

BlueBorne - Impact Analysis

Students:

Kotaiba ALACHKAR
Kees DE JONG
Adrien RAULOT
Shahrukh ZAIDI

Instructor:

Jaap VAN GINKEL

October 29, 2017

Contents

1	Introduction	1
1.1	Related Literature	1
1.2	Methodology	1
1.2.1	Vulnerability Rating and Scoring	1
2	Reverse engineering of the Armis application	4
2.1	Local checks	5
2.2	Remote checks	5
3	Critical Discoverability	7
4	Severity assessment of the BlueBorne vulnerabilities.	7
4.1	CVE-2017-1000251 (Linux)	7
4.2	CVE-2017-1000250 (Linux) and CVE-2017-0785 (Android)	8
4.3	CVE-2017-0781 and CVE-2017-0782 (Android)	8
4.4	CVE-2017-0783 (Android) and CVE-2017-8628 (Windows)	9
4.5	CVE-2017-14315 (iOS)	10

5	Data presentation and analysis	10
5.1	Consolidating the data	10
5.2	Frequency analysis	11
5.3	Vendors impact	12
5.4	Overall risk rates	14
6	Reproducibility of the scan results	14
6.1	CVE-2017-0785	14
6.2	CVE-2017-1000251	15
7	Ethical Considerations	16
8	Conclusion	16
9	Appendices	18
9.1	Scan results	18
9.2	Scan result merge script	32

1 Introduction

Bluetooth has found its way on almost any modern device ranging from desktops to smart watches or even refrigerators. Recently a set of eight zero-day vulnerabilities were discovered of which four were classified as critical. These vulnerabilities are dubbed as “BlueBorne” and are discussed in more detail in chapter 4. What makes these vulnerabilities critical is that it spreads through the air and can take full control of the devices without the victim being instantly aware of it. If a device is infected with e.g. a worm that uses these vulnerabilities, then it can spread to other vulnerable devices within its range. Hence the name BlueBorne, since it has the potential to spread like an airborne virus. Merely having Bluetooth enabled is enough to become a target. Software fixes have been released, but a small subset of these vulnerable devices are unlikely to become a software upgrade candidate due to support policies of certain vendors. In other words, this will be a permanent threat to a range of devices. This makes BlueBorne an interesting subject to study in terms of a threat analysis.

In this study we have set out to analyze the impact of the BlueBorne vulnerabilities. In order to do so we first discuss each of the vulnerabilities individually. A vulnerability rating system is used to get a deeper understanding of the severity of the vulnerabilities. Finally, we have used the BlueBorne vulnerability scanner released by Armis labs to perform scans on remote devices in order to determine what portion devices are vulnerable. To perform these scans, we have limited our scope to the Science Park canteen.

1.1 Related Literature

During this research, we first studied the BlueBorne white paper from Armis[14] to have a better insight of how important these vulnerabilities are and to get a first impression of their potential impact. In this technical white paper, Armis goes through an overview of the impacted Bluetooth stacks and describes each of the eighth discovered vulnerabilities.

1.2 Methodology

1.2.1 Vulnerability Rating and Scoring

Vulnerabilities can pose a great risk to exposed computer systems. However, not all vulnerabilities are equally severe. Therefore, vulnerability analysis is an essential process in the step-wise refinement of software products[9]. In recent years, a number of large computer security vendors have investigated a variety of approaches for measuring the relative severity of software vulnerabilities. Unfortunately, there has been no cohesion or interoperability among the different rating systems. Liu and Zhang [10] have proposed a new method for rating and scoring vulnerabilities. Their Vulnerability Rating and Scoring System (VRSS) combines the advantages of all kinds of vulnerability rating systems in order to unify these different rating frameworks.

VRSS rates a vulnerability in two steps. In the first step, the rating method, the impact of a vulnerability on affected systems is defined as a combination of losses to varying degrees of the confidentiality, integrity and availability properties to evaluate the risk levels of the vulnerability. Each of these properties is assigned one of the possible values, None, Partial or Complete, according to the impact on the respective property. Table 1 lists all the different possibilities along with the corresponding *Qualitative Level* and *Impact Score*:

<i>Possible impact metrics cases</i>	<i>Qualitative level</i>	<i>Impact Score</i>
[C:C/I:C/A:C]	High	9
[C:P/I:C/A:C], [C:C/I:P/A:C], [C:C/I:C/A:P]	High	8
[C:N/I:C/A:C], [C:C/I:N/A:C], [C:C/I:C/A:N]	High	7
[C:C/I:P/A:P], [C:P/I:C/A:P], [C:P/I:P/A:C]	High	6
[C:C/I:P/A:N], [C:C/I:N/A:P], [C:P/I:C/A:N] [C:P/I:N/A:C], [C:N/I:C/A:P], [C:N/I:P/A:C]	Medium	5
[C:C/I:N/A:N], [C:N/I:C/A:N], [C:N/I:N/A:C]	Medium	4
[C:P/I:P/A:P]	Medium	3
[C:N/I:P/A:P], [C:P/I:N/A:P], [C:P/I:P/A:N]	Medium	2
[C:P/I:N/A:N], [C:N/I:P/A:N], [C:N/I:N/A:P]	Low	1
[C:N/I:N/A:N]	Low	0

Table 1: Impact on each of the confidentiality, integrity and availability properties. Each property is assigned a value: None(N), Partial(P) or Complete(C). The corresponding *Qualitative Level* and *Impact Score* are shown.[\[10\]](#)

For a vulnerability with base metric values of “Confidentiality impact: None, Integrity impact: Partial, Availability impact: Complete” the impact vector “[C:N/I:P/A:C]” would be applied. Table 1 shows that this vector has a qualitative rating level of “Medium” and an impact score of “5”. Table 2 below shows how the impact on the different properties is determined:

<i>Confidentiality impact</i>	<i>Description</i>
None (N)	There is no impact to the confidentiality of the system.
Partial (P)	There is considerable informational disclosure. Access to some file systems is possible, but the attacker does not have control over what is obtained, or the scope of the loss is constrained.
Complete (C)	There is total information disclosure, resulting in all files being revealed.
<i>Integrity impact</i>	<i>Description</i>
None (N)	There is no impact to the integrity of the system.
Partial (P)	Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited.
Complete (C)	There is total compromise of system integrity. The attacker can modify any files on the system.
<i>Availability impact</i>	<i>Description</i>
None (N)	There is no impact to the availability of the system.
Partial (P)	There is reduced performance or interruptions in resource availability.
Complete (C)	There is a total shutdown of the attacked resource. The resource is rendered completely unavailable.

Table 2: Confidentiality, Integrity and Availability impact scoring evaluation.[11]

The second step, the scoring method, is to calculate the *Exploitability Score*. The metrics used to calculate this score are listed in table 3. The exploitability metrics capture how the vulnerability is accessed and whether or not extra conditions are required to exploit the vulnerability.

<i>Exploitability metric</i>	<i>Metric value</i>	<i>Quantitative Score</i>
Access Vector	Local/Adjacent Network/Network	0.395/0.646/1.0
Access Complexity	High/Medium/Low	0.35/0.61/0.71
Authentication	None/Single/Multiple	0.704/0.56/0.45

Table 3: Exploitability metrics along with the corresponding *Quantitative Score*. [10]

Table 4 shows how the values for these metrics are determined:

<i>Access Vector</i>	<i>Description</i>
Local	Attacker must either have physical access to the vulnerable system or a local (shell) account.
Adjacent Network	Attacker must have access to either the broadcast or collision domain of the vulnerable system.
Network	The vulnerability is exploitable with network access and, therefore, “remotely exploitable”.
<i>Access Complexity</i>	<i>Description</i>
High	Specialized conditions exist for access to the vulnerability.
Medium	The access conditions are somewhat specialized. Some additional requirements need to be met in order to exploit the vulnerability.
Low	There are no specialized access conditions for access to the vulnerability.
<i>Authentication</i>	<i>Description</i>
Multiple	Exploiting the vulnerability requires the attacker to authenticate two or more times.
Single	One instance of authentication is required in order to access and exploit the vulnerability.
None	Authentication is not required in order to access and exploit the vulnerability.

Table 4: Access Vector, Access Complexity and Authentication scoring evaluation.[11]

The Exploitability Score can be calculated using the following formula:

$$\text{Exploitability Score} = 2 \times \text{Access Vector} \times \text{Access Complexity} \times \text{Authentication}$$

For a vulnerability with exploitability metric values of “Access Vector: Network, Access Complexity: Medium, Authentication: None” the Exploitability Score would be: $2 \times 1.0 \times 0.61 \times 0.704 = 0.86$. Finally, the *Quantitative Score* can now be calculated using the following formula:

$$\text{Quantitative Score} = \text{Impact Score} + \text{Exploitability Score}$$

When analyzing the BlueBorne vulnerabilities in this paper, we will first give a brief description of each of the BlueBorne vulnerabilities. After describing the vulnerability, we elaborate on the risk levels of the vulnerability in order to come to a final assessment of the vulnerability using the previously described rating method. As the technical details of the vulnerability are not relevant for the assessment, we will refrain from going in to too much detail. Instead, we will keep our focus on the impact of the vulnerability. Note that the *Access Vector* and *Authentication* metrics for all of the vulnerabilities described will be “Adjacent Network” and “None” respectively, as the BlueBorne vulnerabilities leverage Bluetooth connections and require no form of authentication.

2 Reverse engineering of the Armis application

In order to gain a comprehensive understanding of the BlueBorne risk rating and in affect how the application determines whether a device is vulnerable or not, we have reversed engineered the BlueBorne vulnerability scanner made by Armis[13].

2.1 Local checks

To determine whether a device that runs the application is vulnerable or not, the system property `Build.VERSION.SECURITY_PATCH` or `ro.build.version.security_patch` is pulled locally from the device and compared to the *GregorianCalendar*(2017, 8, 1) date. If the device security patch date is after 01-08-2017, the application assumes that the device is not vulnerable or at a low risk (according to the application's own risk rating), if it is before that date it is assumed that the device is vulnerable and at a high risk. On 01-08-2017 all the OEMs and device manufacturers released security patches to patch the eight vulnerabilities that form the BlueBorne attack vector.

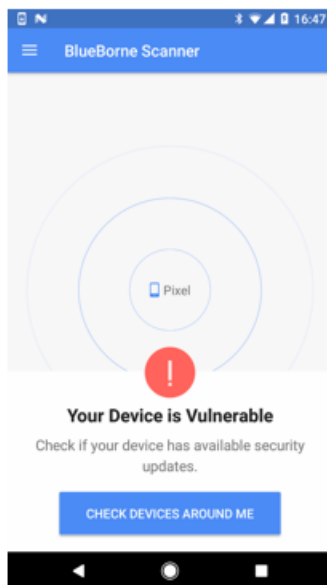


Figure 1: Vulnerable Local Device Check

2.2 Remote checks

The most important part of this application is the possibility to check nearby devices whether they are vulnerable or not. In order to use this functionality, two permissions are requested (Coarse location permission and Bluetooth permission). Then the remote device address prefix is mapped to a manufacturer name from a static list; e.g., "Samsung", or "Apple". These devices are classified based on platform name; e.g., "iOS", "Android". Risk is then rated based on remote device classification:

Classification	Label	Risk
iOS	Medium	2
Android	High	3
Low Energy	Low	1
Unknown Manufacturer	Low	1

Figure 2: Devices classification, label, and Risk

According to the application risk rating, unknown manufacturer devices and low energy devices (DEVICE_TYPE_LE from Bluetooth device types) are considered a low risk. In summary, the “scan nearby devices” functionality checks for the device manufacturer and type, which are then used to rate the risk on scale from 1 to 3 based on an array of static values. e.g., a device manufactured by “HTC” is always an “Android” device which always has a score of 3 (high risk). This risk rating criteria seems relatively impractical since Armis clearly stated that “iOS” devices are at lower risk than Android devices.

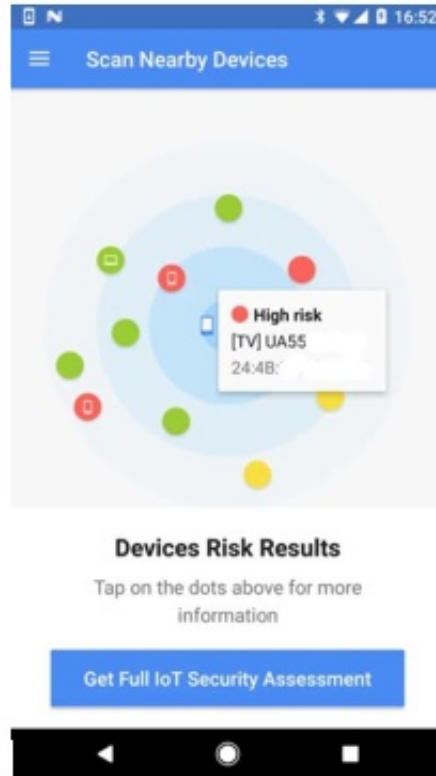


Figure 3: Check Nearby Devices

3 Critical Discoverability

Bluetooth on many devices is turned on by default. On most platforms, when the user is trying to pair a device, the device is in discoverable mode. In any other case, discoverability is disabled. Bluetooth devices are almost always listening for unicast traffic targeted to it, even when it is not set on discoverable mode. For this reason, to establish a connection the initiating party only needs to know the BDADDR (Bluetooth device address, MAC address) of the target device[14]. Once an attacker acquires it, and is in physical proximity of the device (RF range) he or she can reach the surprisingly wide attack surface of its listening Bluetooth services. Discovering BDADDRs of non-discoverable devices is considered difficult due to the assumed lack of hardware capabilities and to the complexity of the Bluetooth protocol. However, this is not the case anymore since open source hardware tools, like Ubertooth, have been available for a number of years. These tools allow to sniff and monitor the protocol in the physical and link layers by sniffing the air for Bluetooth packets. Although the Bluetooth connections are encrypted, the packet headers are in plain text. These headers contain enough information from which the BDADDRs of communicating devices can be derived. This allows an attacker in physical proximity to derive the BDADDR of a device and use it to send unicast traffic.



Figure 4: General Basic Rate packet format

In case the the device generates no Bluetooth traffic, and is only listening, it is still possible to “guess” the BDADDR, by sniffing its Wi-Fi traffic. This is possible since Wi-Fi MAC addresses appear unencrypted over the air and due to the widely accepted norm of OEMs and hardware manufacturers that the MACs of internal Bluetooth/Wi-Fi adapters are either the same, or only differ in the last digit (one being +1 of the other) [14].

4 Severity assessment of the BlueBorne vulnerabilities.

In the following sections we will briefly describe each of the BlueBorne vulnerabilities along with its impact as discussed in the BlueBorne white paper[14]. We then use the previously discussed VRSS in order to rate the severity of each of the vulnerabilities.

4.1 CVE-2017-1000251 (Linux)

This vulnerability is the result of erroneous processing of L2CAP (Logical Link Control and Adaptation Protocol) configuration responses by the Bluetooth stack of the Linux Kernel, which can result in a stack buffer overflow. On systems without mitigation techniques to prevent memory corruption, an attacker could use this flaw to remotely execute arbitrary code on the system with kernel level privileges. As the attacker acquires full control over the system, the confidentiality and integrity properties are highly affected. The attack, however, does not fully prevent the owner from accessing the system. Therefore, availability is affected partially. The BlueBorne white paper[14] mentions that triggering this vulnerability does require some complicated groundwork. A minimum stack implementing the HCI, ACL and L2CAP layers of the

Bluetooth stack have to be created in order to exploit the vulnerability. Therefore, we assign this vulnerability a *Access Complexity* level of “high”. The rating for this vulnerability is shown in table 5:

Table 5: CVE-2017-1000251 vulnerability rating

<i>Impact metrics</i>		<i>Impact Score</i>
Confidentiality: C, Integrity: C, Availability: P		8.0
<i>Exploitability metric</i>	<i>Metric value</i>	<i>Quantitative Score</i>
Access Vector	Adjacent Network	0.646
Access Complexity	High	0.35
Authentication	None	0.704
Quantitative Score = $8 + (2 \times 0.646 \times 0.35 \times 0.704) = 8.32$		

4.2 CVE-2017-1000250 (Linux) and CVE-2017-0785 (Android)

These vulnerabilities reside in the SDP (Service Discovery Protocol) server of the Bluetooth stack. An information-disclosure flaw exists in the `bluetoothd` daemon implementation of this protocol. The vulnerability allows an attackers to send a set of crafted SDP search attribute requests which could expose sensitive data from the `bluetoothd` process memory. This process holds critical data, such as encryption keys used in Bluetooth communications, address space and valuable pointers. Although the information that is retrieved exploiting these vulnerabilities may be extremely sensitive, the attacker does not have access to all the data on the system (yet). These vulnerabilities do not directly affect the integrity and availability properties either. As there are no special conditions for access to the vulnerability, we assign these vulnerabilities an *Access Complexity* of “low”. The rating for these vulnerabilities is shown in table 6:

Table 6: CVE-2017-1000250 and CVE-2017-0785 vulnerability rating

<i>Impact metrics</i>		<i>Impact Score</i>
Confidentiality: P, Integrity: N, Availability: N		1.0
<i>Exploitability metric</i>	<i>Metric value</i>	<i>Quantitative Score</i>
Access Vector	Adjacent Network	0.646
Access Complexity	Low	0.71
Authentication	None	0.704
Quantitative Score = $1 + (2 \times 0.646 \times 0.71 \times 0.704) = 1.65$		

4.3 CVE-2017-0781 and CVE-2017-0782 (Android)

These two Android vulnerabilities are caused by weak buffer size allocation in the BNEP (Bluetooth Network Encapsulation Protocol) service of the Bluetooth stack. Exploitation of the vulnerability can lead to heap

overflows with data that is attacker-controlled, which enables the attacker to run code on the device. The remote code is executed under the privileges of the `com.android.bluetooth` service. As a highly privileged service on the Android system, this grants the attacker access to the file system, full control of the network stack and the ability to simulate an attached keyboard or mouse and consequently have full control over a device. As the attacker acquires full control over the device, the confidentiality and integrity properties are highly affected. The attack, however, does not fully prevent the owner from accessing the system. Therefore, availability is affected partially. The first vulnerability (CVE-2017-0781) does not require any special conditions for access to the vulnerabilities. Therefore, we assign this vulnerability an *Access Complexity* of “low”. The second vulnerability, however, does require grooming of the heap prior to the heap overflow for successful remote code execution. Therefore the *Access Complexity* of this vulnerability is rated as “medium”. The rating for these vulnerabilities is shown in the tables below:

Table 7: CVE-2017-0781 vulnerability rating

<i>Impact metrics</i>		<i>Impact Score</i>
Confidentiality: C, Integrity: C, Availability: P		8.0
<i>Exploitability metric</i>	<i>Metric value</i>	<i>Quantitative Score</i>
Access Vector	Adjacent Network	0.646
Access Complexity	Low	0.71
Authentication	None	0.704
Quantitative Score = $8 + (2 \times 0.646 \times 0.71 \times 0.704) = 8.65$		

Table 8: CVE-2017-0782 vulnerability rating

<i>Impact metrics</i>		<i>Impact Score</i>
Confidentiality: C, Integrity: C, Availability: P		8.0
<i>Exploitability metric</i>	<i>Metric value</i>	<i>Quantitative Score</i>
Access Vector	Adjacent Network	0.646
Access Complexity	Medium	0.61
Authentication	None	0.704
Quantitative Score = $8 + (2 \times 0.646 \times 0.61 \times 0.704) = 8.55$		

4.4 CVE-2017-0783 (Android) and CVE-2017-8628 (Windows)

These vulnerabilities describe a Man-in-the-Middle attack possibility caused by low “Security Level” requirements defined by the Android and Windows Bluetooth stack for the PAN (Personal Area Network) profile. The vulnerability allows an attacker to create a malicious network interface on a vulnerable device and re-configure IP routing so that all traffic is routed through the attacker. This enables the attacker to intercept,

inject or alter data being sent by or destined for the victim. As the scope of the data available to the attacker is limited to the network traffic, the confidentiality metric is impacted partially. The same holds true for the integrity property. Modification of the data is possible. However, the scope of this modification is limited to the network data of the victim. A Man-in-the-Middle attack does not prevent the victim from accessing the system, therefore, the availability is unaffected. The vulnerabilities do not require any special conditions for access. Thus, we assign these vulnerabilities an *Access Complexity* of “low”. The vulnerability rating for these vulnerabilities are shown in table 9:

Table 9: CVE-2017-0783 and CVE-2017-8628 vulnerability rating

<i>Impact metrics</i>		<i>Impact Score</i>
Confidentiality: P, Integrity: P, Availability: N		2.0
<i>Exploitability metric</i>	<i>Metric value</i>	<i>Quantitative Score</i>
Access Vector	Adjacent Network	0.646
Access Complexity	Low	0.71
Authentication	None	0.704
Quantitative Score = $2 + (2 \times 0.646 \times 0.71 \times 0.704) = 2.65$		

4.5 CVE-2017-14315 (iOS)

This vulnerability resides in a protocol invented by Apple, the LEAP (Low energy audio protocol). This protocol is used by iOS to stream audio to low energy audio peripherals. A flaw in the implementation of LEAP, allows an attacker to send a large audio command to the targeted device. As the size of the incoming audio chunks is wrongfully assumed to be a limited number of bytes, a larger packet can create a significant overflow with attacker-controlled data. The memory corruption can lead to remote code execution in the context of iOS’s Bluetooth stack and grant the attacker full control of the device. We have seen in previous vulnerabilities how remote code execution affects the confidentiality, integrity and availability impact metrics. This vulnerability does not require any special conditions for access. The vulnerability rating will, therefore, be similar to the rating for vulnerability CVE-2017-0781 (see table 7).

5 Data presentation and analysis

In order to have a good representation of the BlueBorne impact, we performed seven one minute scans over the course of three weeks. The scans have been performed by our Android devices using the BlueBorne Vulnerability Scanner mobile application by Armis [13], at the University of Amsterdam canteen, at Science Park. To ensure the quality and fidelity of the scans, the same setup has been used every time.

5.1 Consolidating the data

Before analyses could be done we first had to consolidate the data in a concise way and add extra information such as the MAC prefix and vendor ID, a Python script was created for this purpose[3]. The predictable output was presented as a single CSV file (9.1) which made it easy to conduct further data analysis.

5.2 Frequency analysis

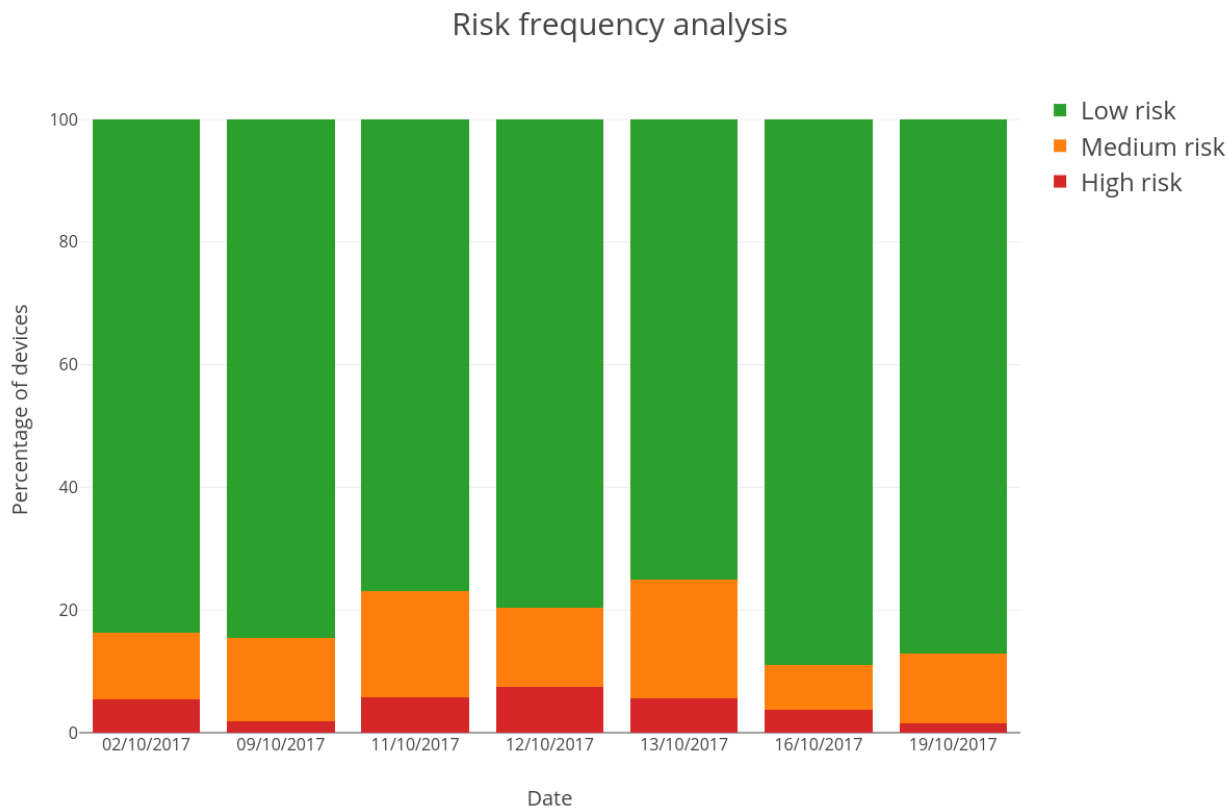


Figure 5: Risk frequency

With this method, it is possible to present an evolution of the impact. We based our risk analysis on the risk levels defined by Armis in their scanner, this include a low, medium and high risk.

In figure 5, the first two days are Mondays. Even though there is a clear predominance of low risk devices, almost 20% of the scanned devices this day are at medium or high risk.

Days from the 9th of October to the 13th are on the same week. Less devices were detected by the scanner on the 9th of October, which can explain the difference in proportion. However, from the 11th to the end of the week, we scanned a lot of devices and the rates of potentially vulnerable devices exceed 20%, the highest being on October 13th.

With a large number of scanned devices in the last week as well, it is interesting to observe the decline of high risk devices. This demonstrates that there is a potentially noticeable increase in patch activity.

5.3 Vendors impact

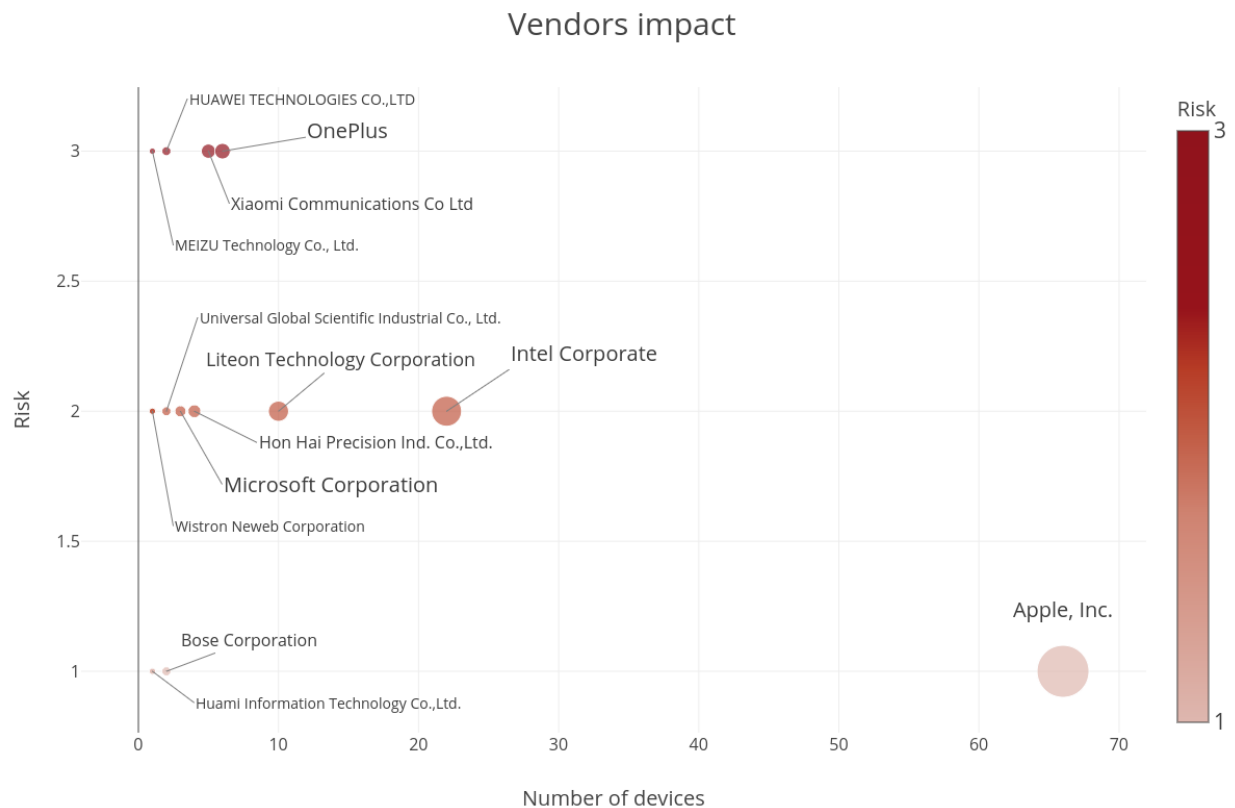


Figure 6: Vendors impact

During our scans and data reporting, we started to notice that Apple devices represented a massive part of the devices scanned at low risk.

If we take a look at the vendor impact figure 6, Apple devices don't seem to be vulnerable. The vulnerability affecting iOS has been mitigated in iOS 10, which could be the reason that no Apple devices were found vulnerable or the Armis scanner considered them not vulnerable. On the other hand, it is important to notice that many other vendors were detected highly vulnerable. It appears most of them are Chinese vendors, the most known being OnePlus which also was the most impacted vendor in our data. Yet, Apple devices are distorting our data in a sense that a lot of scanned devices were Apple products, and none of them has ever been detected vulnerable.

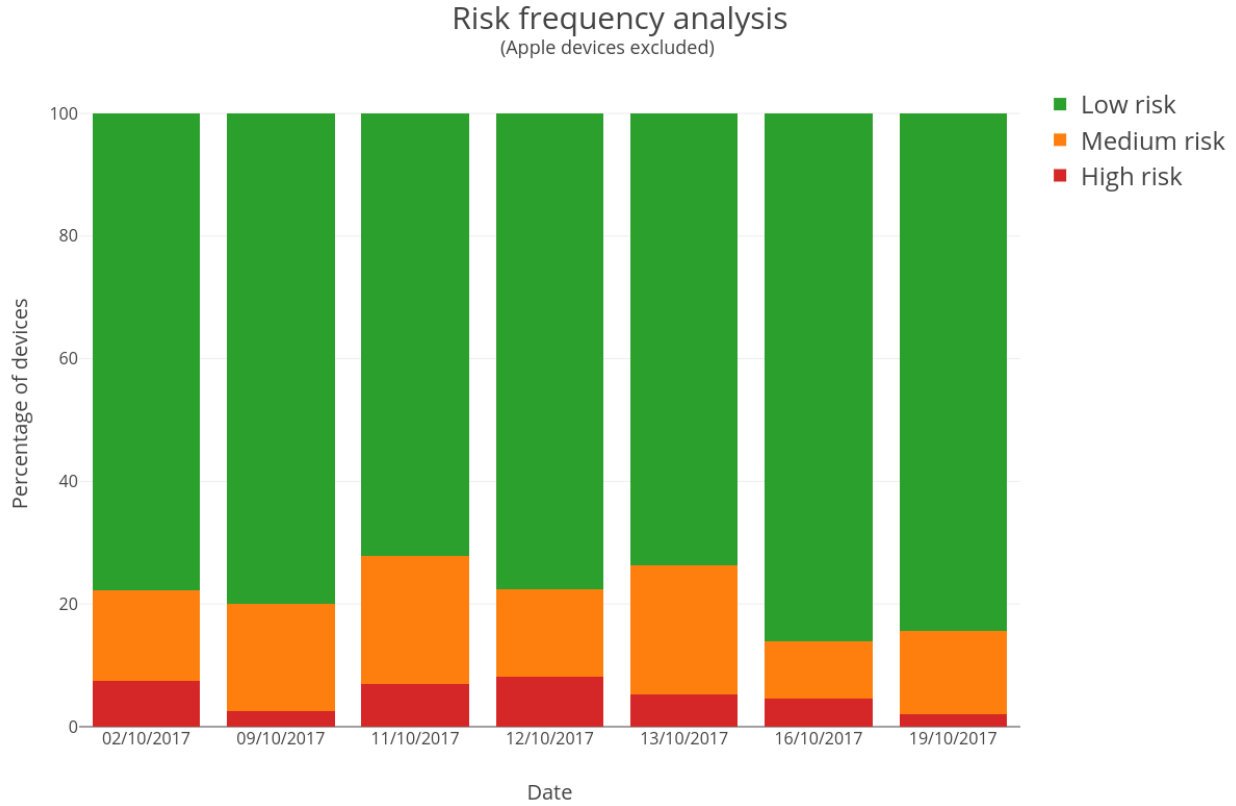


Figure 7: Risk frequency (excluding Apple devices)

Therefore, we decided to exclude Apple devices from our data (figure 7) to have a better impact overview for the other devices. This mainly includes Android, Linux and their derivatives, Windows and Windows mobile.

The overall risk rates of each day is significantly increased. The percentage of potentially vulnerable devices in the first two days is now at 20% and above. The highest rate is seen on the 11th of October with a rate close to 30% which is higher than on October 13th in the previous representation.

Again, it is interesting to see the decreased rate of devices at risk on the last week. On one hand, it is a good sign that more and more people at the University of Amsterdam patch their device. However, we could argue that this decrease is coming late. Patches for a most devices have been released early or mid-September, yet this graph shows us that some devices still haven't got an update and/or users haven't patched their device early enough. In practice, this let a one month window period with a high rate of vulnerable devices susceptible to be attacked.

5.4 Overall risk rates

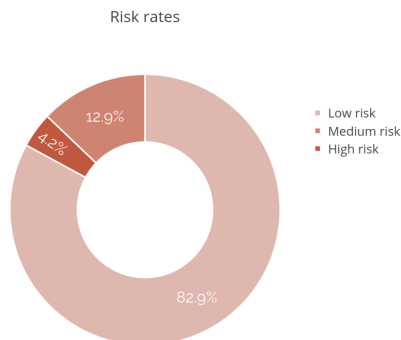


Figure 8: Risk rates

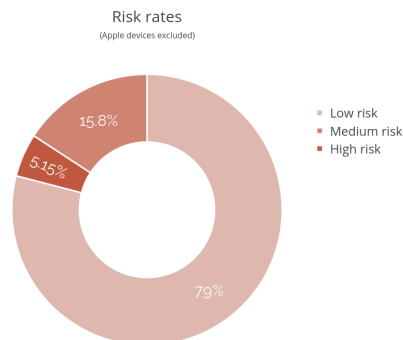


Figure 9: Risk rates (excluding Apple device)

In total, more than 300 devices were scanned on seven days using the Armis scanner. The proportion of devices at low risk is a majority (figure 8), but 17.1% of the devices are still at risk, including 4.2% at high risk. Taking in account the criticality of the vulnerabilities, this is not negligible.

If we exclude Apple devices (figure 9), the total rate of potentially vulnerable devices increased to almost 21% including 5.15% of devices at high risk. This represents an increase of 1% for devices at high risk and 3% for devices at medium risk compared to the previous representation.

With only 79% of the scanned devices at low risk, the impact of the BlueBorne vulnerabilities at the University of Amsterdam has been important. We have noticed a clear improvement within the last week but some devices, even though recent (OnePlus for instance), are still at high risk and could remain vulnerable for a while.

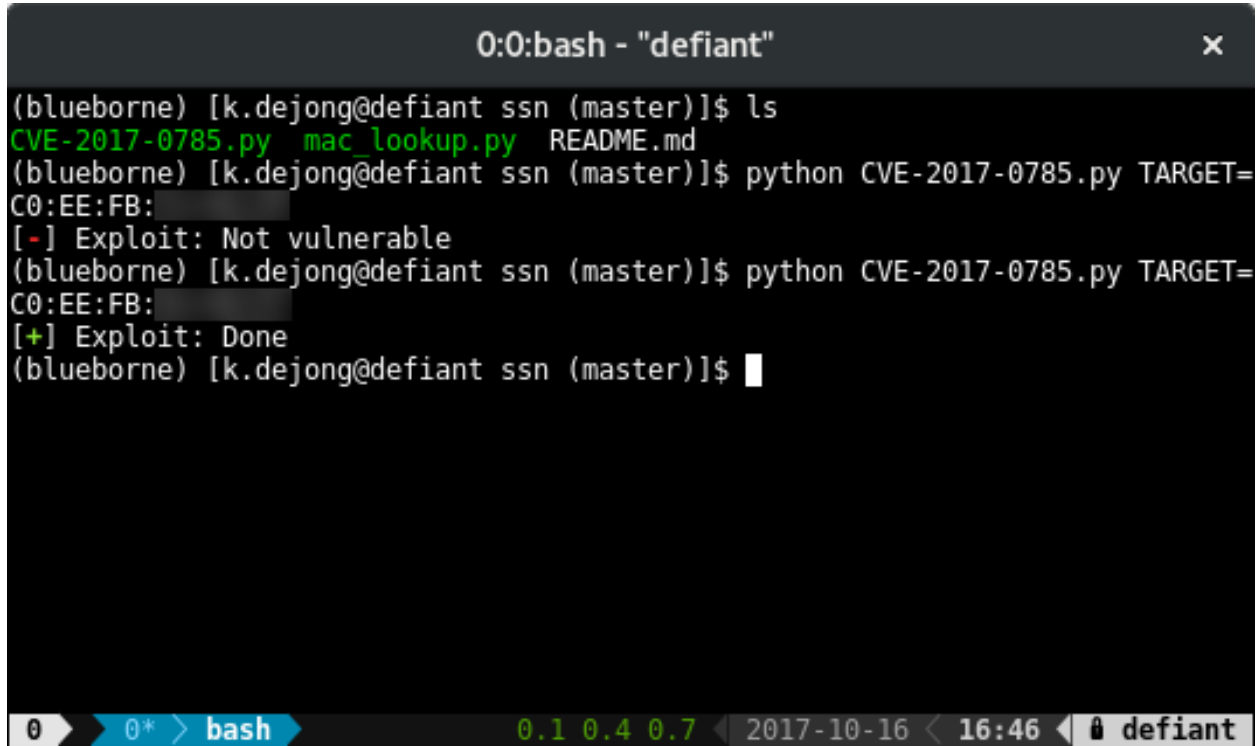
6 Reproducibility of the scan results

We needed to reproduce the results provided by the Armis scanner in order to conclude that these results are indeed factual. We used two project owned test devices for this purpose; a Raspberry Pi to test GNU/Linux and a OnePlus One to test Android. As of October 25th 2017 Armis did not release any code that could be used to effectively exploit a device. Since these vulnerabilities were disclosed shortly after our research began there were close to none publicly available exploits, with the exception of two which we used on our test devices.

6.1 CVE-2017-0785

As discussed in more detail in 4.3, this vulnerability makes an information leak possible by sending fragmented packets to the SDP (Service Discovery Protocol) service which then in turn is tricked to send back “out of bound” bytes which concludes the information leak. We found an already working proof of concept[12] of

this vulnerability and slightly modified it to improve the output result[2], as shown in figure 10. With an October 5th Android security patch level the device is registered as “Not vulnerable”, the second scan is with an August 5th Android security patch level which registers as “Done” which means the information leak occurred. These results collaborate with the results of the Armis scanner.



```
0:0: bash - "defiant"
(blueborne) [k.dejong@defiant ssn (master)]$ ls
CVE-2017-0785.py mac_lookup.py README.md
(blueborne) [k.dejong@defiant ssn (master)]$ python CVE-2017-0785.py TARGET=
C0:EE:FB:
[-] Exploit: Not vulnerable
(blueborne) [k.dejong@defiant ssn (master)]$ python CVE-2017-0785.py TARGET=
C0:EE:FB:
[+] Exploit: Done
(blueborne) [k.dejong@defiant ssn (master)]$
```

The terminal window shows the execution of a proof of concept exploit for CVE-2017-0785. The user runs 'ls' to list files, then 'python CVE-2017-0785.py TARGET=C0:EE:FB:' twice. The first run results in '[-] Exploit: Not vulnerable', and the second run results in '[+] Exploit: Done'. The terminal status bar at the bottom shows '0.1 0.4 0.7' and the date '2017-10-16'.

Figure 10: CVE-2017-0785 Proof of Concept exploit

6.2 CVE-2017-1000251

We found a workable proof of concept exploit[8] for the buffer-overflow vulnerability discussed in 4.1. We failed to get the proof of concept vulnerability working on a February 2016 Raspbian image, which did not contain the BlueBorne software fixes. The Raspberry Pi was unaffected due to the fact that KASLR (Kernel Address Space Layout Randomization) is enabled on the Raspbian distribution. As mentioned on page 14 in the Armis paper[14], KASLR is one of the mitigation implementations that prevents the exploitation of this vulnerability. In the Debian rules file of Bluez[4] it states that the hardening +ALL switch is enabled[5]. This switch includes the PIE (Position Independent Executable) build option, which takes advantage of the KASLR ability. The results collaborated with the Armis scanner which displayed a no risk indicator for both a fully patched Raspberry Pi and a 'vulnerable' one.

7 Ethical Considerations

We are aware that the data collected using the BlueBorne vulnerability scanner could potentially be used by a malicious person in order to target vulnerable devices. Therefore, we want to stress that the collected data has been used strictly for research purposes. We have not used the data to conduct any attack, nor have we published the full MAC addresses of scanned devices. Instead, we only show the MAC prefixes that identify the vendor. In addition to that, before we proceeded into the reverse engineering for the mobile application in our project we took into consideration that the process must be legal. Armis lab did not provide an End user license agreement (EULA) for the application so we refer to the U.S. and EU laws for copyright protection systems. According to U.S. Code, Section 103(f), Title 17, Chapter 12, Paragraph 4 of the Digital Millennium Copyright Act (DMCA)[1], it is legal to apply reverse engineering techniques for academic or learning purposes but it should not be retained longer than necessary to make such good faith determination and may not be used for any other purpose. Also, Article 6 of the 1991 EU Computer Programs Directive[6] allows reverse engineering for the purposes of interoperability, but prohibits it for the purposes of creating a competing product, and also prohibits the public release of information obtained through reverse engineering of software. After completion of this study, all data collected and the application source code obtained through reverse engineering will be destroyed.

8 Conclusion

Based on the trends we see in the graphs shown in section 5, we can conclude that Apple devices are mostly unaffected by BlueBorne. Another notable trend is that the Chinese manufactures have a high vulnerability score. We also noticed a good proportion of devices patched in the last week even though this decrease of devices at high risk happens only one month after the publishing of the BlueBorne vulnerabilities. Through our impact analysis we can clearly observe the important impact of BlueBorne within the University. Moreover, a future work could involve having a look at the devices that won't be patched shortly or will not be patched at all. A good approach could be to test if a restrictive application like the Bluetooth Firewall[7] in the Google Play Store adds any security.

References

- [1] 17 u.s. code 1201 - circumvention of copyright protection systems. <https://www.law.cornell.edu/uscode/text/17/1201>.
- [2] K. de Jong. Cve-2017-0785 poc ('forked' by ssn project group). <https://github.com/AquaLite/ssn/blob/master/CVE-2017-0785.py>.
- [3] K. de Jong. Scan result merge script. https://github.com/AquaLite/ssn/blob/master/mac_lookup.py.
- [4] Debian. Bluez, deb package rules file. <https://anonscm.debian.org/cgiit/pkg-bluetooth/bluez.git/tree/debian/rules#n5>.
- [5] Debian. Hardening, dpkg-buildflags. <https://wiki.debian.org/Hardening#dpkg-buildflags>.
- [6] C. Directive. 91/250/eec of 14 may 1991 on the legal protection of computer programs. *Official Journal L*, 122(17):05, 1991.

- [7] FruitMobile. Bluetooth firewall. <https://play.google.com/store/apps/details?id=com.fruitmobile.android.bluetooth.firewall&hl=en>.
- [8] M. Kozlowski. Cve-2017-1000251 poc. <https://www.exploit-db.com/exploits/42762/>.
- [9] I. Krsul. Computer vulnerability analysis: Thesis proposal. 1997.
- [10] Q. Liu and Y. Zhang. Vrss: A new system for rating and scoring vulnerabilities. *Computer Communications*, 34(3):264–273, 2011.
- [11] P. Mell, K. Scarfone, and S. Romanosky. A complete guide to the common vulnerability scoring system version 2.0. In *Published by FIRST-Forum of Incident Response and Security Teams*, volume 1, page 23, 2007.
- [12] K. Ojasoo. Cve-2017-0785 poc. <https://github.com/ojasookert/CVE-2017-0785.git>.
- [13] A. Security. Blueborne vulnerability scanner by armis (version 1.07)[mobile application software]. https://play.google.com/store/apps/details?id=com.armis.blueborne_detector&hl=nl, 2017.
- [14] B. Seri and G. Vishnepolsky. BlueBorne the dangers of bluetooth implementations: Unveiling zero day vulnerabilities and security flaws in modern bluetooth stacks. <http://go.armis.com/hubfs/BlueBorne%20Technical%20White%20Paper.pdf>.

9 Appendices

9.1 Scan results

Table 10: Scan results, part 1

Brand	Prefix	02-10-2017	09-10-2017	11-10-2017
N/A	53-13-0A	N/A	N/A	N/A
N/A	40-59-EF	N/A	N/A	N/A
N/A	FB-F8-AE	N/A	N/A	N/A
Apple, Inc.	8C-85-90	N/A	N/A	N/A
Liteon Technology Corporation	30-10-B3	N/A	N/A	N/A
Apple, Inc.	F0-79-60	N/A	N/A	N/A
N/A	53-20-0C	GREEN	N/A	N/A
N/A	75-E4-F9	N/A	N/A	N/A
N/A	41-E7-5A	N/A	GREEN	N/A
N/A	68-0A-8A	N/A	N/A	N/A
N/A	49-18-40	N/A	N/A	GREEN
N/A	56-A2-AC	N/A	N/A	N/A
N/A	4E-4C-29	N/A	GREEN	N/A
N/A	78-50-6C	N/A	N/A	N/A
N/A	60-DE-76	N/A	N/A	GREEN
Apple, Inc.	AC-BC-32	N/A	N/A	N/A
N/A	55-00-6E	N/A	N/A	GREEN
N/A	50-55-B0	GREEN	N/A	N/A
N/A	58-1D-F8	N/A	N/A	N/A
PEGATRON CORPORATION	60-02-92	N/A	N/A	N/A
N/A	FE-DD-4F	N/A	N/A	N/A
N/A	58-C0-63	N/A	N/A	N/A
Apple, Inc.	00-16-CB	N/A	N/A	GREEN
N/A	74-88-9F	N/A	N/A	N/A
Intel Corporate	D0-57-7B	N/A	N/A	N/A
N/A	57-A6-72	GREEN	N/A	N/A
N/A	79-08-05	N/A	N/A	N/A
N/A	47-BF-BC	N/A	N/A	GREEN
N/A	7C-25-03	N/A	N/A	N/A
Apple, Inc.	78-4F-43	N/A	N/A	N/A
N/A	5F-17-AA	N/A	N/A	N/A
Apple, Inc.	98-01-A7	N/A	N/A	N/A
N/A	49-A3-4B	N/A	GREEN	N/A
N/A	56-00-42	N/A	N/A	N/A
N/A	62-61-49	N/A	N/A	N/A
N/A	56-B9-E4	N/A	N/A	N/A
N/A	79-F2-6B	N/A	N/A	N/A
Intel Corporate	00-21-5C	N/A	N/A	YELLOW
N/A	49-9B-9A	N/A	N/A	N/A

Table 10: Scan results, part 1

Brand	Prefix	02-10-2017	09-10-2017	11-10-2017
Xiaomi Communications Co Ltd	10-2A-B3	N/A	N/A	N/A
Wistron Neweb Corporation	60-02-B4	N/A	N/A	N/A
Apple, Inc.	B8-E8-56	N/A	N/A	N/A
N/A	48-69-A3	N/A	GREEN	N/A
N/A	41-86-3E	N/A	N/A	N/A
N/A	EC-7A-B8	N/A	N/A	N/A
N/A	4B-93-DB	N/A	N/A	N/A
N/A	60-1A-8D	N/A	N/A	N/A
Huami Information Technology Co.,Ltd.	88-0F-10	N/A	GREEN	N/A
N/A	49-8D-12	N/A	N/A	N/A
N/A	55-B4-5C	N/A	N/A	N/A
N/A	59-0F-60	GREEN	N/A	N/A
Apple, Inc.	60-F8-1D	N/A	N/A	N/A
N/A	FE-08-55	N/A	N/A	GREEN
N/A	4F-3E-35	N/A	GREEN	N/A
N/A	77-C0-29	N/A	N/A	N/A
Apple, Inc.	64-B9-E8	GREEN	GREEN	N/A
N/A	56-A4-01	N/A	N/A	N/A
N/A	7A-94-90	N/A	GREEN	N/A
N/A	88-0F-19	N/A	GREEN	N/A
N/A	7F-52-3A	N/A	N/A	N/A
N/A	7B-53-9D	N/A	N/A	N/A
N/A	45-4E-13	N/A	GREEN	N/A
N/A	61-B8-BE	N/A	N/A	N/A
N/A	4E-C1-14	N/A	N/A	N/A
Apple, Inc.	10-40-F3	N/A	GREEN	N/A
N/A	65-03-64	N/A	GREEN	N/A
Apple, Inc.	78-4F-43	N/A	GREEN	GREEN
N/A	CE-DE-D5	N/A	N/A	N/A
N/A	6A-A8-8A	N/A	N/A	GREEN
N/A	00-EC-0A	N/A	N/A	RED
N/A	75-58-B0	N/A	N/A	N/A
N/A	5F-DA-99	N/A	GREEN	N/A
Apple, Inc.	78-31-C1	GREEN	N/A	GREEN
Apple, Inc.	AC-BC-32	N/A	GREEN	N/A
N/A	79-DD-AA	N/A	N/A	N/A
Apple, Inc.	78-4F-43	N/A	N/A	N/A
OnePlus Tech (Shenzhen) Ltd	C0-EE-FB	N/A	N/A	N/A
N/A	55-51-EF	N/A	N/A	N/A
N/A	6A-29-7D	N/A	GREEN	N/A
N/A	5F-F7-C1	N/A	N/A	N/A
N/A	5B-1A-C5	N/A	GREEN	N/A
Apple, Inc.	B8-E8-56	N/A	GREEN	N/A

Table 10: Scan results, part 1

Brand	Prefix	02-10-2017	09-10-2017	11-10-2017
N/A	C6-17-B6	N/A	N/A	N/A
Apple, Inc.	6C-40-08	N/A	N/A	GREEN
AzureWave Technology Inc.	24-0A-64	N/A	N/A	N/A
N/A	5E-0B-32	N/A	N/A	N/A
Apple, Inc.	D0-A6-37	N/A	N/A	N/A
Microsoft Corporation	BC-83-85	N/A	N/A	YELLOW
N/A	69-B9-62	N/A	N/A	N/A
N/A	79-29-30	N/A	N/A	N/A
Apple, Inc.	78-4F-43	N/A	GREEN	N/A
N/A	E6-AB-E6	GREEN	N/A	N/A
Hon Hai Precision Ind. Co.,Ltd.	0C-84-DC	N/A	N/A	N/A
Microsoft Corporation	BC-83-85	N/A	YELLOW	N/A
Intel Corporate	34-02-86	N/A	N/A	N/A
N/A	5C-00-3C	N/A	N/A	N/A
Universal Global Scientific Industrial Co., Ltd.	E0-2A-82	N/A	N/A	YELLOW
N/A	37-25-86	GREEN	N/A	N/A
N/A	4B-01-21	N/A	N/A	GREEN
N/A	62-A0-F1	N/A	N/A	N/A
N/A	68-60-22	N/A	N/A	GREEN
N/A	13-07-8A	N/A	N/A	N/A
HUAWEI TECHNOLOGIES CO.,LTD	84-BE-52	N/A	N/A	N/A
N/A	40-BC-5B	N/A	N/A	N/A
N/A	65-28-60	GREEN	N/A	N/A
Liteon Technology Corporation	9C-B7-0D	N/A	N/A	N/A
N/A	77-A8-D5	N/A	N/A	N/A
Apple, Inc.	18-EE-69	N/A	GREEN	N/A
N/A	54-EB-00	N/A	N/A	N/A
N/A	7C-6F-59	N/A	N/A	N/A
N/A	47-EB-79	N/A	N/A	N/A
N/A	45-A4-0E	N/A	N/A	N/A
Apple, Inc.	F4-0F-24	GREEN	N/A	N/A
N/A	7D-E1-B5	N/A	N/A	N/A
Bose Corporation	04-52-C7	N/A	N/A	N/A
Apple, Inc.	98-01-A7	N/A	N/A	N/A
N/A	63-5C-9F	N/A	GREEN	N/A
Apple, Inc.	78-4F-43	N/A	GREEN	N/A
N/A	6E-3F-A8	GREEN	N/A	N/A
N/A	63-56-AD	N/A	GREEN	N/A
Apple, Inc.	8C-85-90	N/A	GREEN	N/A
HUAWEI TECHNOLOGIES CO.,LTD	9C-B2-B2	N/A	N/A	N/A
Apple, Inc.	C4-B3-01	N/A	N/A	GREEN
N/A	73-4B-4E	N/A	N/A	N/A
Apple, Inc.	E4-CE-8F	GREEN	N/A	N/A

Table 10: Scan results, part 1

Brand	Prefix	02-10-2017	09-10-2017	11-10-2017
Intel Corporate	FC-F8-AE	YELLOW	YELLOW	N/A
N/A	D5-44-81	N/A	N/A	N/A
Xiaomi Communications Co Ltd	4C-49-E3	RED	N/A	N/A
N/A	7C-6C-67	N/A	N/A	N/A
N/A	55-96-54	N/A	N/A	GREEN
Apple, Inc.	08-6D-41	N/A	N/A	GREEN
N/A	6C-EA-F7	N/A	N/A	N/A
Apple, Inc.	C4-B3-01	GREEN	N/A	N/A
N/A	44-91-19	N/A	N/A	N/A
N/A	67-86-FF	N/A	N/A	N/A
N/A	68-8D-82	N/A	N/A	N/A
N/A	61-D0-64	N/A	N/A	N/A
N/A	50-B8-A8	N/A	N/A	GREEN
N/A	4A-5C-8C	N/A	N/A	GREEN
N/A	65-DD-20	N/A	N/A	N/A
Logitech, Inc	88-C6-26	N/A	N/A	N/A
N/A	55-4E-09	N/A	N/A	N/A
N/A	47-90-6D	GREEN	N/A	N/A
N/A	4D-9F-B5	N/A	N/A	GREEN
N/A	6D-17-BC	N/A	N/A	N/A
N/A	78-2D-14	N/A	N/A	N/A
N/A	69-EF-F5	GREEN	N/A	N/A
N/A	42-8A-E0	N/A	GREEN	GREEN
Liteon Technology Corporation	30-52-CB	N/A	N/A	N/A
Apple, Inc.	04-69-F8	GREEN	N/A	N/A
N/A	C6-39-5B	N/A	N/A	N/A
Intel Corporate	C4-85-08	N/A	N/A	N/A
Hon Hai Precision Ind. Co.,Ltd.	14-2D-27	N/A	N/A	N/A
N/A	63-3D-86	N/A	N/A	GREEN
Liteon Technology Corporation	9C-B7-0D	N/A	N/A	N/A
N/A	62-C2-23	N/A	N/A	N/A
N/A	62-C2-23	N/A	N/A	N/A
N/A	6B-9E-F1	N/A	N/A	N/A
Apple, Inc.	28-CF-DA	N/A	N/A	GREEN
Xiaomi Communications Co Ltd	D4-97-0B	N/A	RED	RED
Liteon Technology Corporation	74-DF-BF	N/A	YELLOW	N/A
N/A	56-B1-0D	N/A	N/A	N/A
MEIZU Technology Co., Ltd.	68-3E-34	N/A	N/A	N/A
N/A	E3-6E-6A	N/A	N/A	N/A
Intel Corporate	A0-88-69	N/A	N/A	N/A
N/A	66-9A-73	N/A	N/A	N/A
Apple, Inc.	B8-E8-56	N/A	GREEN	N/A
N/A	78-6A-0C	N/A	N/A	N/A

Table 10: Scan results, part 1

Brand	Prefix	02-10-2017	09-10-2017	11-10-2017
N/A	73-82-C9	N/A	N/A	N/A
OnePlus Technology (Shenzhen) Co., Ltd	94-65-2D	N/A	N/A	RED
Xiaomi Communications Co Ltd	74-23-44	RED	N/A	N/A
N/A	65-2C-8C	N/A	N/A	N/A
N/A	74-27-F4	N/A	GREEN	N/A
N/A	69-37-9B	N/A	N/A	N/A
OnePlus Tech (Shenzhen) Ltd	C0-EE-FB	N/A	N/A	N/A
OnePlus Technology (Shenzhen) Co., Ltd	94-65-2D	N/A	N/A	N/A
N/A	E6-E7-B6	N/A	N/A	GREEN
Apple, Inc.	B8-8D-12	N/A	N/A	N/A
N/A	75-10-1B	N/A	N/A	N/A
N/A	C6-3E-DB	N/A	N/A	N/A
N/A	5C-67-9E	N/A	N/A	N/A
Apple, Inc.	B8-E8-56	N/A	N/A	GREEN
N/A	7F-C0-98	N/A	N/A	N/A
N/A	77-C0-20	N/A	N/A	N/A
N/A	70-0F-4B	N/A	N/A	N/A
OnePlus Technology (Shenzhen) Co., Ltd	94-65-2D	N/A	N/A	N/A
N/A	74-51-20	N/A	N/A	N/A
Apple, Inc.	18-65-90	N/A	N/A	N/A
SHENZHEN RF-LINK TECHNOLOGY CO.,LTD.	C0-21-0D	N/A	N/A	N/A
N/A	64-31-27	N/A	N/A	N/A
N/A	46-FC-5F	N/A	N/A	GREEN
N/A	75-89-98	N/A	N/A	N/A
N/A	47-B3-77	N/A	N/A	N/A
N/A	79-F7-09	N/A	N/A	N/A
N/A	49-C3-BF	N/A	N/A	N/A
N/A	40-12-B2	N/A	N/A	N/A
N/A	7E-94-CF	N/A	N/A	N/A
N/A	63-05-94	GREEN	N/A	N/A
Universal Global Scientific Industrial Co., Ltd.	E0-2A-82	N/A	N/A	YELLOW
N/A	68-83-AF	N/A	N/A	N/A
N/A	66-74-B6	N/A	N/A	N/A
N/A	5A-05-F7	N/A	N/A	N/A
N/A	6D-FC-2F	N/A	N/A	N/A
N/A	66-B8-C3	N/A	N/A	N/A
N/A	2C-BD-D4	N/A	N/A	N/A
N/A	5B-80-7F	N/A	N/A	GREEN
N/A	4B-CE-E4	N/A	N/A	N/A
N/A	6F-B6-9A	N/A	N/A	N/A
Intel Corporate	68-07-15	GREEN	N/A	YELLOW
N/A	7A-74-0D	N/A	N/A	GREEN
Apple, Inc.	80-E6-50	GREEN	N/A	N/A

Table 10: Scan results, part 1

Brand	Prefix	02-10-2017	09-10-2017	11-10-2017
N/A	62-79-76	N/A	N/A	N/A
N/A	5E-1F-74	GREEN	N/A	N/A
Apple, Inc.	14-10-9F	N/A	N/A	GREEN
N/A	5B-1F-44	N/A	N/A	N/A
N/A	43-12-17	N/A	N/A	N/A
Hon Hai Precision Ind. Co.,Ltd.	A8-6B-AD	N/A	N/A	N/A
N/A	60-EC-44	N/A	N/A	GREEN
N/A	69-9D-2F	N/A	GREEN	N/A
Intel Corporate	E4-A4-71	N/A	N/A	N/A
N/A	4C-DF-7C	N/A	N/A	GREEN
N/A	6B-A5-B6	N/A	GREEN	N/A
Apple, Inc.	A8-66-7F	N/A	N/A	N/A
N/A	76-4F-91	N/A	GREEN	N/A
N/A	76-64-0B	N/A	N/A	N/A
N/A	45-50-E8	N/A	N/A	N/A
N/A	40-C8-C9	N/A	GREEN	N/A
Intel Corporate	AC-ED-5C	N/A	N/A	YELLOW
Apple, Inc.	C4-B3-01	N/A	N/A	N/A
N/A	60-F8-BE	N/A	N/A	N/A
N/A	5F-06-1F	N/A	N/A	N/A
N/A	F5-EB-F0	GREEN	N/A	N/A
N/A	6D-44-55	N/A	GREEN	N/A
N/A	61-8B-94	GREEN	N/A	N/A
N/A	72-29-91	N/A	N/A	N/A
Polar Electro Oy	00-22-D0	N/A	N/A	N/A
N/A	65-CD-B0	N/A	N/A	N/A
Apple, Inc.	34-36-3B	N/A	GREEN	N/A
Intel Corporate	34-E6-AD	N/A	N/A	YELLOW
Bose Corporation	04-52-C7	N/A	N/A	N/A
Liteon Technology Corporation	24-FD-52	N/A	N/A	N/A
N/A	70-6C-B4	N/A	N/A	N/A
N/A	E4-93-E5	N/A	N/A	N/A
N/A	D3-7F-81	GREEN	N/A	N/A
N/A	5D-8B-6C	N/A	N/A	N/A
N/A	5B-E1-20	N/A	N/A	GREEN
N/A	49-32-9E	N/A	N/A	N/A
N/A	40-84-E0	N/A	N/A	GREEN
N/A	7C-B5-51	N/A	N/A	GREEN
N/A	4E-EB-B8	N/A	N/A	GREEN
N/A	66-B1-18	N/A	GREEN	N/A
N/A	7E-94-53	N/A	N/A	N/A
N/A	60-AF-AF	N/A	N/A	N/A
N/A	40-CB-C9	N/A	GREEN	N/A

Table 10: Scan results, part 1

Brand	Prefix	02-10-2017	09-10-2017	11-10-2017
N/A	FC-15-FB	N/A	GREEN	N/A
N/A	4F-F8-CD	N/A	N/A	GREEN
N/A	52-58-A3	N/A	N/A	N/A
N/A	66-0A-9D	N/A	N/A	N/A
N/A	79-32-17	N/A	GREEN	N/A
N/A	3F-53-5B	N/A	N/A	N/A
N/A	50-37-DC	N/A	N/A	N/A
N/A	59-06-51	N/A	N/A	GREEN
N/A	6E-B2-9C	N/A	N/A	N/A
N/A	45-86-8B	N/A	GREEN	N/A
N/A	7D-63-33	GREEN	N/A	N/A
N/A	6B-A3-E7	N/A	N/A	N/A
N/A	5A-61-B2	N/A	GREEN	N/A
N/A	56-53-F4	N/A	N/A	N/A
N/A	7F-62-2A	N/A	GREEN	N/A
Apple, Inc.	8C-85-90	GREEN	N/A	N/A
N/A	71-E6-BF	N/A	N/A	GREEN
Apple, Inc.	DC-A9-04	GREEN	N/A	N/A
N/A	4D-17-50	N/A	N/A	N/A
N/A	6C-5B-B1	N/A	N/A	N/A
N/A	5B-9B-A6	N/A	N/A	N/A
N/A	53-FB-F9	N/A	N/A	GREEN
N/A	51-ED-81	N/A	N/A	N/A
Liteon Technology Corporation	C8-FF-28	N/A	YELLOW	N/A
N/A	6F-63-60	N/A	N/A	N/A
N/A	47-B0-D8	GREEN	N/A	N/A
N/A	6C-CF-60	N/A	N/A	N/A
N/A	5B-33-FE	N/A	N/A	GREEN
Intel Corporate	E4-A4-71	N/A	YELLOW	N/A
Intel Corporate	F8-16-54	N/A	N/A	N/A
N/A	60-09-44	GREEN	N/A	N/A
N/A	5D-1F-3C	N/A	N/A	GREEN
N/A	67-E4-00	N/A	N/A	N/A
Apple, Inc.	2C-F0-EE	N/A	N/A	N/A
Liteon Technology Corporation	48-D2-24	N/A	N/A	YELLOW
N/A	5F-4A-6F	GREEN	N/A	N/A
N/A	78-33-B8	N/A	N/A	N/A
N/A	65-64-41	N/A	N/A	N/A
N/A	44-8F-EE	N/A	N/A	N/A
N/A	4D-CD-4A	N/A	N/A	N/A
N/A	6C-4B-1F	N/A	N/A	N/A
N/A	4F-0A-2B	N/A	N/A	N/A
N/A	72-B5-C7	N/A	N/A	N/A

Table 10: Scan results, part 1

Brand	Prefix	02-10-2017	09-10-2017	11-10-2017
Apple, Inc.	78-CA-39	GREEN	GREEN	N/A
N/A	66-16-43	N/A	N/A	N/A
N/A	5F-EB-F0	GREEN	N/A	N/A
Apple, Inc.	8C-85-90	N/A	N/A	N/A
N/A	20-EF-4C	N/A	GREEN	N/A
N/A	57-36-8E	N/A	N/A	N/A
Intel Corporate	E0-94-67	N/A	N/A	N/A
Intel Corporate	34-02-86	YELLOW	N/A	N/A
Apple, Inc.	A4-5E-60	N/A	N/A	N/A
Intel Corporate	B4-6D-83	YELLOW	YELLOW	N/A
N/A	6F-00-1B	N/A	N/A	N/A
Apple, Inc.	8C-85-90	N/A	N/A	N/A
Liteon Technology Corporation	28-E3-47	N/A	N/A	YELLOW
N/A	57-4F-E5	N/A	N/A	GREEN
Intel Corporate	B0-35-9F	YELLOW	N/A	N/A
N/A	73-85-7F	N/A	GREEN	N/A
N/A	52-F1-CC	N/A	N/A	N/A
Intel Corporate	E4-B3-18	N/A	YELLOW	N/A
N/A	4F-A4-76	N/A	N/A	N/A
N/A	71-AA-03	N/A	GREEN	N/A

Table 11: Scan results, part 2

Brand	Prefix	12-10-2017	13-10-2017	16-10-2017	19-10-2017
N/A	53-13-0A	N/A	GREEN	N/A	N/A
N/A	40-59-EF	N/A	N/A	GREEN	N/A
N/A	FB-F8-AE	N/A	YELLOW	N/A	N/A
Apple, Inc.	8C-85-90	N/A	N/A	GREEN	N/A
Liteon Technology Corporation	30-10-B3	N/A	N/A	N/A	YELLOW
Apple, Inc.	F0-79-60	N/A	N/A	N/A	GREEN
N/A	53-20-0C	N/A	N/A	N/A	N/A
N/A	75-E4-F9	N/A	N/A	N/A	GREEN
N/A	41-E7-5A	N/A	N/A	N/A	N/A
N/A	68-0A-8A	N/A	GREEN	N/A	N/A
N/A	49-18-40	N/A	N/A	N/A	N/A
N/A	56-A2-AC	N/A	GREEN	N/A	N/A
N/A	4E-4C-29	N/A	N/A	N/A	N/A
N/A	78-50-6C	N/A	N/A	GREEN	N/A
N/A	60-DE-76	N/A	N/A	N/A	N/A
Apple, Inc.	AC-BC-32	N/A	GREEN	N/A	N/A
N/A	55-00-6E	N/A	N/A	N/A	N/A
N/A	50-55-B0	N/A	N/A	N/A	N/A
N/A	58-1D-F8	GREEN	N/A	N/A	N/A

Table 11: Scan results, part 2

Brand	Prefix	12-10-2017	13-10-2017	16-10-2017	19-10-2017
PEGATRON CORPORATION	60-02-92	N/A	YELLOW	N/A	N/A
N/A	FE-DD-4F	N/A	N/A	N/A	GREEN
N/A	58-C0-63	GREEN	N/A	N/A	N/A
Apple, Inc.	00-16-CB	N/A	N/A	N/A	N/A
N/A	74-88-9F	GREEN	N/A	N/A	N/A
Intel Corporate	D0-57-7B	N/A	YELLOW	N/A	N/A
N/A	57-A6-72	N/A	N/A	N/A	N/A
N/A	79-08-05	N/A	N/A	GREEN	N/A
N/A	47-BF-BC	N/A	N/A	N/A	N/A
N/A	7C-25-03	GREEN	N/A	N/A	N/A
Apple, Inc.	78-4F-43	GREEN	N/A	N/A	N/A
N/A	5F-17-AA	N/A	N/A	N/A	GREEN
Apple, Inc.	98-01-A7	N/A	N/A	GREEN	GREEN
N/A	49-A3-4B	N/A	N/A	N/A	N/A
N/A	56-00-42	GREEN	N/A	N/A	N/A
N/A	62-61-49	N/A	N/A	GREEN	N/A
N/A	56-B9-E4	GREEN	N/A	N/A	N/A
N/A	79-F2-6B	N/A	N/A	GREEN	N/A
Intel Corporate	00-21-5C	N/A	N/A	N/A	N/A
N/A	49-9B-9A	N/A	N/A	N/A	GREEN
Xiaomi Communications Co Ltd	10-2A-B3	N/A	N/A	N/A	RED
Wistron Neweb Corporation	60-02-B4	YELLOW	N/A	N/A	N/A
Apple, Inc.	B8-E8-56	N/A	N/A	N/A	GREEN
N/A	48-69-A3	N/A	N/A	N/A	N/A
N/A	41-86-3E	GREEN	N/A	N/A	N/A
N/A	EC-7A-B8	N/A	N/A	N/A	GREEN
N/A	4B-93-DB	N/A	N/A	GREEN	N/A
N/A	60-1A-8D	N/A	N/A	N/A	GREEN
Huami Information Technology Co.,Ltd.	88-0F-10	N/A	N/A	N/A	N/A
N/A	49-8D-12	GREEN	N/A	N/A	N/A
N/A	55-B4-5C	N/A	N/A	N/A	GREEN
N/A	59-0F-60	N/A	N/A	N/A	N/A
Apple, Inc.	60-F8-1D	N/A	GREEN	N/A	GREEN
N/A	FE-08-55	N/A	N/A	N/A	N/A
N/A	4F-3E-35	N/A	N/A	N/A	N/A
N/A	77-C0-29	N/A	N/A	GREEN	N/A
Apple, Inc.	64-B9-E8	N/A	GREEN	GREEN	N/A
N/A	56-A4-01	N/A	GREEN	N/A	N/A
N/A	7A-94-90	N/A	N/A	N/A	N/A
N/A	88-0F-19	N/A	N/A	N/A	N/A
N/A	7F-52-3A	N/A	N/A	GREEN	N/A
N/A	7B-53-9D	N/A	GREEN	N/A	N/A
N/A	45-4E-13	GREEN	GREEN	N/A	N/A

Table 11: Scan results, part 2

Brand	Prefix	12-10-2017	13-10-2017	16-10-2017	19-10-2017
N/A	61-B8-BE	N/A	N/A	GREEN	N/A
N/A	4E-C1-14	N/A	N/A	N/A	GREEN
Apple, Inc.	10-40-F3	N/A	N/A	N/A	N/A
N/A	65-03-64	N/A	N/A	N/A	N/A
Apple, Inc.	78-4F-43	N/A	GREEN	N/A	N/A
N/A	CE-DE-D5	N/A	N/A	N/A	GREEN
N/A	6A-A8-8A	N/A	N/A	N/A	N/A
N/A	00-EC-0A	N/A	N/A	N/A	N/A
N/A	75-58-B0	N/A	N/A	N/A	GREEN
N/A	5F-DA-99	N/A	N/A	N/A	N/A
Apple, Inc.	78-31-C1	GREEN	N/A	GREEN	N/A
Apple, Inc.	AC-BC-32	N/A	N/A	N/A	N/A
N/A	79-DD-AA	N/A	N/A	N/A	GREEN
Apple, Inc.	78-4F-43	GREEN	N/A	N/A	N/A
OnePlus Tech (Shenzhen) Ltd	C0-EE-FB	RED	N/A	N/A	N/A
N/A	55-51-EF	N/A	N/A	N/A	GREEN
N/A	6A-29-7D	N/A	N/A	N/A	N/A
N/A	5F-F7-C1	N/A	N/A	GREEN	N/A
N/A	5B-1A-C5	N/A	N/A	N/A	N/A
Apple, Inc.	B8-E8-56	N/A	N/A	N/A	N/A
N/A	C6-17-B6	GREEN	N/A	N/A	N/A
Apple, Inc.	6C-40-08	N/A	N/A	N/A	N/A
AzureWave Technology Inc.	24-0A-64	N/A	YELLOW	N/A	N/A
N/A	5E-0B-32	N/A	N/A	N/A	GREEN
Apple, Inc.	D0-A6-37	N/A	N/A	GREEN	N/A
Microsoft Corporation	BC-83-85	N/A	YELLOW	N/A	N/A
N/A	69-B9-62	GREEN	N/A	N/A	N/A
N/A	79-29-30	N/A	N/A	N/A	GREEN
Apple, Inc.	78-4F-43	N/A	N/A	N/A	N/A
N/A	E6-AB-E6	N/A	N/A	N/A	N/A
Hon Hai Precision Ind. Co.,Ltd.	0C-84-DC	YELLOW	N/A	YELLOW	N/A
Microsoft Corporation	BC-83-85	N/A	N/A	N/A	N/A
Intel Corporate	34-02-86	N/A	N/A	N/A	YELLOW
N/A	5C-00-3C	N/A	N/A	GREEN	N/A
Universal Global Scientific Industrial Co., Ltd.	E0-2A-82	N/A	N/A	N/A	N/A
N/A	37-25-86	N/A	N/A	N/A	N/A
N/A	4B-01-21	N/A	N/A	N/A	N/A
N/A	62-A0-F1	N/A	N/A	GREEN	N/A
N/A	68-60-22	N/A	N/A	N/A	N/A
N/A	13-07-8A	GREEN	N/A	N/A	N/A
HUAWEI TECHNOLOGIES CO.,LTD	84-BE-52	N/A	N/A	RED	N/A
N/A	40-BC-5B	N/A	N/A	N/A	GREEN
N/A	65-28-60	N/A	N/A	N/A	N/A

Table 11: Scan results, part 2

Brand	Prefix	12-10-2017	13-10-2017	16-10-2017	19-10-2017
Liteon Technology Corporation	9C-B7-0D	YELLOW	N/A	N/A	N/A
N/A	77-A8-D5	N/A	N/A	N/A	GREEN
Apple, Inc.	18-EE-69	N/A	N/A	GREEN	GREEN
N/A	54-EB-00	N/A	N/A	N/A	GREEN
N/A	7C-6F-59	N/A	GREEN	N/A	N/A
N/A	47-EB-79	N/A	GREEN	N/A	N/A
N/A	45-A4-0E	N/A	GREEN	N/A	N/A
Apple, Inc.	F4-0F-24	N/A	N/A	N/A	N/A
N/A	7D-E1-B5	N/A	GREEN	N/A	N/A
Bose Corporation	04-52-C7	N/A	N/A	GREEN	N/A
Apple, Inc.	98-01-A7	N/A	N/A	N/A	GREEN
N/A	63-5C-9F	N/A	N/A	N/A	N/A
Apple, Inc.	78-4F-43	N/A	N/A	GREEN	N/A
N/A	6E-3F-A8	N/A	N/A	N/A	N/A
N/A	63-56-AD	N/A	N/A	N/A	N/A
Apple, Inc.	8C-85-90	N/A	N/A	N/A	N/A
HUAWEI TECHNOLOGIES CO.,LTD	9C-B2-B2	RED	N/A	N/A	N/A
Apple, Inc.	C4-B3-01	N/A	N/A	N/A	N/A
N/A	73-4B-4E	N/A	N/A	GREEN	N/A
Apple, Inc.	E4-CE-8F	N/A	N/A	N/A	N/A
Intel Corporate	FC-F8-AE	N/A	YELLOW	N/A	N/A
N/A	D5-44-81	GREEN	N/A	N/A	N/A
Xiaomi Communications Co Ltd	4C-49-E3	N/A	N/A	N/A	N/A
N/A	7C-6C-67	N/A	N/A	N/A	GREEN
N/A	55-96-54	N/A	N/A	N/A	N/A
Apple, Inc.	08-6D-41	N/A	N/A	N/A	N/A
N/A	6C-EA-F7	GREEN	N/A	N/A	N/A
Apple, Inc.	C4-B3-01	N/A	GREEN	N/A	N/A
N/A	44-91-19	N/A	GREEN	N/A	N/A
N/A	67-86-FF	N/A	GREEN	N/A	N/A
N/A	68-8D-82	N/A	N/A	GREEN	N/A
N/A	61-D0-64	N/A	N/A	GREEN	N/A
N/A	50-B8-A8	N/A	N/A	N/A	N/A
N/A	4A-5C-8C	N/A	N/A	N/A	N/A
N/A	65-DD-20	GREEN	N/A	N/A	N/A
Logitech, Inc	88-C6-26	N/A	N/A	N/A	GREEN
N/A	55-4E-09	N/A	N/A	N/A	GREEN
N/A	47-90-6D	N/A	N/A	N/A	N/A
N/A	4D-9F-B5	GREEN	N/A	GREEN	N/A
N/A	6D-17-BC	N/A	N/A	N/A	GREEN
N/A	78-2D-14	N/A	N/A	GREEN	N/A
N/A	69-EF-F5	N/A	N/A	N/A	N/A
N/A	42-8A-E0	N/A	N/A	N/A	N/A

Table 11: Scan results, part 2

Brand	Prefix	12-10-2017	13-10-2017	16-10-2017	19-10-2017
Liteon Technology Corporation	30-52-CB	YELLOW	N/A	N/A	N/A
Apple, Inc.	04-69-F8	N/A	N/A	N/A	N/A
N/A	C6-39-5B	N/A	N/A	N/A	GREEN
Intel Corporate	C4-85-08	N/A	N/A	N/A	YELLOW
Hon Hai Precision Ind. Co.,Ltd.	14-2D-27	N/A	N/A	N/A	YELLOW
N/A	63-3D-86	N/A	N/A	N/A	N/A
Liteon Technology Corporation	9C-B7-0D	YELLOW	N/A	N/A	N/A
N/A	62-C2-23	GREEN	N/A	N/A	N/A
N/A	62-C2-23	GREEN	N/A	N/A	N/A
N/A	6B-9E-F1	GREEN	N/A	N/A	N/A
Apple, Inc.	28-CF-DA	N/A	N/A	N/A	N/A
Xiaomi Communications Co Ltd	D4-97-0B	N/A	N/A	N/A	N/A
Liteon Technology Corporation	74-DF-BF	N/A	N/A	N/A	N/A
N/A	56-B1-0D	GREEN	N/A	N/A	N/A
MEIZU Technology Co., Ltd.	68-3E-34	N/A	N/A	RED	N/A
N/A	E3-6E-6A	N/A	N/A	N/A	GREEN
Intel Corporate	A0-88-69	N/A	YELLOW	N/A	YELLOW
N/A	66-9A-73	GREEN	N/A	N/A	N/A
Apple, Inc.	B8-E8-56	N/A	N/A	N/A	N/A
N/A	78-6A-0C	N/A	N/A	N/A	GREEN
N/A	73-82-C9	N/A	GREEN	N/A	N/A
OnePlus Technology (Shenzhen) Co., Ltd	94-65-2D	N/A	N/A	N/A	N/A
Xiaomi Communications Co Ltd	74-23-44	N/A	N/A	N/A	N/A
N/A	65-2C-8C	N/A	N/A	N/A	GREEN
N/A	74-27-F4	N/A	N/A	N/A	N/A
N/A	69-37-9B	GREEN	N/A	N/A	N/A
OnePlus Tech (Shenzhen) Ltd	C0-EE-FB	RED	RED	N/A	N/A
OnePlus Technology (Shenzhen) Co., Ltd	94-65-2D	RED	N/A	N/A	N/A
N/A	E6-E7-B6	N/A	N/A	N/A	N/A
Apple, Inc.	B8-8D-12	N/A	N/A	GREEN	N/A
N/A	75-10-1B	N/A	N/A	GREEN	N/A
N/A	C6-3E-DB	N/A	GREEN	N/A	N/A
N/A	5C-67-9E	GREEN	N/A	N/A	N/A
Apple, Inc.	B8-E8-56	N/A	N/A	GREEN	N/A
N/A	7F-C0-98	N/A	N/A	N/A	GREEN
N/A	77-C0-20	N/A	N/A	GREEN	N/A
N/A	70-0F-4B	N/A	N/A	N/A	GREEN
OnePlus Technology (Shenzhen) Co., Ltd	94-65-2D	N/A	RED	N/A	N/A
N/A	74-51-20	N/A	N/A	N/A	GREEN
Apple, Inc.	18-65-90	N/A	N/A	N/A	GREEN
SHENZHEN RF-LINK TECHNOLOGY CO.,LTD.	C0-21-0D	N/A	N/A	YELLOW	N/A
N/A	64-31-27	N/A	N/A	N/A	GREEN
N/A	46-FC-5F	N/A	N/A	N/A	N/A

Table 11: Scan results, part 2

Brand	Prefix	12-10-2017	13-10-2017	16-10-2017	19-10-2017
N/A	75-89-98	N/A	GREEN	N/A	N/A
N/A	47-B3-77	GREEN	N/A	N/A	N/A
N/A	79-F7-09	GREEN	N/A	N/A	N/A
N/A	49-C3-BF	N/A	GREEN	GREEN	N/A
N/A	40-12-B2	N/A	N/A	N/A	GREEN
N/A	7E-94-CF	N/A	GREEN	N/A	N/A
N/A	63-05-94	N/A	N/A	N/A	N/A
Universal Global Scientific Industrial Co., Ltd.	E0-2A-82	N/A	N/A	N/A	N/A
N/A	68-83-AF	N/A	N/A	N/A	GREEN
N/A	66-74-B6	GREEN	N/A	N/A	N/A
N/A	5A-05-F7	N/A	N/A	GREEN	N/A
N/A	6D-FC-2F	N/A	N/A	GREEN	N/A
N/A	66-B8-C3	N/A	N/A	GREEN	N/A
N/A	2C-BD-D4	N/A	GREEN	N/A	N/A
N/A	5B-80-7F	N/A	N/A	N/A	N/A
N/A	4B-CE-E4	GREEN	N/A	N/A	N/A
N/A	6F-B6-9A	N/A	N/A	N/A	GREEN
Intel Corporate	68-07-15	N/A	N/A	N/A	N/A
N/A	7A-74-0D	N/A	N/A	N/A	N/A
Apple, Inc.	80-E6-50	N/A	N/A	N/A	N/A
N/A	62-79-76	N/A	N/A	N/A	GREEN
N/A	5E-1F-74	N/A	N/A	N/A	N/A
Apple, Inc.	14-10-9F	N/A	N/A	N/A	N/A
N/A	5B-1F-44	N/A	N/A	GREEN	N/A
N/A	43-12-17	N/A	N/A	GREEN	N/A
Hon Hai Precision Ind. Co.,Ltd.	A8-6B-AD	N/A	N/A	N/A	YELLOW
N/A	60-EC-44	N/A	N/A	N/A	N/A
N/A	69-9D-2F	N/A	N/A	N/A	N/A
Intel Corporate	E4-A4-71	N/A	N/A	N/A	YELLOW
N/A	4C-DF-7C	N/A	N/A	N/A	N/A
N/A	6B-A5-B6	N/A	N/A	N/A	N/A
Apple, Inc.	A8-66-7F	GREEN	N/A	N/A	N/A
N/A	76-4F-91	N/A	N/A	N/A	N/A
N/A	76-64-0B	N/A	N/A	GREEN	N/A
N/A	45-50-E8	N/A	GREEN	N/A	N/A
N/A	40-C8-C9	N/A	N/A	N/A	N/A
Intel Corporate	AC-ED-5C	N/A	N/A	N/A	N/A
Apple, Inc.	C4-B3-01	N/A	N/A	N/A	GREEN
N/A	60-F8-BE	GREEN	N/A	N/A	N/A
N/A	5F-06-1F	N/A	N/A	N/A	GREEN
N/A	F5-EB-F0	N/A	N/A	N/A	N/A
N/A	6D-44-55	N/A	N/A	N/A	N/A
N/A	61-8B-94	N/A	N/A	N/A	N/A

Table 11: Scan results, part 2

Brand	Prefix	12-10-2017	13-10-2017	16-10-2017	19-10-2017
N/A	72-29-91	N/A	N/A	GREEN	N/A
Polar Electro Oy	00-22-D0	N/A	N/A	N/A	GREEN
N/A	65-CD-B0	N/A	N/A	N/A	GREEN
Apple, Inc.	34-36-3B	N/A	N/A	N/A	N/A
Intel Corporate	34-E6-AD	N/A	N/A	N/A	N/A
Bose Corporation	04-52-C7	N/A	GREEN	N/A	N/A
Liteon Technology Corporation	24-FD-52	YELLOW	N/A	YELLOW	N/A
N/A	70-6C-B4	N/A	N/A	N/A	GREEN
N/A	E4-93-E5	N/A	N/A	GREEN	N/A
N/A	D3-7F-81	N/A	N/A	N/A	N/A
N/A	5D-8B-6C	N/A	GREEN	N/A	N/A
N/A	5B-E1-20	N/A	N/A	N/A	N/A
N/A	49-32-9E	N/A	N/A	GREEN	N/A
N/A	40-84-E0	N/A	N/A	N/A	N/A
N/A	7C-B5-51	N/A	N/A	N/A	N/A
N/A	4E-EB-B8	N/A	N/A	N/A	N/A
N/A	66-B1-18	N/A	N/A	N/A	N/A
N/A	7E-94-53	GREEN	N/A	N/A	N/A
N/A	60-AF-AF	GREEN	N/A	N/A	N/A
N/A	40-CB-C9	N/A	N/A	N/A	N/A
N/A	FC-15-FB	N/A	N/A	N/A	N/A
N/A	4F-F8-CD	N/A	N/A	N/A	N/A
N/A	52-58-A3	N/A	GREEN	N/A	N/A
N/A	66-0A-9D	GREEN	N/A	N/A	N/A
N/A	79-32-17	N/A	N/A	N/A	N/A
N/A	3F-53-5B	N/A	N/A	GREEN	N/A
N/A	50-37-DC	GREEN	N/A	N/A	N/A
N/A	59-06-51	N/A	N/A	N/A	N/A
N/A	6E-B2-9C	N/A	N/A	GREEN	N/A
N/A	45-86-8B	N/A	N/A	N/A	N/A
N/A	7D-63-33	N/A	N/A	N/A	N/A
N/A	6B-A3-E7	N/A	N/A	N/A	GREEN
N/A	5A-61-B2	N/A	N/A	N/A	N/A
N/A	56-53-F4	GREEN	N/A	N/A	N/A
N/A	7F-62-2A	N/A	N/A	N/A	N/A
Apple, Inc.	8C-85-90	N/A	N/A	N/A	N/A
N/A	71-E6-BF	N/A	N/A	N/A	N/A
Apple, Inc.	DC-A9-04	N/A	N/A	N/A	N/A
N/A	4D-17-50	N/A	GREEN	N/A	N/A
N/A	6C-5B-B1	N/A	N/A	N/A	GREEN
N/A	5B-9B-A6	N/A	GREEN	N/A	N/A
N/A	53-FB-F9	N/A	N/A	N/A	N/A
N/A	51-ED-81	GREEN	N/A	N/A	N/A

Table 11: Scan results, part 2

Brand	Prefix	12-10-2017	13-10-2017	16-10-2017	19-10-2017
Liteon Technology Corporation	C8-FF-28	N/A	N/A	N/A	N/A
N/A	6F-63-60	N/A	GREEN	N/A	N/A
N/A	47-B0-D8	N/A	N/A	N/A	N/A
N/A	6C-CF-60	N/A	N/A	GREEN	N/A
N/A	5B-33-FE	N/A	N/A	N/A	N/A
Intel Corporate	E4-A4-71	N/A	N/A	N/A	N/A
Intel Corporate	F8-16-54	YELLOW	N/A	YELLOW	N/A
N/A	60-09-44	N/A	N/A	N/A	N/A
N/A	5D-1F-3C	N/A	N/A	N/A	N/A
N/A	67-E4-00	N/A	GREEN	N/A	N/A
Apple, Inc.	2C-F0-EE	N/A	N/A	N/A	GREEN
Liteon Technology Corporation	48-D2-24	N/A	N/A	N/A	N/A
N/A	5F-4A-6F	N/A	N/A	N/A	N/A
N/A	78-33-B8	N/A	GREEN	N/A	N/A
N/A	65-64-41	N/A	N/A	N/A	GREEN
N/A	44-8F-EE	GREEN	N/A	N/A	N/A
N/A	4D-CD-4A	N/A	GREEN	GREEN	GREEN
N/A	6C-4B-1F	N/A	N/A	GREEN	N/A
N/A	4F-0A-2B	GREEN	N/A	N/A	N/A
N/A	72-B5-C7	N/A	N/A	GREEN	N/A
Apple, Inc.	78-CA-39	GREEN	GREEN	GREEN	N/A
N/A	66-16-43	N/A	N/A	N/A	GREEN
N/A	5F-EB-F0	N/A	N/A	N/A	N/A
Apple, Inc.	8C-85-90	N/A	GREEN	N/A	N/A
N/A	20-EF-4C	N/A	N/A	N/A	N/A
N/A	57-36-8E	GREEN	N/A	N/A	N/A
Intel Corporate	E0-94-67	N/A	YELLOW	N/A	N/A
Intel Corporate	34-02-86	N/A	N/A	N/A	N/A
Apple, Inc.	A4-5E-60	N/A	N/A	N/A	GREEN
Intel Corporate	B4-6D-83	N/A	N/A	N/A	N/A
N/A	6F-00-1B	N/A	N/A	GREEN	N/A
Apple, Inc.	8C-85-90	N/A	GREEN	GREEN	GREEN
Liteon Technology Corporation	28-E3-47	N/A	N/A	N/A	N/A
N/A	57-4F-E5	N/A	N/A	N/A	N/A
Intel Corporate	B0-35-9F	N/A	N/A	N/A	N/A
N/A	73-85-7F	N/A	N/A	N/A	N/A
N/A	52-F1-CC	GREEN	N/A	N/A	N/A
Intel Corporate	E4-B3-18	N/A	N/A	N/A	N/A
N/A	4F-A4-76	N/A	N/A	N/A	GREEN
N/A	71-AA-03	N/A	N/A	N/A	N/A

9.2 Scan result merge script

Listing 1: Python script to merge the scan results and append extra data

```
#!/usr/bin/env python

from argparse import ArgumentParser
from os import path
import pandas as pd
import csv, re, urllib2, json, netaddr, sys

class MacLookup(object):
    def __init__(self, csv_src, csv_dst, csv_num):
        self.csv_src = csv_src
        self.csv_dst = csv_dst
        self.numeral_output = csv_num
        self.csv_fields = ["MAC", "Brand", "Prefix"]
        self.vendor_details = {}
        self.state_info = {}
        self.state_rating = {"N/A": 0, "GREEN": 1, "YELLOW": 2, "RED": 3}
        self.get_mac()

    def get_mac(self):
        macs = self.csv_reader()
        r = re.compile(r'(?:[0-9a-fA-F]:?){12}')
        for mac, state in macs.items():
            if r.match(mac):
                vendor_detail = self.get_vendor_details(mac)
                mac_dialect = vendor_detail[0]
                company = vendor_detail[1]
                mac_prefix = vendor_detail[2]
                try:
                    self.vendor_details[mac_dialect] = (company, mac_prefix, state)
                except KeyError as e:
                    print("%s triggered a key error" % (e))
                    sys.exit(1)
            elif mac == self.csv_fields[0]:
                continue
            else:
                print("%s is not a valid MAC" % (mac))
        self.state_info = {key: value[2] for key, value in self.vendor_details.items()}
        self.csv_writer()

    def csv_reader(self):
        macs = {}
        for source in self.csv_src:
            try:
                date = path.splitext(path.basename(source))[0]
```

```

        self.csv_fields.append(date)
        with open(source, "rb") as f:
            reader = csv.reader(f)
            for row in reader:
                mac = row[0]
                state = row[1].upper()
                if self.numeral_output:
                    state = self.state_rating[state]
                if not mac in macs.keys():
                    macs[mac] = ["%s|%s" % (date, state)]
                else:
                    macs[mac].append("%s|%s" % (date, state))
    except IOError as e:
        print("Error_code_%d_while_reading_%s:_%s" %
              (e.errno, source, e.strerror))
        sys.exit(1)
    return(macs)

def csv_writer(self):
    try:
        with open(self.csv_dst, "wb") as f:
            writer = csv.DictWriter(f, fieldnames=self.csv_fields)
            writer.writeheader()
            for key, value in self.vendor_details.items():
                mac = key
                brand = value[0]
                prefix = value[1]
                writer.writerow({"MAC": mac, "Brand": brand, "Prefix": prefix})
    except IOError as e:
        print("Error_code_%d_while_writing_%s:_%s" %
              (e.errno, self.csv_dst, e.strerror))
        sys.exit(1)
    self.csv_update_columns()

def csv_update_columns(self):
    df = pd.read_csv(self.csv_dst, sep=",", keep_default_na=False)
    for key, value in self.state_info.items():
        for x in value:
            x = x.split("|")
            mac = key
            date = x[0]
            state = x[1]
            df.loc[df[self.csv_fields[0]] == mac, date] = state
    if self.numeral_output:
        df = df.replace(r'^$', "0", regex=True)
    else:

```

```

        df = df.replace(r'^$', 'N/A', regex=True)
    df.to_csv(self.csv_dst, sep=",", encoding="utf-8", index=False)

def get_vendor_details(self, mac):
    converted_mac = netaddr.EUI(mac)
    mac_prefix = "-".join(str(converted_mac).split("-")[:3])
    converted_mac.dialect = netaddr.mac_unix
    try:
        oui = converted_mac.oui
    except:
        company = self.get_vendor_details_online(converted_mac)
        return(str(converted_mac), str(company), mac_prefix)
    return(str(converted_mac), str(oui.registration().org), mac_prefix)

def get_vendor_details_online(self, mac):
    opener = urllib2.build_opener()
    opener.addheaders = [("User-Agent", "Mozilla/5.0")]
    response = opener.open("https://macvendors.co/api/%s/json" % (mac))
    vendor_detail = json.loads(response.read())
    if "result" in vendor_detail.keys():
        try:
            company = vendor_detail["result"]["company"]
            return(company)
        except KeyError:
            return("N/A")
    else:
        print("Something_went_wrong_in_get_vendor_details_online,_go_to_red_alert")
        sys.exit(1)

def argument_parser():
    parser = ArgumentParser(description="Mac_address_details_->_csv")
    parser.add_argument("-s", "--src", action="append", type=str,
        required=True, help="Provide_source_csv")
    parser.add_argument("-d", "--dst", type=str,
        required=True, help="Provide_destination_csv")
    parser.add_argument("-n", "--num", action="store_true",
        default=False, required=False, help="Insert_numerical_values_instead_of_strings")
    args = vars(parser.parse_args())
    return(args)

def main():
    args = argument_parser()
    sources = args["src"]
    destination = args["dst"]

```

```
numeral_output = args["num"]  
MacLookup(sources, destination, numeral_output)  
  
if __name__ == "__main__":  
    main()
```