Remote Air-Gap Live Forensics

Tom Van der Mussele(✉), Babak Habibnia, and Pavel Gladyshev

DFIRe Lab, School of Computer Science, University College Dublin, Dublin, Ireland
tom.van-der-mussele@ucdconnect.ie, tomvandermussele@gmail.com
{babak.habibnia, pavel.gladyshev}@ucd.ie

Abstract. This paper describes a solution to build a scalable means to perform remote live forensics, which introduces minimal and traceable changes to the air-gap systems. The solution can respect the air-gap and not introduce network connectivity to the air-gap systems. It provided a central management system with the solution; this allows the solution to be used in an incident across multiple systems. Full traceable actions, built in the solution, allow the investigator to respect the second ACPO rule during the live forensics. The solution introduces low impact changes to aim for maximum stability and preservation of evidence during the investigation of the air-gap system. The solution needs to be operational with minimal interaction behind the keyboard. In this paper, it will compare and benchmark other industry solutions with proposed solution in this research.

Keywords: Digital forensics · Live forensics · Air-gap · Remote forensics · Forensic dongle

1 Introduction

Today, malicious software for air-gap systems is becoming more common. To investigate the active threat as part of the incident response, it is well accepted to perform live forensics. Due to the isolation of these air-gap systems from the normal network connected systems, it is not always straightforward to perform live forensics. Any remote live forensics is difficult to perform without introducing large changes to the infrastructure.

In the case of a confirmed incident, speed of analysis is a contributing factor for success. With air-gap systems, it can be a challenge to bring the skills behind the keyboard when there is no network connectivity. Making systems "air-gapped" is a well-known defence against attackers for high-value target or critical systems such as SCADA (Supervisory Control and Data Acquisition), ICS (Industrial Control Systems) and HMI (Human Machine Interface) of PLC (Programmable Logic Controller). The attacks on air-gap systems are becoming more common, often driven by strategic interest such as cyber warfare. Attacks on air-gapped systems such as Stuxnet [1] and

S. Goel et al. (Eds.): ICDF2C 2020, LNICST 351, pp. 182–203, 2021.
https://doi.org/10.1007/978-3-030-68734-2_10

Flame demonstrate the need for forensic capabilities in the incident response phase of an attack [2].

The main fundamental forensic investigation and incident response principles apply for "air-gapped" controllers as much as "non air-gapped" or connected controllers. However, as air-gapped systems are not connected to the enterprise network or common networks such as the internet, a remote and instant incident response is a challenge. This research will try to solve this by introducing a combination of hardware and software that simulates low impact devices which can be used as a bridge to an 802.11 network. The solution needs to be able to communicate remotely with the system, have a small footprint on the system, be scalable and able to perform live forensics and not expose the system to a network.

2 Scenario

2.1 Problem

In 2010, Stuxnet was found to be distributed and infecting air-gapped systems of a nuclear plant to target the PLCs operating the power plant and potentially damage the plant. Since the system was air-gapped, the infection took place using a USB thumb drive. Described in the analysis by Farwell and Rohozinski (2011) [3], the PLCs were operated by engineers onsite using Siemens software, this Siemens Human Interface or HMI was installed on a standard Windows CC operating system. The engineers, operating the HMI, are not experts in live forensics and only use the Siemens software to provide instructions to the PLCs, not the operating system. In case of an incident, providing field engineers with instructions on how to investigate the system live is a risk. In a typical incident scenario, the forensic investigator needs to travel physically to the site to have the option to investigate the air-gap system. Travel time is a big disadvantage in incident response, the state of the system might change a lot, or an attacker might disappear during that travel time. Often air-gap systems are used in hard to reach places such as oil rigs, nuclear plants, military equipment, kiosks, … and therefore travel time can be long and travel costs can be expensive. This only works against the success rate of the incident response.

2.2 Importance of Remote Live Air-Gap Forensics

Often SCADA use historian servers for storing data and have Human Management Interface (HMI) to display data and controllers to interact with the control systems. Many of these are built on traditional operating systems with special control software to interact with the PCLs [4]. Van der Knijf, in 2014 [5], classified engineering stations (controllers), databases and historians with a priority 3 in value of possible sources of forensic artifacts. The priority was derived by applying the algorithm priority = (2*Likely value) − (effort) + (4*volatility). This makes it a moderate source of value within the investigation strategy [5].

3 Approach

3.1 Common Approach and Its Risks

One might think, simply connecting the air-gap system to a network and use a remote session to perform the forensics real-time is a straightforward approach. This could allow the investigator to connect, authenticate and start the investigation of the HMI. The HMI operating systems often have a longer lifespan and operate critical processes [4, 6]. The introduction of a new network adapter or connection can be risky; a high impact driver change might destabilize the system. Additionally, a network connection might tip off the attacker and even allow for further infection beyond the air-gap system.

To perform live forensic acquisitions on systems, there are two options available. The first one requires the pre-installation of an agent, capable of performing tasks that can extract the data over the network [2]. This is an option frequently used in enterprise networks but can be a challenge in air-gap systems due to the missing connectivity. These agents, usually part of an Endpoint Detection and Response solution are installed on the systems and will connect back to a central server deployed in the cloud or on-premise. The post-installation of an agent on the system can heavily increase the footprint on the air-gap system and requires additional items, such as network connectivity to an on-premise controller or cloud controller. Malware might be monitoring for the installation of agents, after its successful infection, and alter its behavior to avoid detection. A second method is using a modified agent or a tool which are commonly used in incident response to extract or find specific signs of malicious activity. These can be native built-in commands or 3rd party tools such as Mandiant Redline [7].

3.2 Proposed Approach

The goal of this research is to avoid or reduce the risks described in the common approach section. Meanwhile, it wants to respect the air-gap and not introduce any typical network connectivity, by for example connecting an IEEE 802.11 adapter directly to the air-gap system. Instead, it would use a combination of cheap, readily available hardware and custom software which make up a “dongle”. To “operate” the air-gap system it would suggest that on one end, the investigator is connected to an IEEE 802.11 network and communicates over TCP/IP with the dongle.

The other end of the dongle is connected via USB to the air-gap system and simulates a keyboard, COM port and two removable drives. This allows a small footprint on the system, connect the air-gap system to the investigator over unlimited distance and should not destabilize the system. The aim is to send keystrokes via a simulated keyboard to the system which would be a substitute of the typing of an investigator behind the keyboard and use a low impact COM port for communicating with a custom written component running on the air-gap system (Fig. 1).

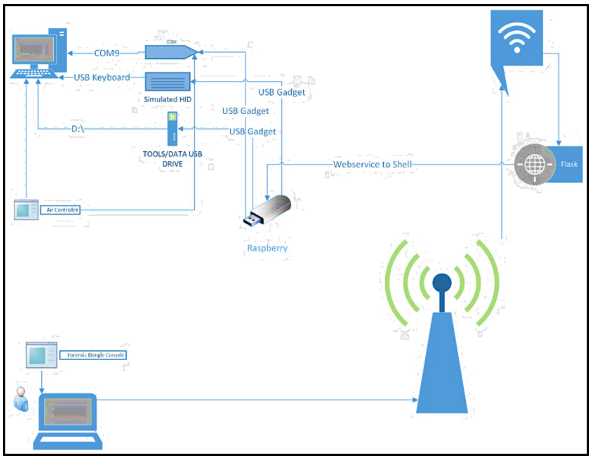

Fig. 1. Overview

The solution needs to be scalable so that multiple systems can be investigated at the same time. This would require some form of the central management system to communicate and interact with the dongles connected to each separate air-gap system. Live forensics does, in contrast with offline forensics, change the state of the system. To respect ACPO rule 2 [8], investigators must demonstrate what and for what reasons changes have been made to the system. The device is simulated on the air-gap system, it allows for setting unique identifiers (i.e.: USB Vendor ID, Device ID) and differentiating itself from other manufacture devices.

Using audit trails and timestamping of the commands issued, the aim is to provide the investigator with enough means to prove the changes and that he/she can demonstrate the impact on the system. The solution will be able to link every executed action on the air-gap system with the original command issued by the investigator. An independent review should be able to prove what commands were executed by the investigator, in light of the investigation. The proposal is to use unique tracking IDs for every executed command, synchronization of time sources using Network Time Protocol (NTP) and sufficient logging.

The final goal for this study is to come up with a solution (named *Forensic Dongle*) that,

- can perform live forensics on air-gap systems
- has a small footprint on the systems
- is scalable
- can respect ACPO rule 2
- is cheap

The research selected and researched tools which attempt to address the same problem of remote live forensics (on air-gap systems). It was the intention to select tools which fall in a similar cost range. The solutions that match closest with the "*Forensic Dongle*", are "Google Rapid Response" (GRR)[1], "Mozilla Investigator" (MIG)[2] and "Facebook osquery" (osquery)[3].

For each of goals, a benchmark or comparison with the selected solutions capable of remote live forensics was performed.

"Facebook osquery" is by default a local solution which makes remote live (real-time) forensics not possible. "Facebook osquery", however, comes with plugins which allow logging to a remote log source (i.e.: syslog). When choosing this option, the tool will query the system using a schedule and send the system data to the remote log source. This makes it, that the last option is not real-time or live. To separately compare these configurations, it will name the configuration with a local logging "osquery A" and the configuration with remote logging "osquery B".

3.3 Overview

The solution exists of 3 major logic components (Fig. 2).

The physical dongle ("*Forensic Dongle*"), which is connected to the air-gap system by a field engineer. The dongles can be used on a regular connected machine as well. The dongle will be able to execute tasks/scripts/commands on the connected host. To operate the dongles in a scalable way, they need to communicate with the "*Forensic Dongle Console*". This VB.NET[4] program serves as a master polling console which allows to task the dongles with different commands in real-time. This central management system also allows adding/remove or edit dongle configurations of expected dongles.

The "*Forensic Dongle Console*" keeps track of the deployed dongles and allows the investigator to submit tasks on multiple hosts involved in the incident in real-time. The third component is the "*Air Controller*", this component is the execution piece on the air-gap system. The program captures the commands from the simulated devices and is responsible for decoding & executing the commands and returning the output over the simulated devices. The program is written in Visual Basic 6.0[5] for maximum compatibility with legacy systems and systems lacking the latest updates and features.

[1] https://storage.googleapis.com/releases.grr-response.com/grr-server_3.3.0-4_amd64.deb.

[2] https://github.com/mozilla/mig/archive/20170308-0.e7a93ea.dev.zip.

[3] https://github.com/osquery/osquery/releases/.

[4] https://docs.microsoft.com/en-us/dotnet/visual-basic/.

[5] https://en.wikipedia.org/wiki/Visual_Basic.

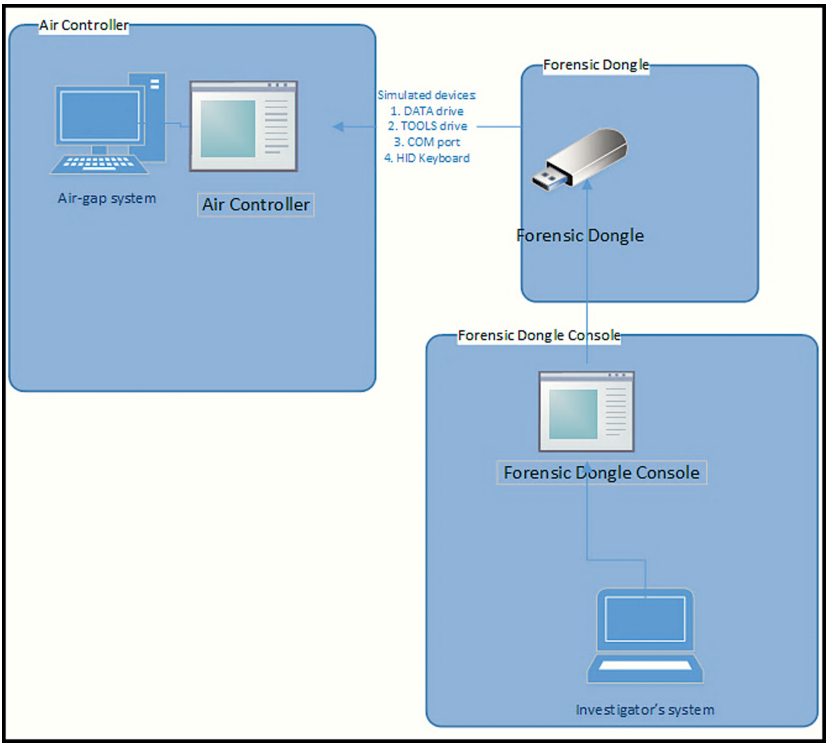

Fig. 2. Major components

4 Goals

4.1 Can it perform remote live forensics on an air-gap system?

There are 3 main aspects to this goal:

- Remote; an investigator must have the ability to be in a different physical location and operate the technology.
- Live forensics; the process of performing analysis and preserving evidence on a running system. This is performed real-time.
- Air-gap; the system has no direct network connection to the investigator or other systems.

Remote. The Raspberry Pi Zero[6] W is a low-cost piece of hardware with limited resources and capable of booting Raspbian OS[7], a Linux variant. There are included two features that will apply of them. One, the Raspberry Pi Zero W comes with an

[6] https://www.raspberrypi.org/products/raspberry-pi-zero-w/.

[7] https://www.raspberrypi.org/downloads/raspberry-pi-desktop/.

integrated IEEE 802.11n adapter. That will use this network adapter as the management interface of the "*Forensic Dongle*".

The second feature of interest is that the Raspberry can be used as a USB device. The goal will be, to create a controlled "bridge" between the air-gap system and the investigator. The 802.11 interfaces, embedded in the Raspberry Pi Zero W, is not hosted on the operating system of the air-gap system and therefore not interfering the air-gap. The dongle simulates low impact devices such as USB keyboard, USB thumb drives and a COM port. All these devices will be used to operate the air-gap system, while only the COM port is used for real-time bi-directional communication.

The investigator operates the "*Forensic Dongle Console*", a central management tool which can detect the dongles and issue commands to individual dongles (Fig. 3).

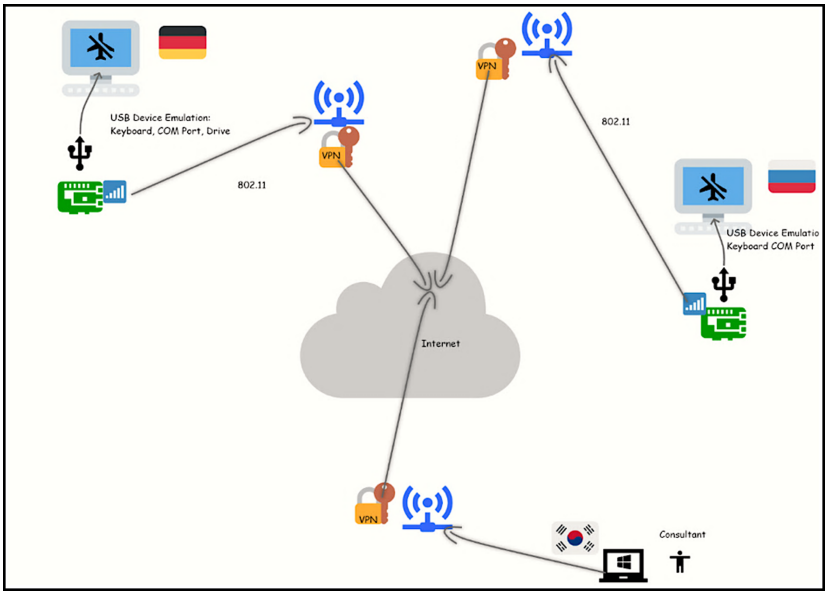

Fig. 3. Remote investigation scenario

For the investigator to address each dongle individually, each dongle needs to be unique. The hostname as a unique identifier for a dongle was chosen. An investigator can change the hostname using the "*Forensic Dongle Console*" or via quick editing a plain-text configuration file. The "*Forensic Dongle Console*" can, by using mDNS, detect all the dongles in the field and does not need to know any IP addresses assigned to the dongle, as they might be assigned by a DHCP server of for example the Wi-Fi access point. The solution deliberately uses mDNS. A field engineer, who is no expert, can just plug in a guest Access Point and connect the dongle. The investigator does not need to send a pre-configured image with fixed IPs in it and this allows him/her the maximum flexibility and speed. Once the "*Forensic Dongle Console*" can "see" the dongle, the investigator can issue an encoded command to the Flask Web service of the

dongle. The Web service will pass the command over to the air-gap system via the simulated hardware. A simulation of 4 devices will happen on the air-gap system.

1. A read-only USB Drive containing all the investigator's tools. (i.e.: Microsoft Sysinternals) (TOOLS drive)
2. A writable USB Drive which will be used by the "*Air Controller*" to preserve all captured evidence. (DATA drive)
3. A HID keyboard which will be used to send keystrokes to the system. (i.e.: simulating the investigator behind the keyboard)
4. A COM port which will be used for receiving commands and sending the output back to the investigator.

The commands will be passed through the COM port or the keyboard to the air-gap system. The "*Air Controller*" captures the encoded command and decodes the exact payload. The "*Air Controller*" is then responsible for the execution of the command, capturing of evidence and returning data through the COM port (Fig. 4).

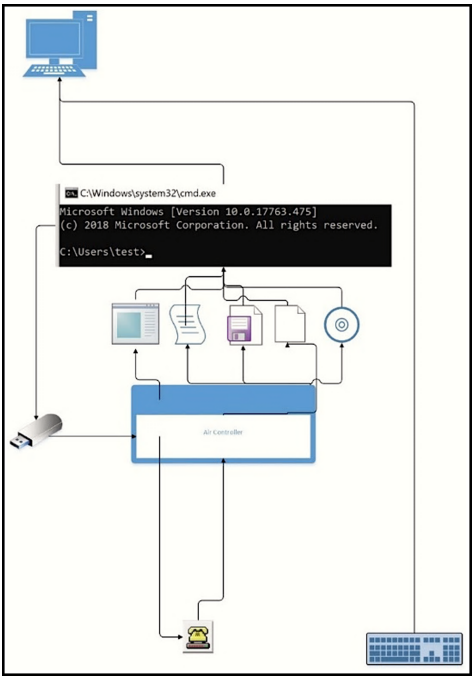

Fig. 4. Hardware simulation flow

When the "*Air Controller*" sends back data, a service running on the dongle called "*Forwarder*" hosts a TCP listener where the "*Forensic Dongle Console*" can retrieve that data from. When comparing the provided solution in this research with the other solutions, it was observed that all, except "osquery A", require a central management

system or remote logging server (i.e.: syslog). The IP connections towards these central management solutions must originate from the air-gap system. This will defeat the air gap. The overview can be found in Table 1.

Live Forensics. Live forensics is the capturing and preserving of evidence on systems while they are running. In contrast with dead box or offline forensics, the data which is of interest could be volatile. This means that the data is only available while the system is running, turning off the system might render the evidence useless. The magnitude and type of incident might favour for live forensics, it can be too labor-intensive to perform dead box forensics over multiple systems. As shown in Fig. 1, the "*Forensic Dongle Console*" communicates in real-time with the Flask Web service of each dongle. By using the "*Forensic Dongle Console*" (Fig. 5), the investigator can send commands in real-time to the dongles. The command gets encoded and packaged with a UUID and the "*Forensic Dongle Console*" will send it to the Web service of that specific dongle.

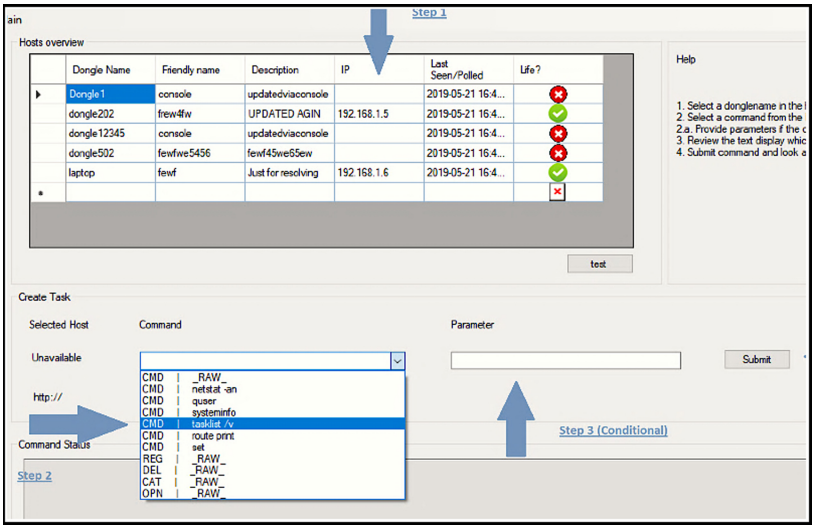

Fig. 5. "*Forensic Dongle Console*" – Issuing a command

In real-time, the "*Air Controller*", running on the air-gap system, intercepts the input of the COM port and executes the instructions received. Two methods of capturing evidence are available. The "*Air Controller*" will always write the details or pieces of evidence on the simulated DATA drive. For each execution, two files will be written, the job details and the job output (evidence). This makes that all details will be easily available for the investigator to analyze. The second method of capturing data is the "*Air Controller*" sending real-time output through the simulated COM port.

A service, called "*Forwarder*", that runs on the dongle, hosts a TCP listener. All data received by the COM port will be streamed to the TCP listener. The output is displayed to the investigator (Fig. 6) through the "*Forensic Dongle Console*", the same tool which is used to issue the commands. This output can only be text, binary captures will not be displayed and only written to the DATA drive for later investigation.

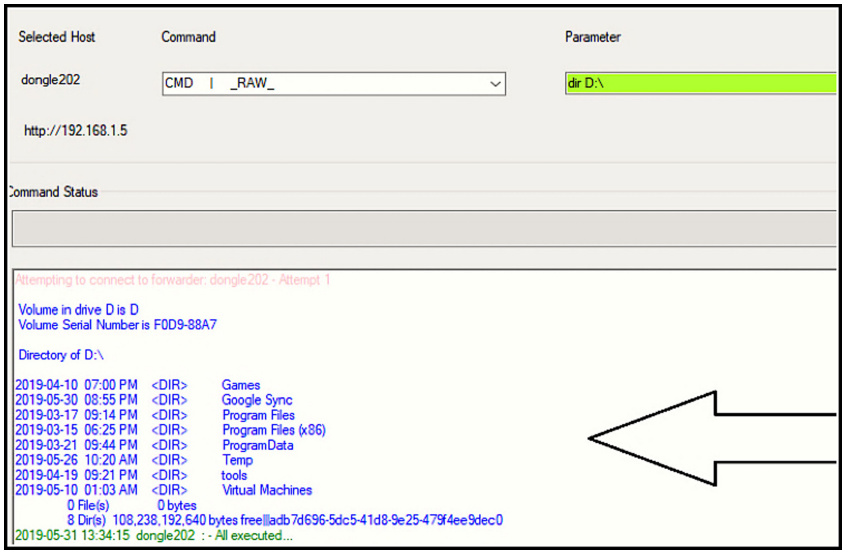

Fig. 6. "*Forensic Dongle Console*" – Real-time output

Air-gap. Air-gap systems are deliberately physically separate and have no connectivity (i.e.: TCP/IP, IPX, …) to other systems or the internet. Since this approach has been chosen for a security reason, it is important to respect this. To maintain the air-gap aspect of the system and yet being able to operate, choosing for simulating low-impact devices on the air-gap system. The simulated hardware, which is not capable of working as a network controller, is operated in a controlled environment by the investigator.

The air-gap system does not receive any network capability but instead for any two-way communication; a COM port is simulated. By putting this COM port in a controlled environment, from the "*Air Controller*" to the "*Forwarder*", this respects the air-gap. The simulation would probably not tip off an attacker and further infection of other systems remains impossible.

GRR and MIG require a TCP/IP connectivity from the air-gap system to the central controller and is thus not air-gap. Osquery A is a local solution and needs to be operated locally, so the air-gap is maintained, however, the investigator needs to be physically behind the system and there does not meet the first goal as mentioned above. Osquery B requires an IP connection to use alternative logging tested using syslog, and therefore is not air-gapped. Additionally, osquery B requires scheduling of sending the data to the syslog server, as a real-time and continuous is impossible due to the amount of data.

Goal Summary. By researching and evaluating all the different solutions, this research was able to conclude that the "*Forensic Dongle*" is the only solution that fully meets the goal.

Table 1. Goal 1 – Can it perform remote live forensics on air-gap systems?

	Forensic Dongle	GRR	MIG	"osquery A"	"osquery B"
Remote	Yes	Yes	Yes	No	Yes
Live forensics	Yes	Yes	Yes	Yes	No
Air-gap	Yes	No	No	Yes	No

4.2 Is It Scalable?

A coordinated attack might involve multiple (air-gap) systems. In a connected environment, an attacker might be infecting multiple hosts and pivoting through the environment. An attack on multiple air-gap systems would be advanced and requires a good bit of planning but is feasible. In the case of cyber warfare, attacking multiple power plants at the same time would be of significant advantage for the country that is attacking. The enemy's defence will be weakened significantly due to the lack of regular power sources. The goal is to build a flexible solution that allowed an investigator to analyze multiple systems, in real-time and with minimal configuration effort. Every dongle deployed in the field should be similar for the investigator.

When using multiple dongles, it needs to make sure that every dongle is unique, so it is possible to task each dongle with individual commands. The aim for a very simple setup is using the hostname as a unique identifier. When adding a dongle to the "*Forensic Dongle Console*", 3 parameters are needed (Fig. 7).

- A unique hostname – a unique name which could have no meaning to the investigator (i.e.: dongle + incremental number) (Fig. 7).
- A friendly description – A meaningful name for the investigator to provide transparency over the dongles (Fig. 7).
- A comment or note – A comment to enhance the context of the dongle (Fig. 7).

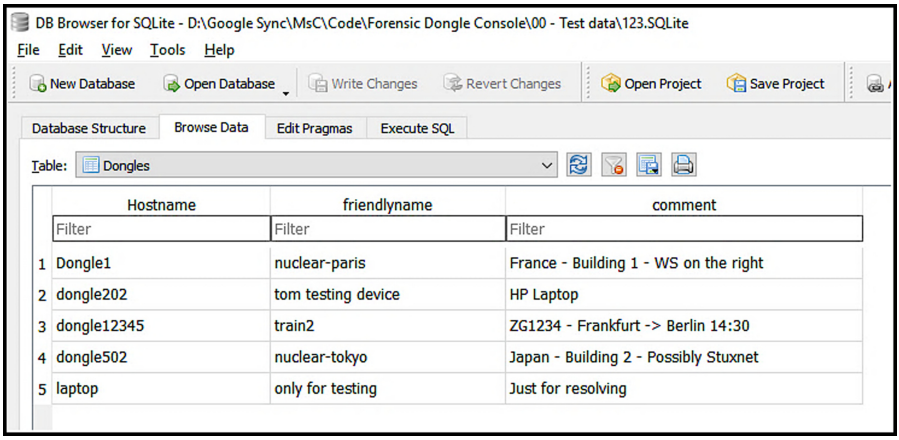

Fig. 7. "*Forensic Dongle Console*" – Dongle list.

Using the hostname and the friendly name, the investigator can create a context of which dongle is placed where. By using the hostname as a unique identifier, the investigator could use mDNS to "discover" the dongles, the investigator does not need to know the IP address assigned to the dongle. This allows the field engineer to just "plug in" the dongle with nothing to configure. The goal here was to make it as easy as possible for the field engineer, all he/she must do, tell what system he/she connects it to. The field engineer should have no ability to configure an address for the dongle or any access to the operating system of the dongle.

The solution also chooses an easy configuration for the investigator. The investigator can prepare dongle (images) upfront by either using the "*Forensic Console Dongle*" or editing a single plain-text configuration file. The image can easily be hosted and has no IP address and only a generic name. The dongle is now ready to be deployed. Once the dongle is plugged in, the "*Forensic Dongle Console*" can discover the dongle and perform a mapping from generic name to friendly name and IP address. This gives maximum flexibility for the investigator.

GRR packages binaries for deployment. The package can be installed on multiple hosts, once the hosts are reporting back to the GRR server, the investigator can use "hunts" to perform investigations on multiple systems, that can be concluded that GRR is scalable.

Mozilla Investigator can be installed on multiple hosts and offers the possibility to prepare tasks, these tasks are distributed to the different agents in batch fashion. It is concluded that MIG is scalable.

Facebook osquery B, using a remote logging mechanism, is scalable and sends data from multiple hosts to the log server. On a scheduled basis, data is collected and sent to a remote logging mechanism. The results of the remote logging can be analyzed using a log analysis tool. It is concluded it is scalable. In contrast with osquery A, where the data is kept locally, and queries must be performed behind the keyboard.

The conclusion is that osquery A is not scalable.

Table 2. Goal 2 – Is it scalable?

	Forensic Dongle	GRR	Mozilla Investigator	"osquery A"	"osquery B"
Scalable	Yes	Yes	Yes	No	Yes

4.3 Has It a Small Footprint?

The goal for having a small footprint is to minimize impact and therefore try and avoid system instability. Additionally, the larger the footprint or changes introduced to the system, the bigger the chance that valuable volatile data will be lost.

GRR, MIG and osquery require the installation of an agent, this makes it that the solutions are less suitable during the incident when the agents are not pre-installed. The "*Forensic Dongle*" is the sole solution which does not require a pre-installed agent and thus can easily be used in an unprepared or uncontrolled environment.

To measure the footprint change, the research focused on areas which are typically affected when changes are introduced due to insertion of hardware, persistence keeping, or file system changes.

- Driver installation
  The "*Forensic Dongle*" solution simulates hardware to achieve the first goal (Remote-Live Forensic on air-gap systems). On a Windows 7 system, an extra driver needs to be installed for the COM port and mass storage simulation to work simultaneously.
- Simulated hardware
  The "*Forensic Dongle*" solution simulates hardware. This area will measure the number of simulated hardware devices that are initialized by Windows.
- Windows Services
  Windows Services is a mechanism to have Windows automatically start programs and DLLs. This mechanism is often used to achieve persistence. These services are started by the Service Control Manager and can be run under a different user context, including high privileged ones, such as SYSTEM.
- Registry Changes
  The number of unique changes is introduced, changed or deleted. These changes can be made for persistence, configuration settings, hardware functionality. The requirement for the change count was that it was directly linked to the technology, not inherited effects of the Operating System. i.e.: MRU…
- Persistent file changes/ additions (Windows, AppData, Program Files, ProgamData, Program Files (x86))
  Persistent file changes which are needed for the functionality of the solution, this could be installation files, DLLs, configuration files, … Inherited file changes, such as prefetch items are not included in this count.

Table 3. Goal 3 – Has it a small footprint?

	Forensic Dongle		Google Rapid Response		Mozilla Investigator		"osquery"	
	Win7	Win10	Win7	Win10	Win7	Win10	Win7	Win10
Driver installation	1	0	0	0	0	0	0	0
Simulated hardware	4	4	0	0	0	0	0	0
Windows Services	0	0	1	1	1	1	1	1
Registry Changes	46	46	17	17	10	10	0	0
File Changes	1	0	75	75	9	9	20	20

Comparing the impact of the solutions, it shows that GRR, MIG and osquery change the filesystem heavily and are persistent. Even though the "*Forensic Dongle*" generates the most Registry entries, many of them are volatile. And many of them are traceable as shown in 4.4.

The devices simulated to the air-gap system are of low impact, most operating systems have built-in drivers and modules to support these devices.

4.4 Is It Able to Respect ACPO Rule 2?

"In circumstances where a person finds it necessary to access original data, that person must be competent to do so and be able to give evidence explaining the relevance and the implications of their actions" [8].

An investigator needs to be able to demonstrate that actions performed are necessary for the investigation. One necessity is that the investigator can demonstrate the changes he/she made from the normal operational computer activity and/or attacker activity. Therefore, any activity needs to be documented, have an audit trial and timestamps to correlate it back with changes on the filesystem. Within the "*Forensic Dongle*" solution, in this research, it was able to create 3 features that allow an investigator to respect ACPO rule 2 [8]. That incorporated "Unique identification of issued commands", "Timestamps on issued commands" and log capacity on both Central Management side as client-side.

Unique Identification of Issued Commands. To have a full trace of an investigator preparing a command and have it executed on the air-gap system.

The investigator must be able to correlate the output with the original command, so it requires a unique identifier with sufficient randomness. That used Microsoft's implementation of RFC 4122[8] (Fig. 8).

[8] https://tools.ietf.org/html/rfc4122.

```
Public Function GenerateID() As String
    GenerateID = System.Guid.NewGuid.ToString()
End Function
```

Fig. 8. "Forensics Dongle Console" – GenerateID().

When issuing a command to a dongle using the "*Forensic Dongle Console*", a UUID (Universally unique identifier) gets generate using GenerateID(). Then prepare command and encode the UUID in the command (Fig. 9).

```
tempUUID = GenerateID()
tempType = Trim(stuk(0))
tempPayload = Trim(stuk(1))

payload = tempUUID & "|" & tempType & "|" & tempPayload
encodedpayload = ConvertBase64(payload)
Log2File(timestamp() & " - " & tempIP & ":3000/payload?ID=" & encodedpayload & " - " & payload, LogFile)
```

Fig. 9. "*Forensic Dongle Console*" – Encoding UUID in command.

The encoded command is passed onto the Flask Web service and without change sent to the COM port. The "*Air Controller*" will then capture this and decode the instructions, containing the UUID (Fig. 10).

```
unction RawSplit(ByVal RAW As String, ByRef refUUID As String, ByRef refCommand As String, ByRef refpayload As String) As Boolean
im argNo As Integer
im midden As String
rgNo = 1
or I = 1 To Len(RAW)
    midden = Mid(RAW, I, 1)
    If midden <> "|" Then
        If argNo = 1 Then
            'UUID
            refUUID = refUUID & midden

        ElseIf argNo = 2 Then
            'COMMAND
            refCommand = refCommand & midden
        Else
            'payload
            refpayload = refpayload & midden
        End If
    Else
        argNo = argNo + 1
    End If

ext I
D = refUUID
awSplit = True
nd Function
```

Fig. 10. "*Air Controller*" – Parsing UUID RawSplit().

After execution of the parsed instruction, the "*Air Controller*" encodes the output including the UUID and sends it to the investigator (Fig. 11).

```
'Dump Passwords'
If InStr(1, PA, "Dump Password") Then
    Call CMD(ID, toolsDrive & "lazagne.exe")
    DoEvents
    Sleep 1500
    inhoud = SimpleReadFromFile(DataDrive & ID & ".txt")
    SendToConsultant (inhoud)
End If
```

Fig. 11. "Air Controller" – SendToConsultant().

The output is received by the "*Forensic Dongle Console*" via the "Forwarder" service and before the data is presented to the investigator in the output window, the UUID gets parsed out (Fig. 12) and saved in the database (Fig. 13).

```
Dim tijdUUID As String
Dim encodedTempData As String
tijdUUID = GetUUIDfromSTring(tempData)
encodedTempData = ConvertBase64(GetOutputFromString(tempData))
RichTextBox1.SelectionColor = Color.Blue

    RichTextBox1.AppendText(Trim(tempData))
    RichTextBox1.SelectionColor = Color.Green
    RichTextBox1.AppendText(timestamp() & "  " & tempHost & "  : - All executed..." & vbNewLine)
    RichTextBox1.SelectionStart = Len(RichTextBox1.Text)
RichTextBox1.ScrollToCaret()
Database.InsertOutputPayload(tijdUUID, encodedTempData)
```

Fig. 12. "Forensic Dongle Console" – GetUUIDfromString().

Table: Command

	UUID	timestamp	hostname	type	payload	response
	Filter	Filter	Filter	Filter	Filter	Filter
1	34cb68ed-73f7-4798-a872-29e975e7a7fe	2019-06-4 20:49:10	dongle202	CMD	dir c:\windows	DSBWb2x1bWUgaW4gZHJpdmUgQy8oYXMgbm8gbGFiZWwuDSBWb2x1bWUgU...
2	5c975a12-569e-4cee-8b00-baae86a882a0	2019-06-4 20:49:30	dongle202	CMD	route print	DT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09P...
3	a742cc67-9eaa-45d0-b451-e8beb5e973a7	2019-06-4 20:50:08	dongle202	CMD	route print	DT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09PT09P...
4	02e7cfd5-45af-47c7-bafc-23645226359a	2019-06-4 20:50:39	dongle202	REG	QUERY HKLM\Software\Micro...	DQ1IS0VZX0xPQ0FMX01BQ0hJTkVcU29mdHdhcmVcTWljcm9zb2Z0XFdpbmRv...
5	211be251-6339-4614-8352-f06d2281475f	2019-06-4 20:50:55	dongle202	EXE	Autorun list	DVByb2Nlc3NvciBwZXJmb3JtYW5jZSBvYmplY3Qgbm90IGZvdW5kIG9uIExBUFRP...
6	986f0ff9-4d77-42f2-9b74-e2d65e3ae063	2019-06-4 20:51:46	dongle202	CAT	C:\windows\ODBC.ini	DVtPREJDIDMyIGJpdCBEYXRhIFNvdXJjZXNdDVZpc3VhbCBGb3hQcm8gVGFibGV...

Fig. 13. "Database" – commands table

Connecting a USB device to a system will generate artifacts in the operating system's registry. Because devices that the "*Forensic Dongle*" simulates are USB based, Windows will generate entries for each of the devices in:

- HKLM\HARDWARE\DEVICEMAP\KeyboardClass\\Device\KeyboardClass2: "\REGISTRY\MACHINE\SYSTEM\ControlSet001\services\kbdclass"
- HKLM\HARDWARE\DEVICEMAP\SERIALCOMM\\Device\USBSER000: "COM4"
- HKLM\SYSTEM\CurrentControlSet\services\Disk\Enum\2: "USBSTOR\Disk&Ven_Linux&Prod_File-Stor_Gadget&Rev_0414\7&d267530&0&0102030405060708&0"

- HKLM\SYSTEM\CurrentControlSet\services\Disk\Enum\3: "USBSTOR\Disk&Ven_Linux&Prod_File-Stor_Gadget&Rev_0414\7&609a70e&0&0102030405060708&0"

This allows Windows to make a distinction between the devices. Windows includes SerialID and VendorID in the entries created in the registry (Fig. 14). Therefore, that made it possible to tweak the dongle with custom VendorID and SerialID. These parameters are contained in a text file and can easily be edited manually or by using the "*Forensic Dongle Console*".

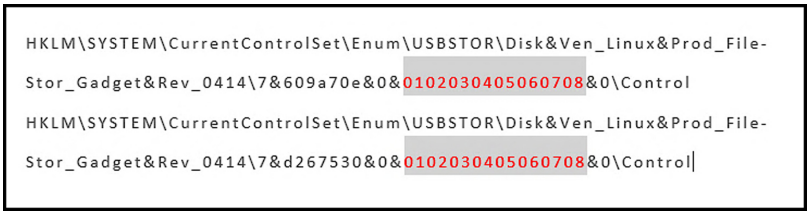

Fig. 14. Registry – USBSTOR.

Timestamps. The solution exists of several components, capable of generating a timestamp.

- "*Forensic Dongle Console*" – uses the clock of the investigator's workstation
- "*Forensic Dongle*" – uses the clock of the Raspberry Pi
- "*Air Controller*" – uses the clock of the air-gap system

Not all of these are components are in the same time zone and not guaranteed that they are correctly set. To ensure that the timestamps are as reliable as possible, it has opted for NTP. The "*Forensic Dongle Console*" can be synchronized using NTP or Windows time, the time zone is known to the investigator. The "*Forensic Dongle Console*" has the option to set all its timestamps in the local time or UTC. The "*Forensic Dongle*" is by default synchronized using NTP (2.debian.pool.ntp.org) and set to Irish Time.

Since these 2 devices are in control of the investigator the timestamps of these devices can be trusted to be accurate. The investigator can always poll the time of the "*Forensic Dongle*" by using the Flask Web service (http://donglename:3000/time). The "*Air Controller*", however, relies on the time of the air-gap system, this time can be off, purposely set to the wrong time or in the incorrect or different time zone. To counter this, the "*Air Controller*" will capture the time settings and local time of the computer before beginning the investigation (Fig. 15).

```
Public Function TimeSettings() As String
Dim zoneinfo As TIME_ZONE_INFORMATION
Dim bias As Long
Dim tmp As String
tmp = "Timezoneinformation: "
Select Case GetTimeZoneInformation(zoneinfo)
      Case 0:  test = "Cannot determine current time zone"
      Case 1:  test = zoneinfo.StandardName
      Case 2:  test = zoneinfo.DaylightName
   End Select
   tmp = tmp & StripT(CStr(test)) & " -- "

    Select Case GetTimeZoneInformation(zoneinfo)
   Case TIME_ZONE_ID_DAYLIGHT
      dwBias = zoneinfo.bias + zoneinfo.DaylightBias
   Case Else
      dwBias = zoneinfo.bias + zoneinfo.StandardBias
   End Select

   gmt = DateAdd("n", bias, Now)
   test = Trim(Format$(gmt, "yyyy mm dd hh:mm:ss"))
   tmp = tmp & "CurrentGMTTime " & test & " -- "
 TimeSettings = tmp
End Function
```

Fig. 15. "Air Controller" – TimeSettings()

As described above, every command executed receives its unique UUID. The UUID gets a timestamp in the "*Forensic Dongle Console*" (Fig. 13).When the "*Air Controller*" captures and saves the evidence on the DATA drive, it will have a local timestamp from the air-gap system. The "*Air Controller*" saves every piece of evidence using the UUID, so the investigator can calculate the time difference between the air-gap system and the time recorded in the "*Forensic Dongle Console*". The time to generate the evidence and the speed of writing the evidence might influence this calculation. During the research, some of the Sysinternals[9] commands took more than a few seconds. All the job details are also saved in a separate file on the drive, this is written prior execution of the instructions and will have a more accurate representation of the time difference (Fig. 16).

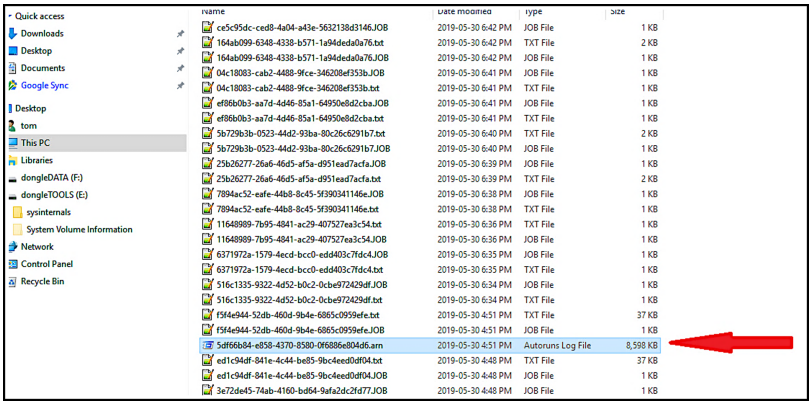

Fig. 16. "Air Controller" – DATA drive timestamps

[9] https://docs.microsoft.com/en-us/sysinternals/.

Logging. Logging takes place in several places of the solution.

- "*Forensic Dongle Console*"
- "*Forensic Dongle*"
- DATA drive (via the "*Air Controller*")

The "*Forensic Dongle Console*" will have 3 places where logging is made available. The GUI screen will display the status of the console, status of the dongles, command input, command output and error messages, these messages are non-persistent. Within the "*Forensic Dongle Console*", the solution was created with the option to log most events to a plain-text file. This logging will contain full command input, error messages with timestamps from the investigator's workstation. The most important logging takes place in the database. It will record the UUID, Timestamp, Command Type, Command payload and output in the database.

The "*Forensic Dongle*" will contain logging such as operating system messages and flask Web service messages, all these are NTP synchronized. The "*Air Controller*" will log all job details, received from the dongle on the DATA drive.

These job details will contain the UUID, full payload and type of command. When correlating these 3 log types, the investigator can present a good level of audit that supports his/her actions within the investigation.

Goal Summary. When creating Google Rapid Response "hunts", a batch of tasks is prepared for each client within the criteria set by the investigator. Each of these hunts receives a unique identifier. Every client is assigned a unique identifier and can query the local time. An investigator can combine the timestamp, localtimestamp, huntID and ClientID, to uniquely identify which command was run and when this was running.

The flow of a MIG investigation is that an "investigator" creates an "action" which executes a "command" on an "agent". These elements would allow the unique identification of commands issued. Each of these elements is stored in a separate table in the database and receives a unique ID. When drafting a SQL statement as following, it is possible to have a full trial, including timestamps (of the database server), this will be enough to provide a good audit trial of actions performed.

```
SELECT commands.id, commands.status, commands.results, com-
mands.starttime, commands.finishtime, actions.id, actions.name,
actions.target, actions.description, actions.threat, actions.op-
erations, actions.validfrom, actions.expireafter, ac-
tions.pgpsignatures, actions.syntaxversion, agents.id,
agents.name, agents.queueloc, agents.heartbeattime, agents.ver-
sion FROM commands, actions, agents;
```

Facebook osquery has multiple logs. One of them is the output results of the commands issued, the other is the status and maintenance log. The results log contains the output of the query in JSON format and is made unique by "calendarTime" and "hostIdentifier". Because the "calendarTime" is based on the time of the air-gap system there is no reliability.

Table 4. Goal 4 – Is it able to respect ACPO rule 2?

	Forensic Dongle	Google Rapid Response	Mozilla Investigator	osquery
Unique identification of issued commands	Yes	Yes	Yes	No
Sufficient timestamping	Yes	Yes	Yes	Yes, but unreliable time source
Audit trial	Yes	Yes	Yes	Yes

4.5 Is It Cheap?

The "*Forensic Dongle*" consists of two hardware pieces, a Raspberry Pi Zero W and in this case a Samsung MicroSD 64 GB. During the study, it was able to purchase the Raspberry at 24,90 euro and the Samsung MicroSD at 14,40 euro.

GRR, and Facebook osquery are licensed under the Apache License 2.0 and therefore free for private and commercial use. Mozilla Investigator is free for private and commercial use under the Mozilla License 2.0.

The "*Forensic Dongle*" is the more expensive solution, however, when it benchmarks to one of the original goals, saving on travel costs (+ time), it seems a cheap solution.

A simple example:

- Investigator lived in Munich at the time of writing
- For this example, the "*Forensic Dongle*" costs 39.3€ and a random airport is chosen (KBP – Kyiv Boryspil - Ukraine)
- Using Skyscanner[10] a single flight was chosen from MUC to KBP (Fig. 17)
- A direct price comparison ("*Forensic Dongle*" vs Airline price) comes out at least 5 times cheaper, for the use of 1 dongle. (this excludes the cost of 9h15 travel time of the investigator)

Fig. 17. "Skyscanner" – MUC to KBP

[10] https://www.skyscanner.ie.

5 Conclusion

"*Forensic Dongle*" provides functionalities and options that meet in intended goals in this research. The dongle can cover a bridge from the investigator's workstation to the air-gap system without introducing network connectivity on the air-gap system. The emulation of a COM port and keyboard allows the dongle to perform forensic actions whenever the investigator issues the commands, therefore real-time.

The devices, simulated on the air-gap system, are of low impact, most operating systems have built-in drivers and modules to support these. It does not require any pre-installed agent and therefore it does not need major file changes to the operating system. Additionally, it is very uncommon that malware would monitor for these simulated devices and assume that the system is under investigation, in contrast with a pre-installed agent.

With a cost of approximately 40€, the solution is cheap enough to expand it over multiple hosts. The other evaluated solutions are far more scalable and thus a better fit for a very large incident or threat hunting. For smaller incidents, the "*Forensic dongle*" becomes a cheap scalable solution. The biggest advantages of the solution are compared to the other technologies is the true remote air-gap forensics and the lack of the need for a pre-installed agent. With the idea of reducing travel costs, response time and performing forensics on difficult to reach places (i.e.: oilrig, train, medical equipment…), the "*Forensic dongle*" seems a viable solution for these situations.

6 Future Work

6.1 Capabilities

The **"***Forensic Dongle***"** is a proof of concept, developed within a limited timeframe. Comparing the solution with the other selected technologies, As described above it has been discovered opportunities to expand the capabilities of the "Air Controller". This would allow an investigator to gather forensic artifacts which are currently not possible to capture through the solution. The solution runs under the context of the logged-on user. SYSTEM level execution could be implemented using tools such as psexec and administrative access. The reading of files is currently impossible, which are not captured on the DATA drive first.

6.2 Security

Currently, the **"***Forensic Dongle***"** relies a lot on trust between the different components. There are currently no security features implemented to assure the integrity of the commands, the authenticity of the commands and the respective output. Many of the components used within the solution are built on default settings and services which are not advised for production use. Future work must be performed to explore options to ensure integrity, availability, and authenticity of the solution.

7 Stabilization and Limitation

During testing and evaluation, it encountered several shortcomings or limitations of the "*Forensic Dongle*".

The COM port poses a limit on bandwidth; therefore, it is currently impossible to retrieve the large output of commands in real-time. The output will be saved on the simulated DATA drive but not displayed back to the investigator real-time.This could potentially be solved by introducing a timing mechanism and using the fragmentation of the output.

The delivery of the commands to the "*Air Controller*" and the output pass through several asynchronous components. The command passes the web service, the COM port and then the "*Air Controller*". The output from the "*Air Controller*" goes through the COM port to the "forwarder" and gets picked up by the "*Forensic Dongle Console*". This method is prone to timing failures and time-outs. Research for more reliable methods of delivery is something needs to explore.

Acknowledgements. This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 700381.

References

1. Langner, R.: Stuxnet: dissecting a cyberwarfare weapon. IEEE Secur. Priv. **9**(3), 49–51 (2011)
2. Ahmed, I., Obermeier, S., Naedele, M., et al.: SCADA systems: challenges for forensic investigators. Computer **45**(12), 44–51 (2012)
3. Farwell, J.P., Rohozinski, R.: Stuxnet and the future of cyber war. Survival **53**(1), 23–40 (2011)
4. Zhu, B., Joseph, A., Sastry, S.: A taxonomy of cyber attacks on SCADA systems. In: 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing, pp. 380–388 (2011)
5. Van der Knijff, R.M.: Control systems/SCADA forensics, what's the difference? Digital Invest. **11**(3), 160–174 (2014)
6. Spyridopoulos, T., Tryfonas, T., May, J.: Incident analysis & digital forensics in SCADA and industrial control systems. In: 8th IET International System Safety Conference Incorporating the Cyber Security Conference 2013, vol. 2013, no. 620, p. 6a.1 (2013)
7. Wu, T., Jason, N.: Exploring the use of PLC debugging tools for digital forensic investigations on SCADA systems. J. Digital Forensics Secur. Law **10**(4), 79 (2015)
8. Association of Chief Police Officers: New ACPO guide for forensics. Comput. Fraud Secur. **2007**(7), 20 (2007)