts via the interfaces. As we know the attack path and possible commands, we can analyze which commands the attacker can use and thereby the risk.

Moreover if a host is compromised and we want to mitigate the attack, CBCP can be used to revoke the privileges. If for example Office Worker is compromised, we can revoke any capabilities Office Worker has, and it is no longer possible for Office Worker to send any command to its neighbours. To do so in the example, a revocation command is sent to the File Server, which revokes the capability by setting all of the permissions as invalid without disrupting the other parties. The compromised Office Worker-machine can be cleaned and rejoin the network when the capabilities is enabled again. If it is believed that the encryption keys has been leaked, it is possible to generate a complete new set of CBCP-Database files containing new encryption keys.

4 Technical Introduction

This section provides an overview of how CBCP works and how to use it in development and operations. For more details please consult the CBCP specification and other materials available at www.quasios.com.

4.1 How CBCP Works

CBCP uses two channels of communication. The first channel is the control channel, which is used to establish a connection and an associated symmetric encryption key. The connection is established using a handshake where the asymmetric encryption keys are used to encrypt the communication and for the hosts to prove their identities to each other. The other channel of communication is the command channel, which is used to communicate the commands on a request-response basis. Multiple connections can be open between a pair of hosts, which makes it easier to work with multi-threaded applications.

To execute a command at a server, the client must use a capability encoding the right to do so. When a server receives a request then it attempts to verify

the capability and executes the command only if it verifies correctly. If the verification fails, then the incident can be logged and/or reported and the request ignored.

Remote commands executed over CBCP can be considered remote procedure calls. The CBCP remote commands and their responses operate on "raw" buffers of bytes, which offers the greatest flexibility. It is left to the application programmer and middleware to encode and decode the contents of these buffers if/as required.

4.2 Usage in Development

CBCP works with cryptography and it is therefore recommended to use a library implementation of CBCP.

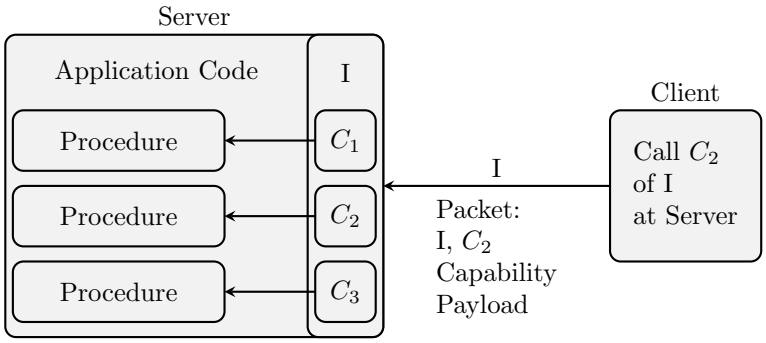

Figure 4: A CBCP server and client communication.

A CBCP communication involves a client and a server and a process can be a client and a server at the same time.

For a remote command to be executed through CBCP some initial work must be completed. Firstly, the server must bind an implementation procedure to each command that it provides. Secondly, for command requests to be communicated, the client must open a CBCP connection to the server, which is done over the control port. Once these steps have been completed any number of command requests can be sent from the client to the server, for which the server will execute the bound procedures for the requests that it verifies successfully.

Figure 4 illustrates a client sending a command to a server. In the illustrated setup the server has a single interface with three commands, each of which invoke a different procedure upon successful verification. The client needs to send its request and receive the response. The server has upon program startup bound procedures to each of the commands it offers. If the capability sent by the client can be verified by the server and encodes C_2, then the bound procedure will be executed.

The procedures bound to the CBCP commands are passed the data in the payload, similarly to reading from a raw TCP socket. It is therefore important when using CBCP without additional abstraction to be careful when implementing the bound procedures. Mistakes in the implementation of bound procedures can result in security exploits by malware infected clients.

4.3 Usage in Operations

A network of machines communicating over CBCP requires that all participants hold cryptographic information. This includes all machines used by operators to access the network. To send commands for management purposes, the operator sending the commands must hold the required capabilities. Once the operator holds the capabilities the application can offer the same user experience as a non-CBCP application.

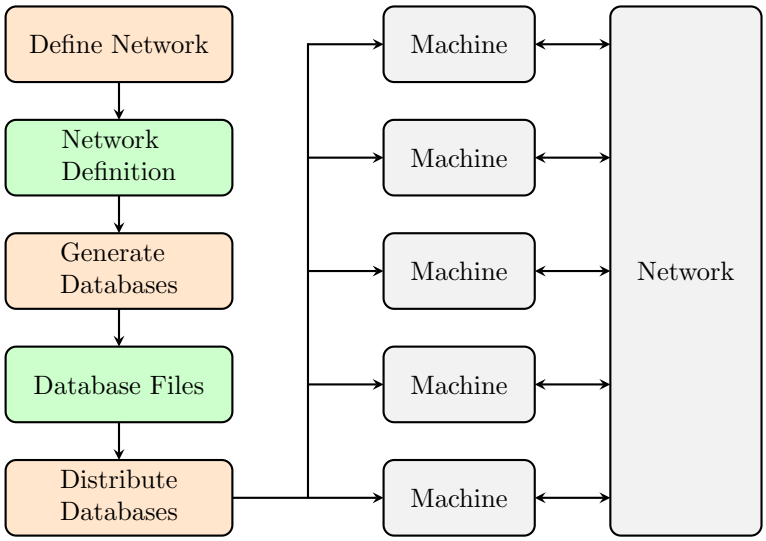

Figure 5: Workflow to use CBCP.

The intended workflow is illustrated in Figure 5. The means of defining the network that is currently provided is using a textual definition language, but defining the network can be done using any method and the network definition can be any format that can be used to generate the databases.

If the network changes then the network definition must be updated and the databases re-generated. The new database files then need to be distributed to the machines as a part of the new deployment. To limit the scope of updates, the re-generation can be reduced to a difference from the old network. Then only the affected machines need to be updated with the database diffs.

The distribution of database file updates may be done in any way that respects the threat model and its automation is left as future work.

5 Example of Use Cases

5.1 Construction Sites

A construction site has large quantities of materials that are moved around periodically. The logistics of moving materials around to their usage sites can be handled by using mobile robotics.

Assume for the purposes of this case story the involvement of three major components: a worker requesting bricks, a mobile robot fetching the bricks, and a robot arm loading the mobile robot with bricks. The worker simply requests the bricks and the mobile robot then fetches the bricks. The mobile robot moves over to the robot arm and coordinates the loading of bricks, after which the mobile robot returns to the worker that made the request.

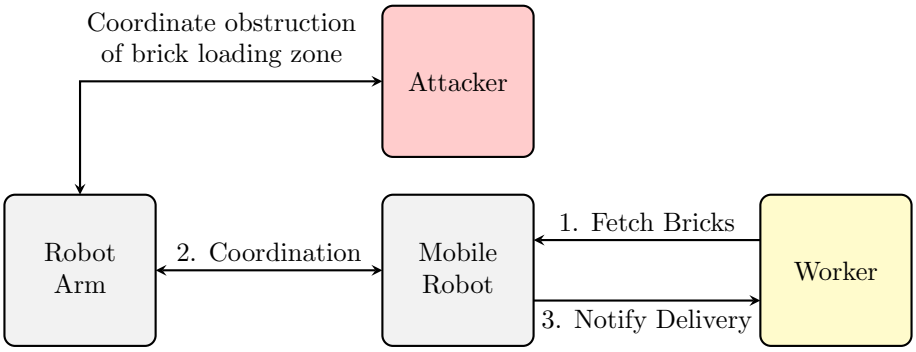

Figure 6: Construction Sites network.

A current-day solution would be to operate all of the involved machines and programs in a closed wifi network under the assumption that the network is safe. This assumption is however hazardous. It is possible that an attacker may gain direct or indirect access to the network, notably because the attacker can physically access the machines. When the attacker has gained access to the vulnerable network, it is then possible to execute commands on the construction site machines, which may endanger the workers.

To secure the construction site the ramifications of a compromised machine must be restricted. The use of CBCP allows the implementation of this restriction. The communication between the program used by the worker and the mobile robot can be secured to prevent attackers from using it. The coordination can

also be meaningfully secured with CBCP, as this can prevent an attacker from controlling the robot arm.

CBCP imposes new requirements for the attacker to successfully execute a specific attack: obtaining control over a machine on the network with the capability to perform the command of interest. This restriction imposes new challenges on the attacker and presents more possibilities for attack failure.

5.2 Universal Robots Dashboard Server

Universal Robots' collaborative robot arms have an optional feature called "Dashboard Server". The Dashboard Server is a robot subsystem that allows a user on the network to send a limited variety of commands to the robot (e.g. to start it or change the robot program it is running). The protocol it uses is however not built with IT-security in mind, as it is a plaintext protocol that is neither encrypted or authenticated.

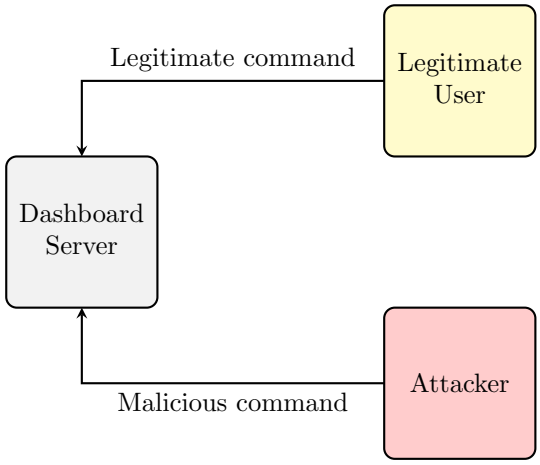

Figure 7: The UR Dashboard Server network.

By encapsulating the communication with CBCP it is possible to create a "DashboardShield" to protect the Dashboard Server from being abused by attackers. CBCP protects the communication itself and additionally provides new features on top of the Dashboard Server, as it allows the administrator fine-grained control over what users are allowed to send the various commands. DashboardShield relays legitimate command requests to the Dashboard Server and discards illegitimate requests.

CBCP can thus be used to hide away insecure systems through encapsulation methods.

6 Conclusion

CBCP is a powerful and flexible addition to the cyber security toolbox. The capabilities provide much more fine-grained permission control than currently widespread solutions. Additionally, the formal CBCP network definition provides an additional tool for security risk assessment.

CBCP shows great promise for use in real-world applications, such as securing the robot infrastructure at construction sites and, it has already been tested as a security layer for the Universal Robots Dashboard Server.

7 References

[1] ssh.com. *Insight from real customer cases.* `https://www.ssh.com/academy/iam/ssh-key-management#insight-from-real-customer-cases`. Accessed: 2022-02-07.

[2] Lanfranco Lopriore. "Access right management by extended password capabilities". In: *International Journal of Information Security* 17.5 (2018), pp. 603–612.

A Appendix A

```
!CBCP 1.0

!HOSTS
Office_Worker ; TCP,1.1.1.1:9000
File_Server   ; TCP,1.1.1.2:9000
Operator      ; TCP,1.1.1.3:9000
Robot1        ; TCP,1.1.1.4:9000
Robot2        ; TCP,1.1.1.5:9000
Robot3        ; TCP,1.1.1.6:9000
Monitor       ; TCP,1.1.1.7:9000
CEO           ; TCP,1.1.1.8:9000

!GROUPS
@Robot        ; Robot1, Robot2, Robot3

!INTERFACES
File_Access   ; read, write, execute
Control       ; start, stop
Log           ; send log
Reporting     ; send alert, request report

!IMPLEMENTS
File_Server   ; File_Access
@Robot        ; Control
Monitor       ; Reporting, Log
CEO           ; Reporting

!CAPABILITIES
Office_Worker ; File_Server ; File_Access ; read, write, execute
Operator      ; File_Server ; File_Access ; read, write
Operator      ; @Robot      ; Control     ; start, stop
@Robot        ; Monitor     ; Log         ; send log
Monitor       ; CEO         ; Reporting   ; send alert
CEO           ; Monitor     ; Reporting   ; request report
```

Listing 1: CBCP config for security risk assessment example Server