Int. J. Inf. Secur. (2016) 15:455–473
DOI 10.1007/s10207-015-0310-0

REGULAR CONTRIBUTION

New facets of mobile botnet: architecture and evaluation

Marios Anagnostopoulos[1] · **Georgios Kambourakis**[1] · **Stefanos Gritzalis**[1]

Published online: 31 December 2015

Abstract It is without a doubt that botnets pose a growing threat to the Internet, with DDoS attacks of any kind carried out by botnets to be on the rise. Nowadays, botmasters rely on advanced Command and Control (C&C) infrastructures to achieve their goals and most importantly to remain undetected. This work introduces two novel botnet architectures that consist only of mobile devices and evaluates both their impact in terms of DNS amplification and TCP flooding attacks, and their cost pertaining to the maintenance of the C&C channel. The first one puts forward the idea of using a continually changing mobile HTTP proxy in front of the botherder, while the other capitalizes on DNS protocol as a covert channel for coordinating the botnet. That is, for the latter, the messages exchanged among the bots and the herder appear as legitimate DNS transactions. Also, a third architecture is described and assessed, which is basically an optimized variation of the first one. Namely, it utilizes a mixed layout where all the attacking bots are mobile, but the proxy machines are typical PCs not involved in the actual attack. For the DNS amplification attack, which is by nature more powerful, we report an amplification factor that fluctuates between 32.7 and 34.1. Also, regarding the imposed C&C cost, we assert that it is minimal (about 0.25 Mbps) per bot in the worst case happening momentarily when the bot learns about the parameters of the attack.

Keywords Mobile botnets · DNS amplification · Covert channel · Cyber security

✉ Marios Anagnostopoulos
managn@aegean.gr

1 Info-Sec-Lab Laboratory of Information and Communication Systems Security, Department of Information and Communication Systems Engineering, University of the Aegean, 83200 Samos, Greece

1 Introduction

A botnet can be considered as a network consisting of infected and compromised computers, called bots, zombies or slaves, which are controlled by an attacker, known as botmaster or botherder. A bot agent obeys every command received by its botmaster ordering it to initiate or terminate an attack. Botnets pose a serious threat to Internet, since they are capable of disrupting the normal operation of services, networks and systems at will of their botmaster. For instance, botnets could be used for launching distributed denial-of-service (DDoS) attacks, sending spam emails on a massive scale, identity theft, distributing malware or even copyrighted material, and so forth [1]. Usually, a device is turned into a bot client, by malware infection, for instance by a malicious software or by accessing an infected website. After that, the device participates to a network of bots waiting for commands. The bot takes action only whenever the botherder says so, through a covert Command and Control (C&C) channel, while the remaining time stays silent. In this respect, C&C enables a bot to acquire new instructions and malicious capabilities, as injected by a remote ill-motivated entity. In the literature, a variety of C&C topologies have been explored by botmasters with the dual aim to curtail network chatter and system failures and to cope with deployed defenses, hijacking attempts and legal shutdowns.

On the other hand, perhaps the most vital demand for maintaining control of the entire botnet is the ability for a bot to constantly stay in touch with C&C infrastructure. As discussed further down in Sect. 2.1, this requirement is especially true for botnets that hinge on centralized C&C. That is, a bot will not be able to receive new instructions if the C&C cannot be located, and continue to probe the vanished C&C in vain. In this direction, botmasters employ a number of techniques not only to minimize the probability of bots losing contact with their C&C infrastructure, but also to render their botnet more agile to hijacking and stoppage attempts. “Fluxing” seems to be the preferred technology to deal with the aforementioned issues, i.e., uninterrupted provision of C&C location resolution and failover resilience. Currently, *IP Flux* and *Domain Flux* are the two dominant ways of “Fluxing.” The first one involves the regular altering of IP address pertaining to a particular fully qualified domain name (FQDN). This potential is particularly fruitful for botnet operators because it enables them to associate multiple IP addresses with a specific host name and change the linked addresses at a rapid pace (also well known as “fast flux”) [2,3]. Domain flux on the other hand is essentially the opposite of IP flux, enabling botmasters to continuously alter and associate multiple FQDNs to a single IP address or C&C infrastructure. Mainly, domain fluxing is achieved with the help of a technique known as domain generation algorithm (DGA). Given a random seed, DGA produces a number of unique pseudorandom domain names that may resolve to the IP address of C&C server [4]. The efforts of the defenders focus on the timely prediction of the generated domain names in an effort to sinkhole them. In this manner, the bots will be unable to connect to the C&C infrastructure and receive orders.

1.1 Our contribution

The focus of the paper at hand is on mobile botnets. As the most valuable asset for a botnet operator is to retain its anonymity, we came with the idea of employing one or multiple mobile proxies (HTTP servers) in front of the botmaster. This means that all bot agents communicate with a proxy rather than the botmaster directly. More importantly, this proxy’s domain name and IP address changes constantly (i.e., both domain and IP flux) with the intent of minimizing the chances of having the botnet detected. On the other hand, using a mobile device as a HTTP proxy to carry out C&C could significantly deplete its energy reserves. So, a betterment is offered, having a separate PC-based botnet as a sidekick to handle proxy operations. Although this may slightly complicate the deployment of the botnet, as the botherder needs to also infect typical PCs, it is estimated to increase the overall stability of the botnet. On top of that, another powerful setup of the C&C infrastructure is put forward in which all botnet communications exploit DNS protocol as a covert channel. This further contributes in keeping the botnet operation obscured as all C&C transactions appear to the security systems, say, a firewall or intrusion detection system (IDS) perfectly legitimate in the form of DNS queries and responses. Moreover, this layout infuses simplicity given that no HTTP proxy or other intermediate server is required. All the above-mentioned architectures are evaluated through the use of a legacy TCP flooding attack and a more advanced DNS amplification one as given in [5]. Note that while detailed results on the impact of the attacks are offered, a large part of the focus is not on them; rather they are utilized to provide proofs on the sound use of the novel C&C architectures introduced. On the other hand, considering the literature on mobile botnets, so far nearly all works concentrate on botnet architectures without evaluating them nonetheless. In this respect, the current paper attempts a dual contribution; it not only offers some advanced C&C mobile botnet architectures, but also evaluates them using real-world attack scenarios.

The rest of the paper is organized as follows. The next section succinctly discusses botnet architectures and addresses related work on the topic. The novel C&C architectures we introduce are presented in Sect. 3. In the same section, we detail on the implemented attack scenarios. The evaluation results for all the attack scenarios are given in Sect. 4. A discussion on the findings along with a comparison between the different setups is included as well. Section 5 elaborates on possible countermeasures. The last section concludes and proposes some lines for future research.

2 Background and related work

In this section, we provide some background information on botnet architectures and pay special attention to mobile ones which are the focus of this study. Related work on the topic is interwoven into this section as well.

2.1 Architecture

Depending on how the bots are remotely controlled by their master, i.e., how the C&C channel is structured, one is able to classify them into centralized, decentralized or hybrid architectures. Indeed, as already pointed out, the C&C channel constitutes the most critical part of a botnet. For the attacker is the way they coordinate the bots and disseminate their commands, while for the defender represents the point which once it is detected the whole botnet can be eliminated, hijacked or neutralized.

The centralized infrastructure is based on the client server model, where all bots are directly connected with one, or few, C&C servers. These servers undertake to coordinate the bots and instruct them to take action. Although a centralized botnet exhibits optimum coordination and rapid dissemination of the commands, it also poses a single point of failure. From the moment the C&C server is detected and deactivated, the entire botnet is turned off. As a rule of thumb, the communication channels in this approach are based on HTTP or IRC protocol. In the first case, the communication is disguised inside the normal Web network traffic as the usage of Web is allowed in most networks, including corporate ones [6]. On the other hand, in IRC-based architecture the bots are connected to IRC channels and waiting for commands from the botherder. Of course, the messages on the IRC channel are in an obfuscated custom dialect, e.g., encrypted or hashed to avoid disclosure [7]. Apart from HTTP and IRC, DNS has been also exploited as a carrier protocol for establishing a covert channel [8]. With the help of a technique called DNS tunneling, the botnet entities transfer data embedded within the RRs of a DNS message, either query or response. Usually, the first labels of the domain name in a query are utilized by bots for transmitting data to the botmaster, while TXT RRs in responses contain the botmaster's commands [9]. The key benefit of employing DNS is that it is one of the few protocols that is rarely filtered out by firewalls.

To overcome the aforementioned weaknesses and to evade detection, a decentralized architecture may be selected to carry out the C&C mechanism. In this approach, there is not a central C&C server, but rather the various bots communicate with each other via P2P protocols. In other words, the bots behave as C&C server and client at the same time. Therefore, if any of the bots is tracked down and deactivated, there are no implications to the robustness of the entire network [10].

The hybrid architecture combines the advantages of both the centralized and decentralized ones. That is, in this setting, the bot agents exhibit diverse functionalities. Some of them temporarily undertake the C&C server role, with the aim to coordinate the botnet and disseminate the instructions, while the others wait for commands before springing to action [11].

In [12], the authors present a theoretical "random" botnet infrastructure in which the botherder or any other member of the botnet possesses no information for the other members of the botnet. This model is completely contrary to the centralized model, where the botnet operator knows beforehand all the members of the botnet. In a random topology, whenever the botmaster wishes to command a bot has to randomly scan the Internet to locate it. This model has the benefit that the tracking of a single bot will not reveal any information about the botnet or the botherder. Of course, the applicability of such a random structure is questionable, as it imposes limited coordination whenever an instruction needs to be delivered to the bots.

2.2 Mobile botnet

Over the last years, mobile devices quickly evolved from pure telecommunication devices to small and ubiquitous computing platforms. Few will argue that nowadays such devices are equipped with enough capabilities to even replace the usage of laptops. Therefore, it should come at no surprise that they attract the attention of resourceful attackers. Whereas the first appearance of malware that spread through mobile devices has been reported back in 2004 with the emergence of Cabir worm for Symbian OS, the initial materialization of malware with C&C capabilities happened some years later [13].

In 2009, Symbian/Yxes worm was detected to infect mobile devices with Sybian OS 9. This worm has the functionality to stealthy communicate via device's network interface with a remote server in order to send an HTTP request. Although this malware should not be considered as creating a C&C channel, it demonstrates that with few modifications to its functionalities it can easily obtain one [14]. In fact, later that year (November 2009), the first ever mobile botnet appeared, namely iKee.B bot client. This malware targeted jailbroken iPhones with the aim of turning them to bots. When the binary of iKee.B is installed on the infected iPhone, among other malicious functionalities, it periodically sends a HTTP GET request to the C&C server (based on a centralized architecture) in order to receive new scripts to update its binary. The corresponding malware was released in Europe, while the C&C server was located in Lithuania [15]. For the case of Android platform, Geinimi malware emerged in December 2010, presenting similar to iKee.B C&C capabilities based on HTTP protocol. This malware actually receives commands instructing it to steal private data from the mobile device [16]. A more advanced way for botnet coordination is exhibited recently by the AnserverBot, where the bot agents acquire the commands from encrypted contents of blog sites [17].

Excluding legacy network interfaces for connecting to the Internet (e.g., via Wi-Fi, 3G, etc), in a mobile botnet the bots can communicate via Bluetooth, SMS or MMS messages with their master or with each other [18]. Singh et al. [19] evaluate the Bluetooth technology as a means for establishing a C&C channel. They propose a scenario in which the herder sends the commands to devices with the "highest" Bluetooth connectivity, that is, to bots that are connected with many infected devices. Afterward, these bots forward the commands to their connections. They conclude that the usage of Bluetooth will endorse the spread of the commands and reduce the detectability of the underlying botnet, as the volume of traffic which passes through observable channels (Wi-Fi or 3G data) will be minimum. Moreover, the work in [20] presents a botnet capitalizing on SMS messages to realize its C&C channel. In this decentralized architecture, the various bots forward the commands to neighboring nodes via texting, thus forming a P2P infrastructure. Expanding this approach, Mulliner and Seifert [21] propose a hybrid botnet that combines SMS and HTTP protocol as C&C in order to reduce the billing cost and therefore avoid to draw the suspicion of the owner of the device.

More recently, Xiang et al. [22] propose Andbot, an advanced mobile botnet for Android mobile devices. Andbot uses a centralized C&C topology, that is, the bots connect to specific MicroBlog services. However, the herder utilizes different blog pages of the services, with the URL of the blogs generated by a predefined algorithm. Therefore, if a blog is blacklisted, the bots will connect to the next blog page to retrieve their orders. The authors call this technique as "URL Flux" in correspondence with that of Fast Flux. Similarly, SoCellBot given in [23] takes advantage of the social networks (SN). Specifically, this type of bot exploits the messaging system of SN to infect mobile devices and also as a covert communication channel for the coordination among the bot and the herder. In the same context, Zhao et al. [24] abuse cloud-based messaging services to distribute orders to infected devices. Finally, an interesting approach is presented in [25], where the authors investigate not the network capabilities of a mobile device but rather its embedded sensors including optical, audio, vibration and magnetic field ones as a means for creating a covert communication channel. For example, a potential botherder could disseminate commands via acoustic or magnetic signals in nearby infected devices.

2.2.1 Benefits and limitations of mobile botnet

Given the fact that mobile devices are equipped with powerful capabilities and features, it is corollary to be targeted by potential attackers. As already pointed out, nowadays, mobile devices have networking functionalities, i.e., they can connect to Internet via WiFi connections or via mobile broadband, and hence are able to utilize popular network protocols such as HTTP and DNS. Even more, their owners tend to constantly keep the wireless or data connection turned on in order to stay tuned with their favorite social network or connected to an instant messaging (IM) or voice over IP (VoIP) service. Another advantage of the mobile botnets is that they do not exhibit diurnal behavior as that of the equivalent PC-based botnets [26], since mobile devices rarely get turned off during the night period. Moreover, mobile devices are capable of acquiring new IP addresses very often as matter of few minutes [27]. Thereupon, traditional defense mechanisms that block request from blacklisted IP addresses are not normally applicable to this case. Furthermore, a resourceful botmaster could configure the bots to utilize only open Wi-Fi networks in an effort to eliminate the traces of the true perpetrator behind the DDoS incident [28].

On the other hand, there are several issues that complicate the deployment of a mobile botnet and therefore it should carefully considered by the botnet operator [22]. Firstly, the battery consumption of the mobile device is a critical factor. In the case the power consumption—as the bot agent drains its battery power—is far more quicker than could ever result from normal usage, then it will notify the end user that something goes wrong with his device. This situation would probably alert him to deactivate the device and, at least temporarily, disconnect it from the botnet. Secondly, providing that the volume of the data traffic created by the C&C channel or the execution of the commands, exceeds a normal threshold, then the overloaded network connection or even worst the increased billing cost will certainly raise the suspicion of the owner of the device. Finally, the usage of internal IP addresses rather than external, it will hamper the creation of C&C channel similar to the PC-based botnets.

3 Proposed C&C architectures and attack scenarios

3.1 Preliminaries and planning

The basic idea behind the introduced C&C architectures is to design and evaluate a mobile botnet that will be able to launch DDoS attacks against DNS servers based on the DNS amplification attack scenario given in [5]. The latter work brings out a new flavor of DNS amplification assault which is reported to achieve a 44 amplification factor at best. Its main advantage is that it does not disclose any illicit or dubious activity during its execution and thus to our knowledge is the most advanced of its kind so far. Namely, the network traffic during the execution of the attack seems to be perfectly legitimate (excluding the flooding effect of course). Moreover, the attack is very hard to be traced back to the perpetrator who, as a result, enjoys the advantage of anonymity. As detailed in [5], the scenario exploits the existence of network devices which operate as (open) DNS forwarders and simultaneously utilizes large DNSSEC [29–31] resource records (RRs) as payload to maximize the overall amplification factor. As it is known, DNSSEC-related RRs (i.e., RRSIG, DNSKEY, DS, NSEC/NSEC3) are large in size [32]. A very recent work [33] strongly supports this argument that DNSSEC-related RR can be exploited to augment the amplification factor of a DNS amplification attack. On the other hand, as already mentioned, the mushrooming of mobile devices connected to the network and the absence of security measures predicts the spread of mobile botnets.

Furthermore, a second variation of DDoS attack is examined exploiting the proposed C&C architectures. Its aim is to demonstrate that such architectures can be employed also for any kind of DDoS attack and is not solely dedicated to DNS amplification. In this case, the bots unleash a hefty number of TCP packets toward network ports that are well known to host popular services, for example, WEB, FTP and SSH.

For the implementation of a DNS amplification attack or any other type of reflection DoS attack, it is crucial for the mobile device to be able to conduct IP packet spoofing and in particular spoof the source IP address of UDP packets. As Google's Android OS does not permit apps to invoke such actions, we develop a dedicated malicious app that has embedded the libraries for executing Scapy tool [34] and other necessary apps and scripts. However, this app needs to gain administrative privileges (root) for executing installation on the infected mobile device, as every malicious app does.

Therefore, our intention is to disguise the aforementioned app into a legitimate looking one, such as an anti-malware scanner. This way, the owner of the device will provide the required permissions without knowing the true purpose of the app. However, it is to be noticed that the way the bots are infected lies out of scope of this work.

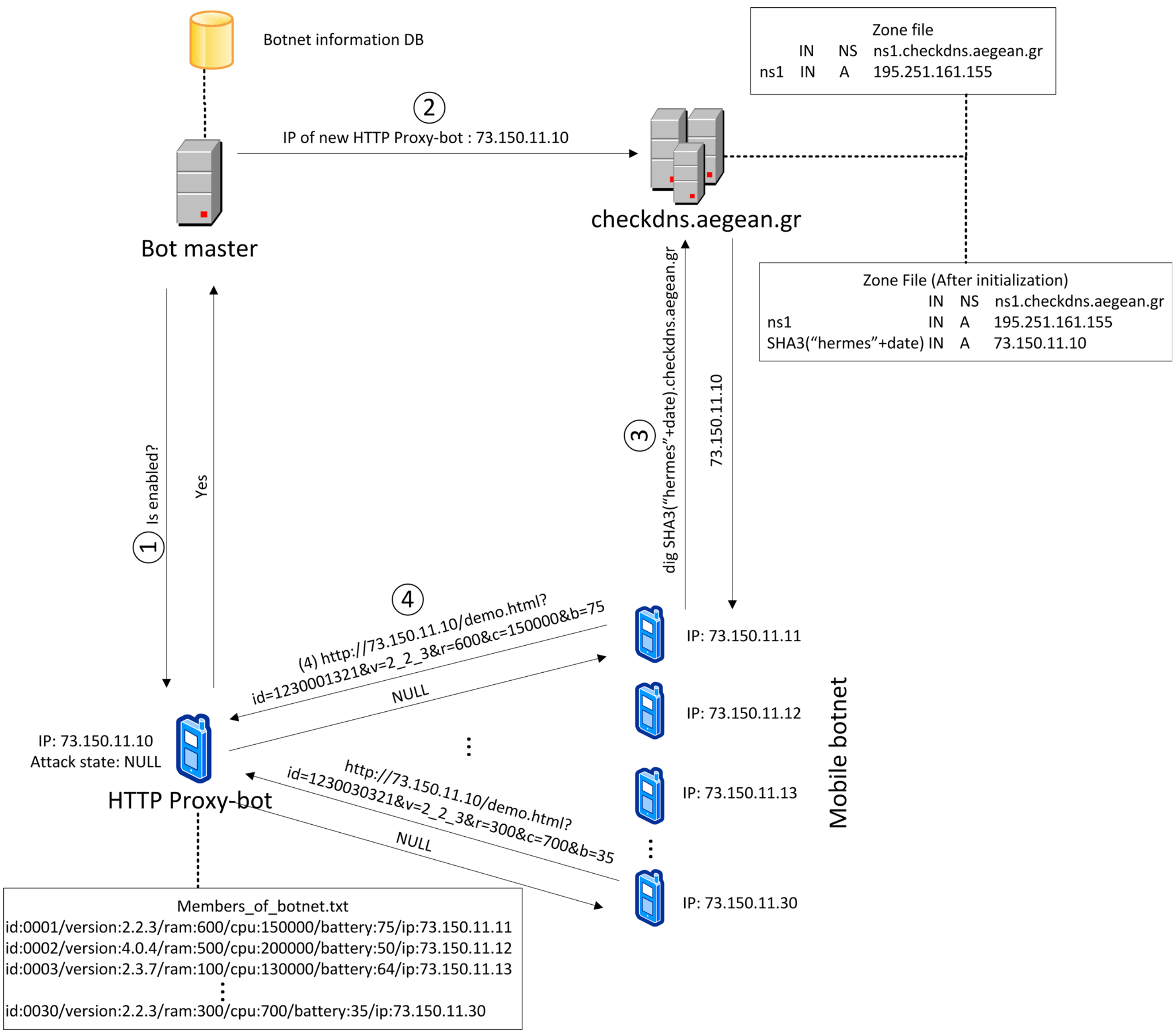
We design three scenarios for the creation of the C&C channel. The first two employ a HTTP server as the C&C server, while the third uses DNS protocol as the covert communication channel.

3.2 First scenario: a purely mobile botnet

The first scenario is shown in Figs. 1, 2 and 3. As observed from the figures, this scenario involves the following entities:

- A typical computer that is responsible for the coordination of the bots and therefore is controlled by the botnet operator.
- An HTTP server (hereinafter also referred to as "proxy"). As detailed in the following, this role is interim; it is assigned to one of the mobile bots and frequently reassigned to another.
- A DNS authoritative server that is responsible for the resolution of the proxy's domain names.
- The bot agents, i.e., infected mobile devices under the botmaster's implicit control.

In this architecture, the botmaster communicates directly only with the bot that is currently the proxy. Then, the proxy undertakes to disseminate the commands to the rest of the


Botnet information DB
Bot master
2
IP of new HTTP Proxy-bot : 73.150.11.10
checkdns.aegean.gr
1 Is enabled?
Yes
3
dig SHA3("hermes"+date).checkdns.aegean.gr
73.150.11.10
4
(4) http://73.150.11.10/demo.html?id=1230001321&v=2_2_3&r=600&c=150000&b=75
NULL
http://73.150.11.10/demo.html?id=1230030321&v=2_2_3&r=300&c=700&b=35
NULL
IP: 73.150.11.10
Attack state: NULL
HTTP Proxy-bot
IP: 73.150.11.11
IP: 73.150.11.12
IP: 73.150.11.13
IP: 73.150.11.30
Mobile botnet

Zone file

	IN	NS	ns1.checkdns.aegean.gr
ns1	IN	A	195.251.161.155

Zone File (After initialization)

	IN	NS	ns1.checkdns.aegean.gr
ns1	IN	A	195.251.161.155
SHA3("hermes"+date)	IN	A	73.150.11.10

```
Members_of_botnet.txt
id:0001/version:2.2.3/ram:600/cpu:150000/battery:75/ip:73.150.11.11
id:0002/version:4.0.4/ram:500/cpu:200000/battery:50/ip:73.150.11.12
id:0003/version:2.3.7/ram:100/cpu:130000/battery:64/ip:73.150.11.13
⋮
id:0030/version:2.2.3/ram:300/cpu:700/battery:35/ip:73.150.11.30
```

Fig. 1 Scenario 1—initialization phase

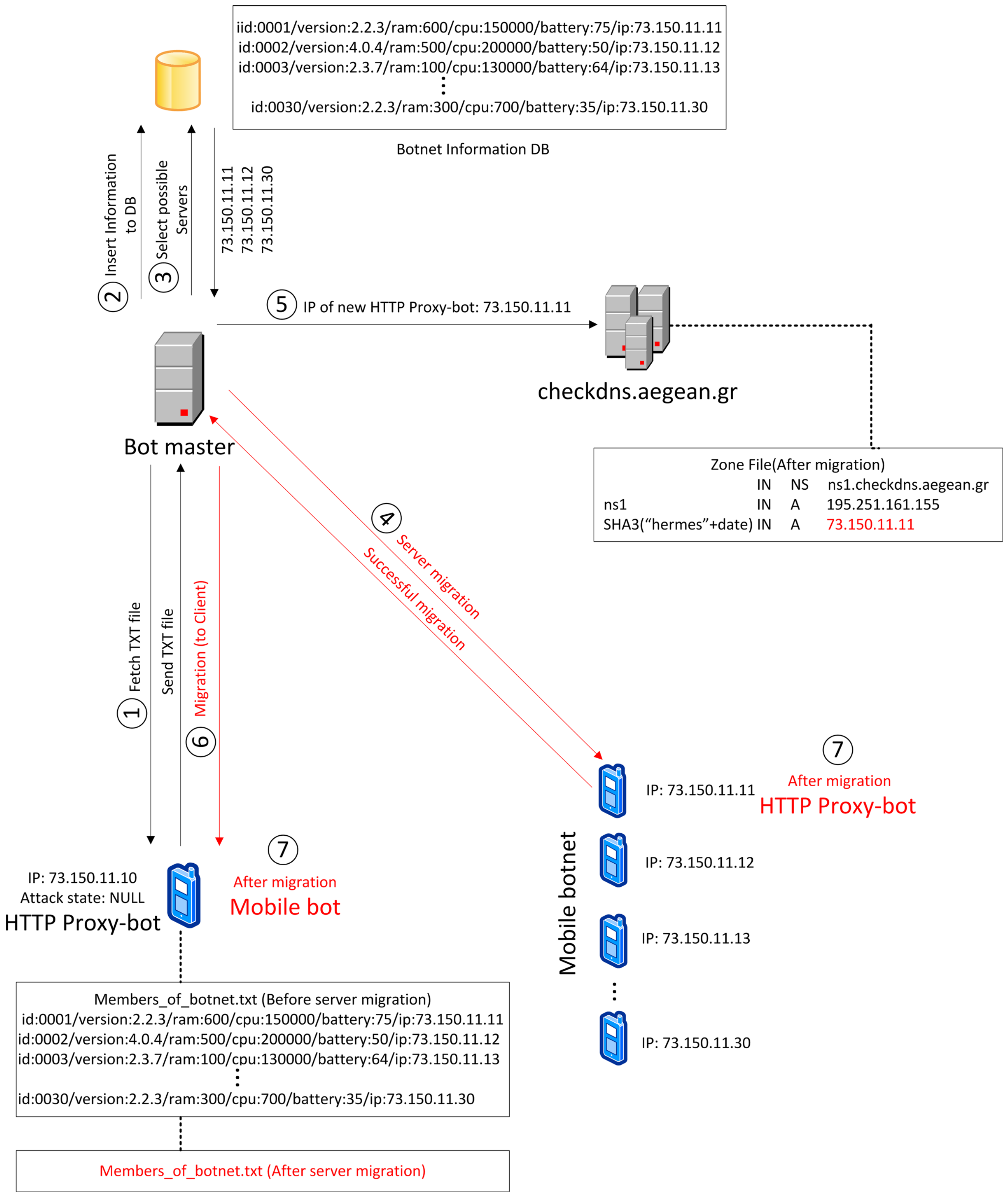

iid:0001/version:2.2.3/ram:600/cpu:150000/battery:75/ip:73.150.11.11
id:0002/version:4.0.4/ram:500/cpu:200000/battery:50/ip:73.150.11.12
id:0003/version:2.3.7/ram:100/cpu:130000/battery:64/ip:73.150.11.13
id:0030/version:2.2.3/ram:300/cpu:700/battery:35/ip:73.150.11.30
Botnet Information DB
2 Insert Information to DB
3 Select possible Servers
73.150.11.11
73.150.11.12
73.150.11.30
5 IP of new HTTP Proxy-bot: 73.150.11.11
checkdns.aegean.gr
Bot master
Zone File(After migration)
IN NS ns1.checkdns.aegean.gr
ns1 IN A 195.251.161.155
SHA3("hermes"+date) IN A 73.150.11.11
4 Server migration
Successful migration
1 Fetch TXT file
Send TXT file
6 Migration (to Client)
7 After migration HTTP Proxy-bot
IP: 73.150.11.11
IP: 73.150.11.12
IP: 73.150.11.13
IP: 73.150.11.30
Mobile botnet
IP: 73.150.11.10
Attack state: NULL
HTTP Proxy-bot
7 After migration Mobile bot
Members_of_botnet.txt (Before server migration)
id:0001/version:2.2.3/ram:600/cpu:150000/battery:75/ip:73.150.11.11
id:0002/version:4.0.4/ram:500/cpu:200000/battery:50/ip:73.150.11.12
id:0003/version:2.3.7/ram:100/cpu:130000/battery:64/ip:73.150.11.13
id:0030/version:2.2.3/ram:300/cpu:700/battery:35/ip:73.150.11.30
Members_of_botnet.txt (After server migration)

Fig. 2 Scenario 1—migration phase

members of the botnet. Following, the various bot agents send a DNS query to the DNS authoritative server with the aim to acquire the IP address of the proxy. As a final step, they transmit an HTTP request to the proxy and receive as HTTP response the commands placed by the botmaster. It is therefore obvious that the botherder remains always hidden

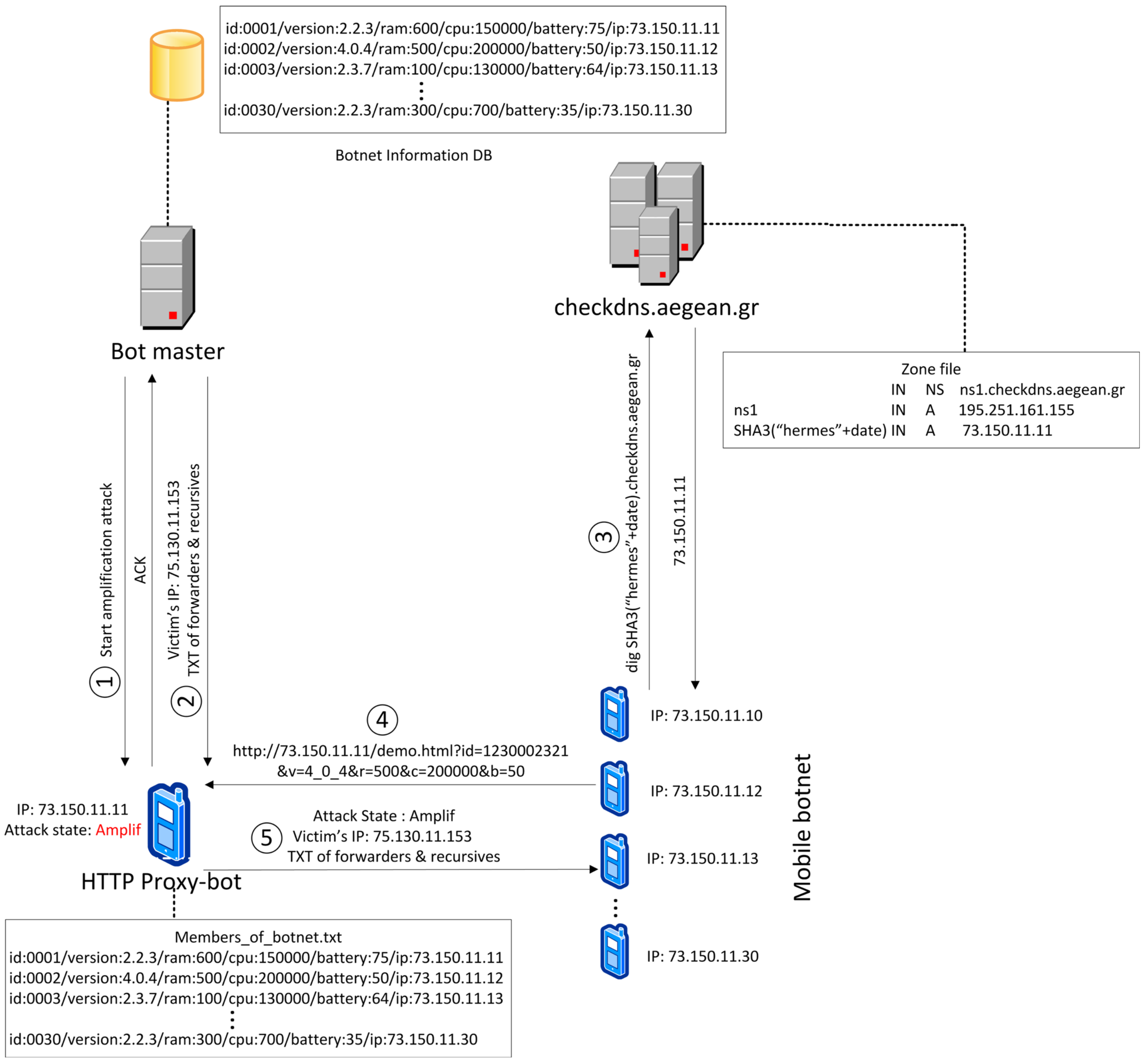

id:0001/version:2.2.3/ram:600/cpu:150000/battery:75/ip:73.150.11.11
id:0002/version:4.0.4/ram:500/cpu:200000/battery:50/ip:73.150.11.12
id:0003/version:2.3.7/ram:100/cpu:130000/battery:64/ip:73.150.11.13
id:0030/version:2.2.3/ram:300/cpu:700/battery:35/ip:73.150.11.30
Botnet Information DB
checkdns.aegean.gr
Bot master
Zone file
IN NS ns1.checkdns.aegean.gr
ns1 IN A 195.251.161.155
SHA3("hermes"+date) IN A 73.150.11.11
① Start amplification attack
ACK
② Victim's IP: 75.130.11.153
TXT of forwarders & recursives
③ dig SHA3("hermes"+date).checkdns.aegean.gr
73.150.11.11
④ http://73.150.11.11/demo.html?id=1230002321
&v=4_0_4&r=500&c=200000&b=50
IP: 73.150.11.10
IP: 73.150.11.12
IP: 73.150.11.13
IP: 73.150.11.30
Mobile botnet
IP: 73.150.11.11
Attack state: Amplif
⑤ Attack State : Amplif
Victim's IP: 75.130.11.153
TXT of forwarders & recursives
HTTP Proxy-bot
Members_of_botnet.txt
id:0001/version:2.2.3/ram:600/cpu:150000/battery:75/ip:73.150.11.11
id:0002/version:4.0.4/ram:500/cpu:200000/battery:50/ip:73.150.11.12
id:0003/version:2.3.7/ram:100/cpu:130000/battery:64/ip:73.150.11.13
id:0030/version:2.2.3/ram:300/cpu:700/battery:35/ip:73.150.11.30

Fig. 3 Scenario 1—amplification attack

behind the transient proxy. The various phases of the botnet C&C establishment are analyzed below:

– *Initialization phase*—For the botnet to boot up, we consider that the very first proxy-bot exists on a specific IP address. This is a logical assumption as the botnet operator can always employ a mobile device on her own. As the proxy will shortly shift, we assert that this would not jeopardize the camouflage of the botnet. First off, as it is depicted in Fig. 1, the botmaster queries whether the initiatory proxy is enabled and listening ①. If an affirmative response is received, she updates the zone file of the DNS server with the RR that links the domain name of the proxy with its IP, that is, following a fast-flux strategy. Naturally, this domain name is not a trivial one, but rather the output of the keyed-hash message authentication code (HMAC-SHA256) of the current global date with a secret code, which in our case is the string "hermes" ②. Note that with the use of the HMAC function, the domain name of the proxy will be different every day and thus evade suspicion. Also, this rapid turnover makes it very hard to track down or block every possible domain name. Although in our case only the leftmost label of the domain label is generated, a potential attacker will easily register FQDNs that will be produced by the aforementioned generation algorithm or even employ dynamic DNS providers. In such a case, she is able to leak out even less traces of the botnet's infrastructure. Following, all the bots resolve the domain name of the proxy ③ and are able to connect with it via HTTP protocol. This query is repeatedly issued for example every one or two minutes in order the bots to become aware of any change regarding the proxy's IP address. After that, as the bots know their proxy, they create an HTTP GET request that includes information of their operational status and settings, namely RAM, CPU, level of battery power, device's ID (International Mobile Equipment Identity (IMEI) number) and operating system (OS) version ④. In our case, these values are transmitted in cleartext. Nevertheless, a more careful botherder could encrypt them with the help of, say, a symmetric cipher. On the opposite, an encryption process will cause significant power consumption and require some sort of key management. In any case, we obfuscate the IMEI by padding meaningless numbers at the front and the end of the value. It can be safely argued that the remaining values do not directly provide any useful info for the defender to trace the botnet. This provided information is used later on for proxy migration. In turn, the proxy collects all the information, adds to each record the bot's IP and responds with the status of the attack, meaning what action the various bots should take. For the moment, as the botnet is in its initialization phase the bots should remain silent.

– *Proxy migration phase*—Every, say, three minutes the proxy will migrate to another bot. This procedure is presented in Fig. 2. As observed in the figure, before the migration begins, the botmaster requests the file that contains all the information of the bots ①. For the protection of the latter, this file is kept (and sent) encrypted with the public key of the botmaster. When the herder receives and decrypts the data with her private key, she updates the corresponding database (DB) with the members of the bot. If a member is already registered in DB, she simply updates its IP address and battery level, otherwise she creates a new entry based on the device ID ②. Next, she inquires which bots are capable of undertaking the role of proxy, based on their status, i.e., have enough computational power and acceptable battery level ③. From the possible candidates, she chooses randomly one and notifies it to change its mode from simple bot to proxy ④. Also, she updates the attack status in the new proxy to make it available to the bots to acquire. In case a migration problem occurs, the botmaster selects another candidate, otherwise she updates the zone file ⑤. Finally, she informs the previous proxy to change its mode back to bot and to erase any information related to the botnet, namely the encrypted text file that contains the information for all the members of the botnet ⑥.
– *Recovery phase*—One significant issue that needs to be considered is the case where the proxy fails. This, for example, may happen because it is deactivated by its owner or its battery reserves have been depleted. In the case a recovery process has not been foreseen, then the botnet will be uncoordinated for some time until the next proxy migration phase takes place. For this reason, the herder regularly sends an HTTP request (heartbeat) to the current proxy to examine whether it is alive. If not, the recovery phase initiates. Similarly to the mitigation phase, the botmaster chooses one of the candidates for the role of the proxy, notifies it to change its mode and the botnet attack status, while she updates the zone file for announcing the new proxy to the bots. As the connection with the previous proxy was lost, in the case of restart, the app on the bot side is configured to operate as client and to also erase the information of the botnet, if any. This way, the previous proxy will begin to operate as a simple bot client and therefore wipe out all the valuable traces of the botnet members.
– *Attack phase*—Whenever, the botnet operator decides to launch an attack, she sets off the attack phase. As it is depicted in Fig. 3, she notifies the proxy for the status of the attack ① and the corresponding parameters ②. In our case, we implement two DDoS attacks: the first one is based on [5] and targets mainly DNS authoritative servers, while the latter unleashes a typical TCP flooding attack targeting a multipurpose server. As it is illustrated in the figure, for the amplification attack, the botmaster sends the IP of the victim and a text file containing all the (open) DNS forwarders and recursive servers that will be exploited during the assault. Note that these two messages are silently send without an ACK, since the proxy affirms that it is alive in ① message round trip. Following, the proxy changes its attack status to "Amplif," meaning that a DNS Amplification attack must be unleashed. As with the initialization phase, as soon as the bots resolve the generated domain name of the proxy ③ and connect to it ④, they receive the attack status and the associated parameters ⑤. Soon after, they start to launch the DDoS attack. At the time the botmaster wishes to terminate the assault, she changes the attack status to the string "NULL." The bots are informed for that change since as already pointed out in the initialization phase, the bots repetitively query the proxy to learn about its status. Bear in mind that during the attack phase the proxy could migrate to another bot. However, the attack continues without cease because the new proxy is updated with the current attack status and the relative parameters. In a similar manner, for the case of a TCP connection flooding, the botmaster changes the attack status to "TCP" ① and informs about the IP address of the victim ②. When

the bots become aware of the target, they continuously issue large TCP packets to the network ports of the victim that are well known to host popular services, e.g., WEB, FTP, SSH and DNS.

It is stressed that depending on the size of the botnet, the botherder will enable two or more proxies for the dissemination of the commands. Thus, the zone file will contain one RR for each proxy and the various bots will randomly choose one of them to connect, say, in a round-robin fashion. This segmentation is also backed up by the fact that the mobile devices may not have the sufficient computational power to serve a large amount of HTTP requests.

3.3 Second scenario: mobile botnet with PC-based proxies

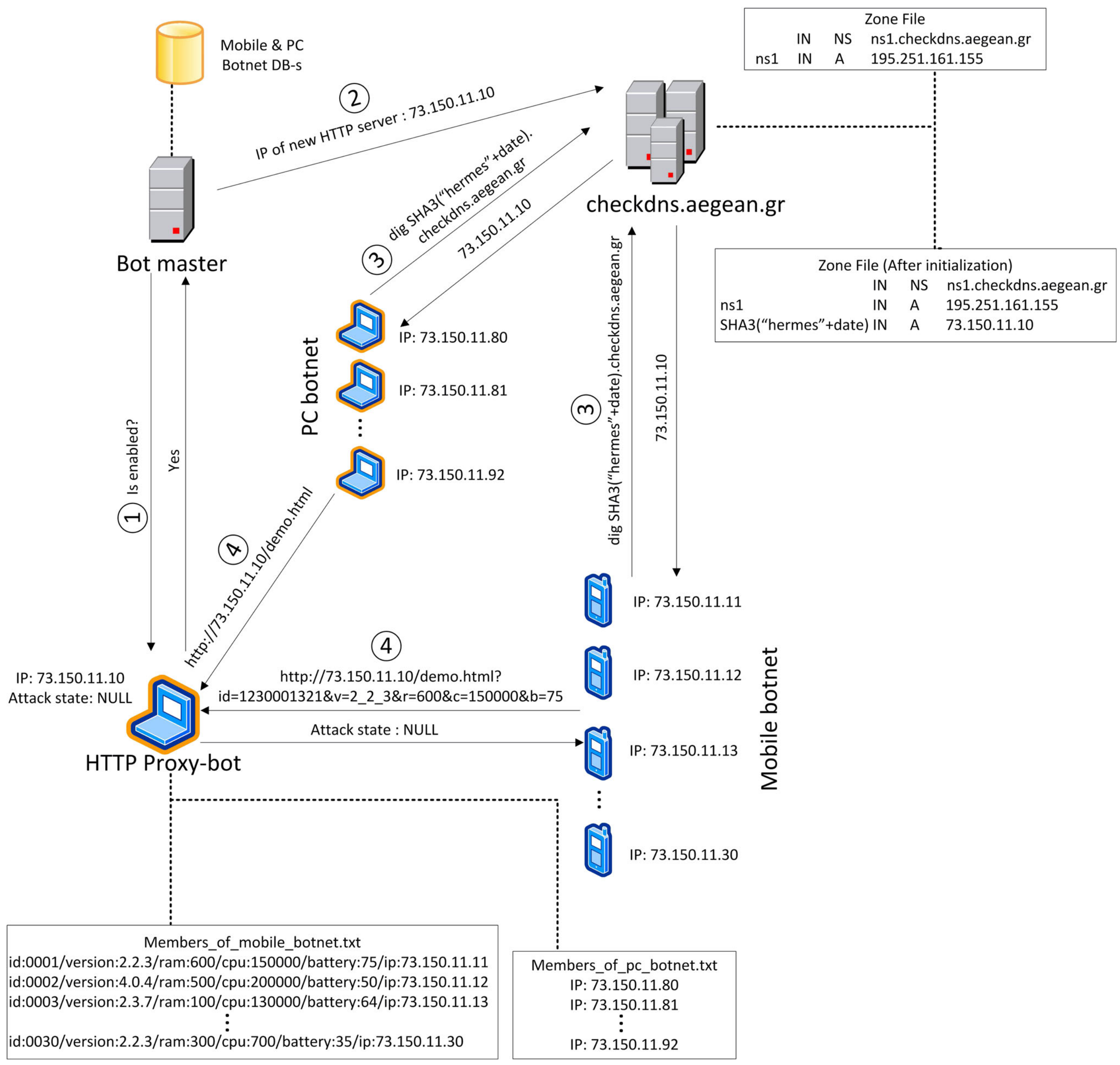
The previous scenario could be considered as a pure mobile botnet, given that all bots are mobile devices. However, for diminishing the communication and processing costs, we came up with the idea of employing standard PC bots as proxies. Consequently, this second scenario constitutes a variant of the first one, with the difference that only PC bots acquire the role of the proxy machine (HTTP server) and not mobile ones. The advantage of doing so is that a PC has fewer chances to become non-operational due to power constraints, etc. Also, it has more computational capabilities to serve a larger number of bots. Putting it another way, in this architecture, the burden of the attack is undertaken by a mobile botnet, while the coordination of the mobile botnet is carried out by a PC-based botnet. To sum up, this scenario uses the same entities as the first one except the PC bots.

- *Initialization phase*—As depicted in Fig. 4, similarly to the first scenario, the botmaster queries whether the proxy is alive ① and updates the zone file with its IP, if this is the case ②. The main difference is perceived in steps ③ & ④, where the members of the PC-based botnet are introduced. In ③ the PC bots resolve the domain name of the proxy to which following they send an HTTP request for registering ④. Additionally, the proxy manages a list with the details of the PC-based botnet, from which the botmaster chooses the candidates for determining the next proxy.
- *Proxy migration phase*—Like in the first scenario, the migration phase happens regularly for the transition of the proxy to be completed. As shown in Fig. 5, prior to migration, the botmaster requests the files that contain all the operational information of both the PC and mobile bots ① and updates the corresponding database ②. From the available PC bots, the botherder chooses randomly one ③ and notifies it to alter its mode from simple bot to that of a proxy ④.
- *Recovery phase*—This phase is exactly the same as in the case of the first scenario, where the candidates are picked up from the list of the infected PCs. However, we expect that the probability of PC-based proxy to crash is much lesser than that of a mobile one.
- *Attack phase*—For the launch of the attack, the botherder changes the attack status in the proxy ① and provides the required parameters ②, namely the IP address of the victim and the type of the attack (Fig. 6). Once the mobile bots resolve the domain name of the current PC-based proxy ③, they are able to issue HTTP GET requests ④. This way, the commands are being disseminated to the bots ⑤ for them to initiate the DDoS attack. Note that the PCs of the PC-based botnet do not participate in the actual attack as an extra precaution against detection since their role is more valuable as proxies than attacking bots. Actually, this decision is up to the botherder and usually depends on several factors, including the number of the PC-based bots and their geographical dispersion.

3.4 Third scenario: exploiting DNS as covert C&C channel

The main concept behind this third scenario is to use DNS protocol as a covert channel for coordinating the mobile botnet. An overview of this architecture is depicted in Fig. 7. The core idea here is that the botmaster controls a DNS authoritative name server and publish the relative parameters of the attack as RRs to that zone. Thus, the various mobile bots will neither contact the botmaster directly nor via a proxy, but instead they receive their instructions through the DNS server. As observed from the figure, the domain name of the RR is not a trivial name, but rather the output of the HMAC-SHA256 hash function taking as input the current global date and a secret alphanumeric. Note that the use of a hash value ensures that the domain name will be disparate every day and thus evade suspicion. Once more, although in our proposal only the leftmost label of the domain label is random, a potential attacker will easily register domain names that will be produced by the generation algorithm. In such a case, she is able to exploit various DNS authoritative servers and leave minimum traces of the botnet's C&C activity. The two phases that complete this attack variation are given below.

- *Initialization phase*—Originally, the zone file of the DNS server contains only the RRs that are required for the operation of the zone, namely which nameserver is responsible for that zone (NS record) and in which IP address it is located (A record). For the coordination of the attack, the botmaster updates dynamically the zone file with two RRs of type A ①. This type of RR maps a domain name with its corresponding IP address. The first A record corresponds to the amplification attack,



Zone File

	IN	NS	ns1.checkdns.aegean.gr
ns1	IN	A	195.251.161.155

Zone File (After initialization)

	IN	NS	ns1.checkdns.aegean.gr
ns1	IN	A	195.251.161.155
SHA3("hermes"+date)	IN	A	73.150.11.10

```
Members_of_mobile_botnet.txt
id:0001/version:2.2.3/ram:600/cpu:150000/battery:75/ip:73.150.11.11
id:0002/version:4.0.4/ram:500/cpu:200000/battery:50/ip:73.150.11.12
id:0003/version:2.3.7/ram:100/cpu:130000/battery:64/ip:73.150.11.13
⋮
id:0030/version:2.2.3/ram:300/cpu:700/battery:35/ip:73.150.11.30
```

```
Members_of_pc_botnet.txt
IP: 73.150.11.80
IP: 73.150.11.81
⋮
IP: 73.150.11.92
```

Fig. 4 Scenario 2—initialization phase

while the second to the TCP flooding one. The purpose of these records is to signal the beginning of the attack and indicate which the target is. The domain names of these records are generated by the HMAC-SHA256 operation getting as input the current date and the alphanumeric "zeus" and "artemis," respectively. With the usage of the current global date, the name of the RR is different every day. While the botnet stays silent, the A RRs map to the private IP address 192.168.1.1, indicating that the bots must not take any action and wait. All the mobile bots periodically generate the random domain names and resolve their IP addresses ② & ③. If the answer contains the IP address 192.168.1.1, then the bots stay silent. Otherwise, they obey the attack instructions as the case may be. It is not compulsory for the various bots to directly issue the DNS queries toward the authoritative server controlled by the botmaster, but rather they could consult the local recursive resolver. In such a case, RRs should have a zero time-to-live (TTL) value.

– *Attack phase*—Whenever the botmaster wishes to unleash a DDoS assault, she dynamically modifies the contents of the zone file with the parameters of the attack. Depending on which kind of attack she desires to execute, i.e., DNS amplification (Fig. 8) or TCP flooding, or even

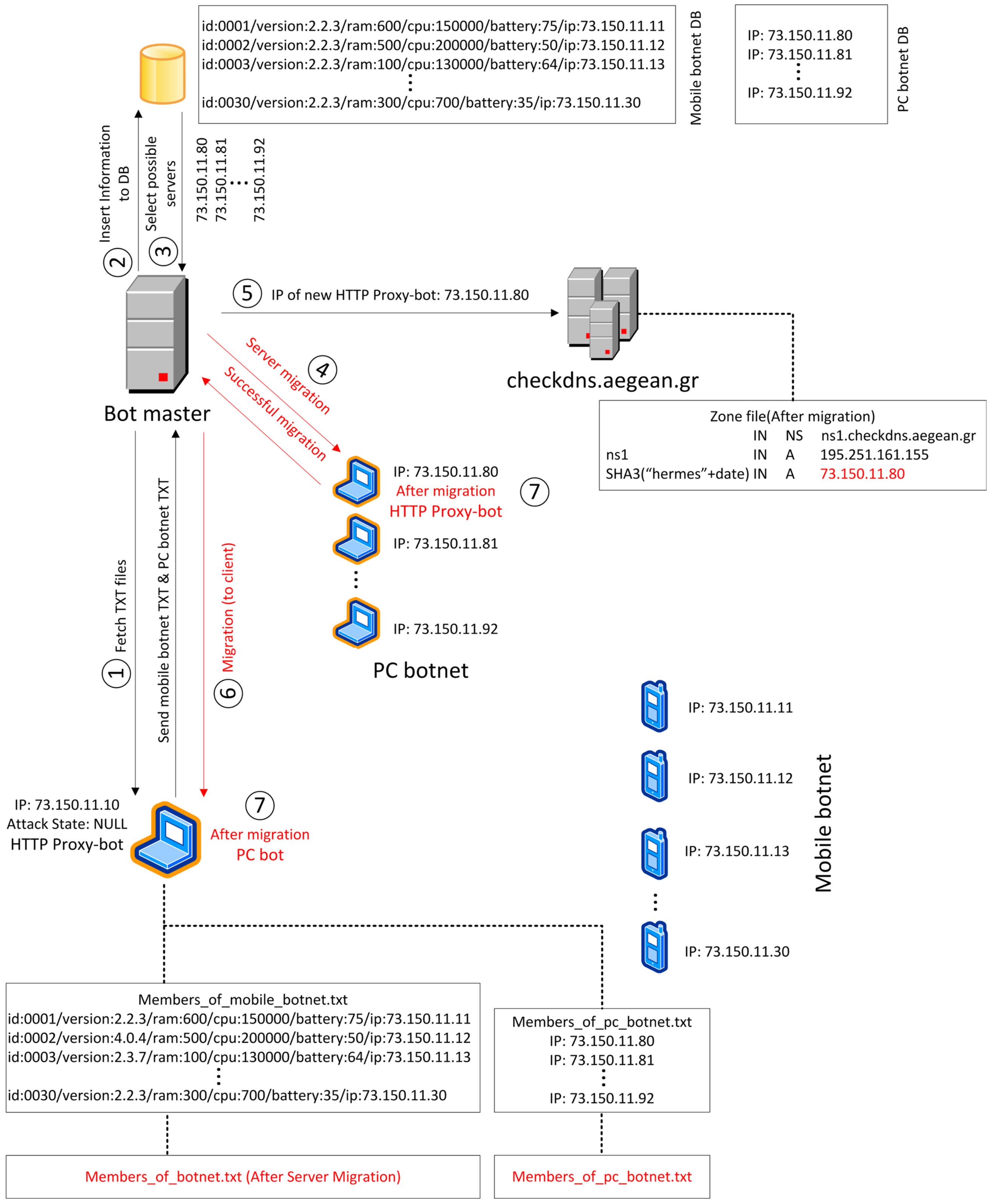

id:0001/version:2.2.3/ram:600/cpu:150000/battery:75/ip:73.150.11.11
id:0002/version:2.2.3/ram:500/cpu:200000/battery:50/ip:73.150.11.12
id:0003/version:2.2.3/ram:100/cpu:130000/battery:64/ip:73.150.11.13
⋮
id:0030/version:2.2.3/ram:300/cpu:700/battery:35/ip:73.150.11.30
Mobile botnet DB
IP: 73.150.11.80
IP: 73.150.11.81
⋮
IP: 73.150.11.92
PC botnet DB
② Insert Information to DB
③ Select possible servers
73.150.11.80
73.150.11.81
…
73.150.11.92
⑤ IP of new HTTP Proxy-bot: 73.150.11.80
checkdns.aegean.gr
Bot master
Server migration ④
Successful migration
Zone file(After migration)
IN NS ns1.checkdns.aegean.gr
ns1 IN A 195.251.161.155
SHA3("hermes"+date) IN A 73.150.11.80
IP: 73.150.11.80
After migration
HTTP Proxy-bot
⑦
IP: 73.150.11.81
⋮
IP: 73.150.11.92
PC botnet
① Fetch TXT files
Send mobile botnet TXT & PC botnet TXT
⑥ Migration (to client)
IP: 73.150.11.11
IP: 73.150.11.12
IP: 73.150.11.13
⋮
IP: 73.150.11.30
Mobile botnet
IP: 73.150.11.10
Attack State: NULL
HTTP Proxy-bot
⑦
After migration
PC bot
Members_of_mobile_botnet.txt
id:0001/version:2.2.3/ram:600/cpu:150000/battery:75/ip:73.150.11.11
id:0002/version:4.0.4/ram:500/cpu:200000/battery:50/ip:73.150.11.12
id:0003/version:2.3.7/ram:100/cpu:130000/battery:64/ip:73.150.11.13
⋮
id:0030/version:2.2.3/ram:300/cpu:700/battery:35/ip:73.150.11.30
Members_of_pc_botnet.txt
IP: 73.150.11.80
IP: 73.150.11.81
⋮
IP: 73.150.11.92
Members_of_botnet.txt (After Server Migration)
Members_of_pc_botnet.txt

Fig. 5 Scenario 2—migration phase

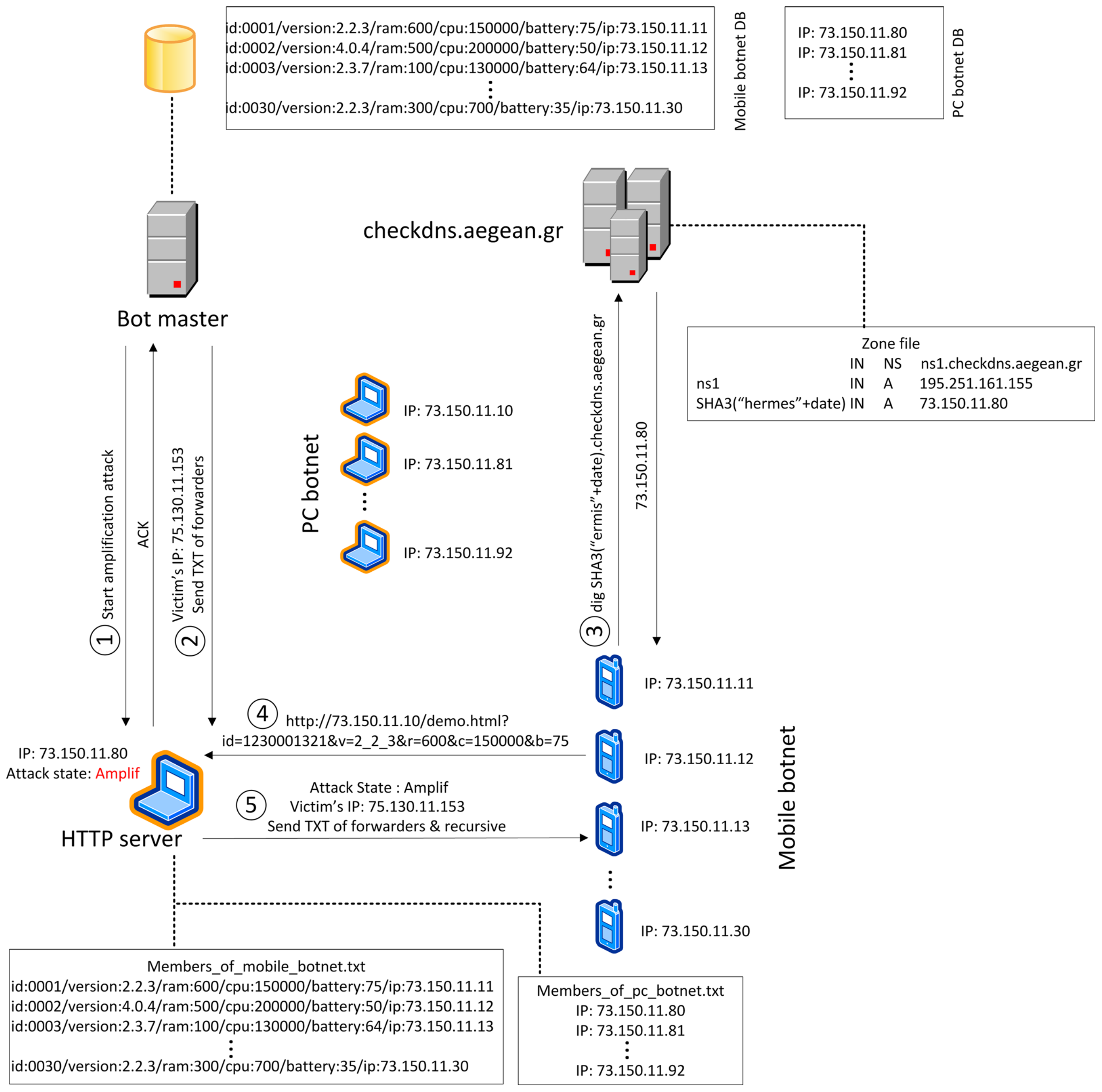

id:0001/version:2.2.3/ram:600/cpu:150000/battery:75/ip:73.150.11.11
id:0002/version:4.0.4/ram:500/cpu:200000/battery:50/ip:73.150.11.12
id:0003/version:2.3.7/ram:100/cpu:130000/battery:64/ip:73.150.11.13
id:0030/version:2.2.3/ram:300/cpu:700/battery:35/ip:73.150.11.30
Mobile botnet DB
IP: 73.150.11.80
IP: 73.150.11.81
IP: 73.150.11.92
PC botnet DB
checkdns.aegean.gr
Bot master
Zone file
IN NS ns1.checkdns.aegean.gr
ns1 IN A 195.251.161.155
SHA3("hermes"+date) IN A 73.150.11.80
1 Start amplification attack
ACK
2 Victim's IP: 75.130.11.153 Send TXT of forwarders
PC botnet
IP: 73.150.11.10
IP: 73.150.11.81
IP: 73.150.11.92
3 dig SHA3("ermis"+date).checkdns.aegean.gr
73.150.11.80
IP: 73.150.11.11
IP: 73.150.11.12
IP: 73.150.11.13
IP: 73.150.11.30
Mobile botnet
4 http://73.150.11.10/demo.html?id=1230001321&v=2_2_3&r=600&c=150000&b=75
IP: 73.150.11.80
Attack state: Amplif
HTTP server
5 Attack State : Amplif
Victim's IP: 75.130.11.153
Send TXT of forwarders & recursive
Members_of_mobile_botnet.txt
id:0001/version:2.2.3/ram:600/cpu:150000/battery:75/ip:73.150.11.11
id:0002/version:4.0.4/ram:500/cpu:200000/battery:50/ip:73.150.11.12
id:0003/version:2.3.7/ram:100/cpu:130000/battery:64/ip:73.150.11.13
id:0030/version:2.2.3/ram:300/cpu:700/battery:35/ip:73.150.11.30
Members_of_pc_botnet.txt
IP: 73.150.11.80
IP: 73.150.11.81
IP: 73.150.11.92

Fig. 6 Scenario 2—amplification attack

both, she replaces the corresponding A RR with that of the victim's IP address ①. As for the DNS amplification attack described in [5], the bots require the list of the exploited DNS forwarders. As a result the botmaster adds one or more TXT RRs that contain this list. Usually, the number of the DNS forwarders employed in [5] is large enough; thus, the added TXT RR is more than one. Hence, the TXT RR that matches to the domain name *HMAC-SHA256("zeus"+date)* indicates the number of the required TXT RR for the publication of the complete list of the forwarders, while the TXT RRs resolving to the domain names *HMAC-SHA256("zeus"+date)1*, *HMAC-SHA256("zeus"+date)2*, etc., contain a fragment of that list. Consequently, at the time the bots observe the change in the matching A RR they know that they should commence the DNS amplification DDoS attack ②. After that, they resolve the TXT record to get informed of the size of the list ④. Following, the bot agents issue a number of DNS queries about TXT RRs in a round-robin fashion to acquire a portion of the list of the available DNS forwarders ⑤. Finally, they are ready to initiate the attack. Any time the attacker desires to terminate the

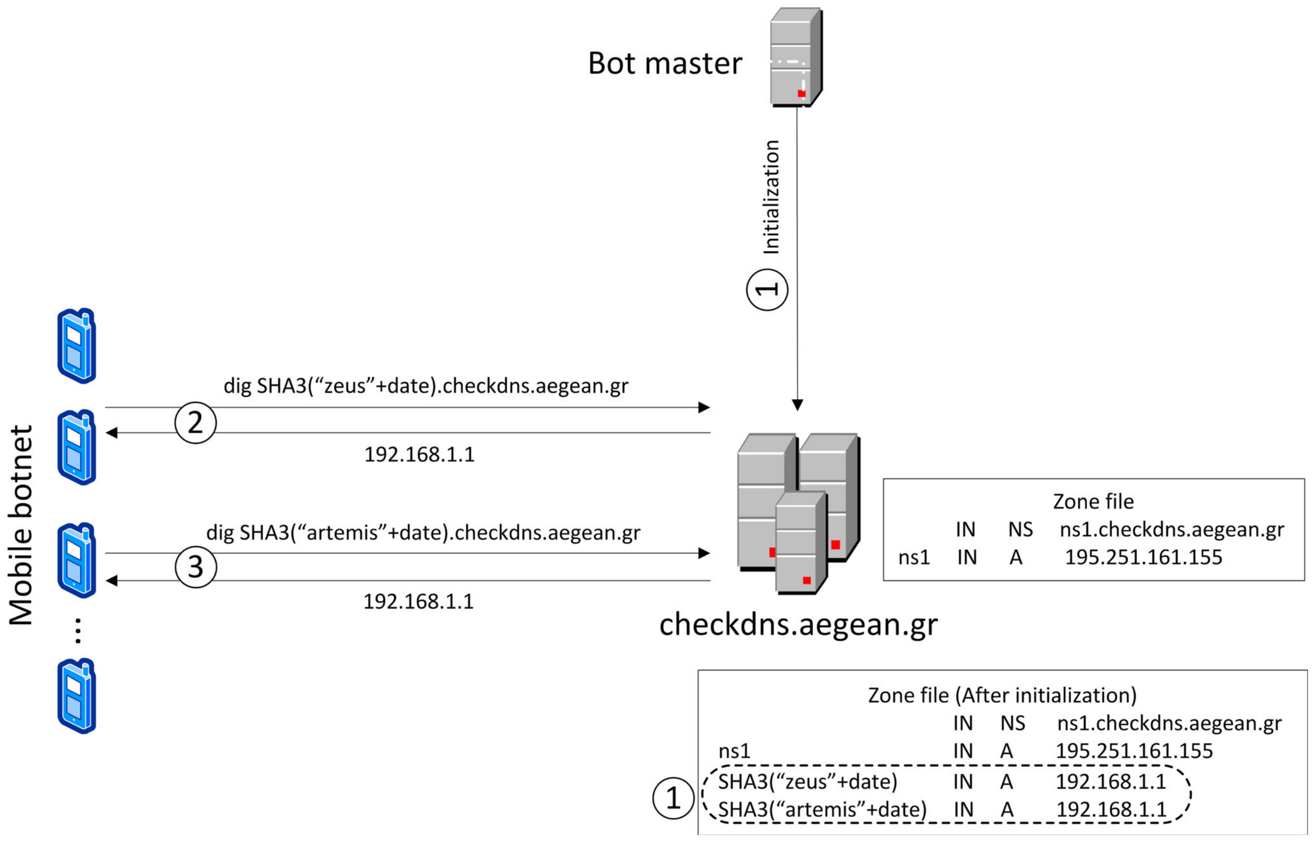

Bot master
Initialization
1
Mobile botnet
dig SHA3("zeus"+date).checkdns.aegean.gr
2
192.168.1.1
dig SHA3("artemis"+date).checkdns.aegean.gr
3
192.168.1.1
checkdns.aegean.gr
Zone file
IN NS ns1.checkdns.aegean.gr
ns1 IN A 195.251.161.155
Zone file (After initialization)
IN NS ns1.checkdns.aegean.gr
ns1 IN A 195.251.161.155
SHA3("zeus"+date) IN A 192.168.1.1
SHA3("artemis"+date) IN A 192.168.1.1

Fig. 7 Scenario 3—initialization phase

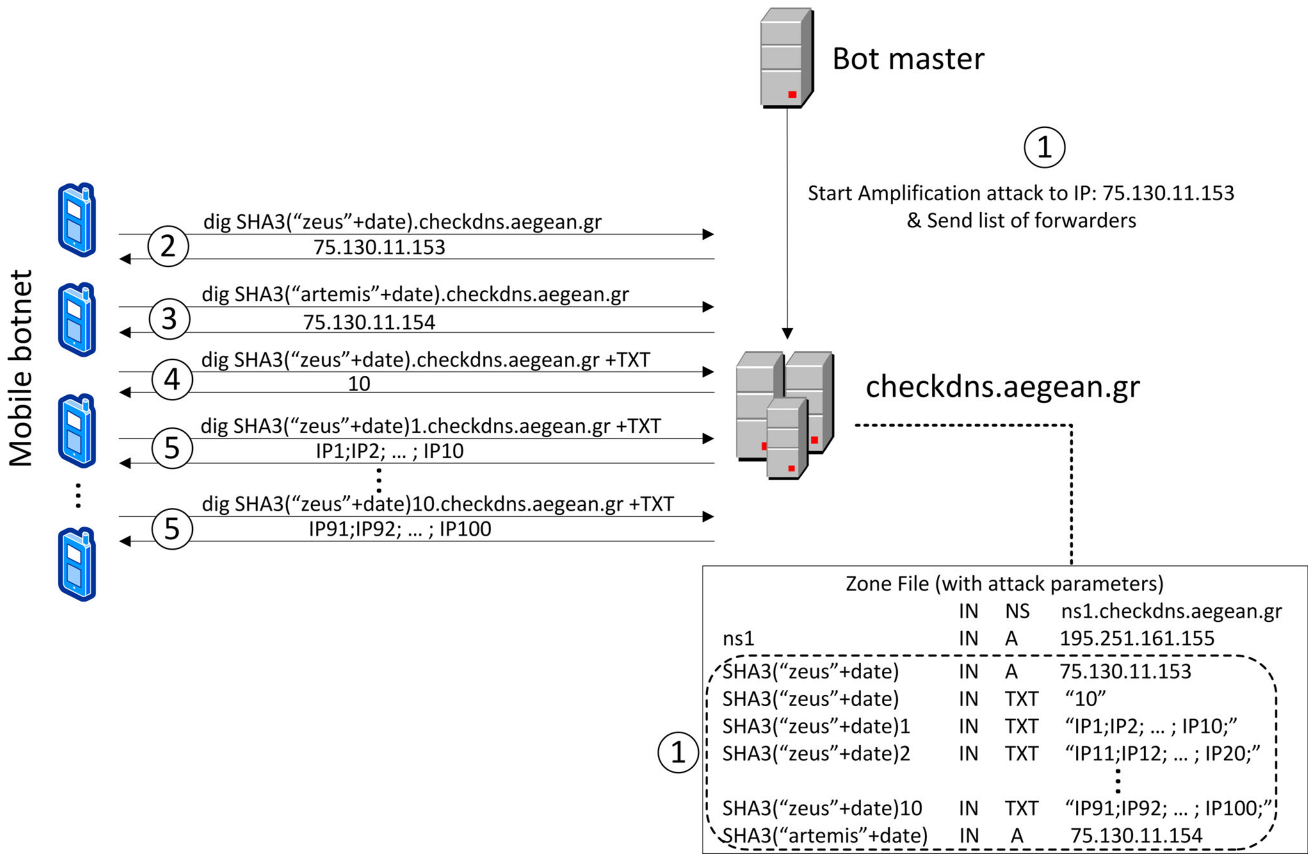
assault, she modifies the A RR to the private IP address 192.168.1.1 and removes the appropriate TXT RRs. So, the bots are notified of the change and they cease their part of the attack instantly. Similarly, for the TCP flooding attack case, the aggressor updates the RR corresponding to the domain name *HMAC-SHA256("artermis"+date)* with the IP address of the victim. Thereupon, the bots become aware that they have to start TCP connections to that IP address ③. From that point onward, the bots continuously issue TCP queries until the botherder modifies the IP address of the RR back to 192.168.1.1.

3.5 Other considerations

The decision of using a collateral PC-based botnet for dispatching proxy communications depends on the size of the mobile botnet and other parameters as the case may be. For example, it is natural to think that the greater the number of mobile bots the greater the need for PC-based proxies in order to easily control all of them and increase the total strength. However, if the botmaster already is in control of such a PC-based botnet, it can always take advantage of its services. Furthermore, depending on the impact the botmaster wishes to accomplish, i.e., immediate collapse of the target or low and slow attack, he will select the size of the botnet. Segmentation of the whole space of the botnet is also possible. So overall, in the eyes of the botmaster it is basically a matter of how rich, populous and scattered the bot arsenal is.

4 Comparison and results

The aforementioned scenarios present a novel mechanism to coordinate a mobile botnet. The first two exploit one of the infected device to act as HTTP (proxy) server, while the third one uses a DNS authoritative nameserver as the means for disseminating the commands. The difference between the first two scenarios resides on the fact that a mobile device has limited resources to handle a large number of bot clients. On the other hand, a PC-based proxy is more reliable and has fewer chances to crash or go offline. Additionally, the PC-based bots do not participate in the final DDoS attack(s), they do not reveal directly their location, and thus it is more challenging to get detected. In any case, our results described next indicate that if one employs multiple proxies simultaneously, the impact of the attack will be the same in both cases (i.e., either with mobile or PC-based proxies). The most notable advantage of the third scenario is that the bots do not directly connect to the botmaster, but rather they issue legitimate DNS queries for frequently changing RR that corresponds to the



Bot master
1
Start Amplification attack to IP: 75.130.11.153
& Send list of forwarders
Mobile botnet
2
dig SHA3("zeus"+date).checkdns.aegean.gr
75.130.11.153
3
dig SHA3("artemis"+date).checkdns.aegean.gr
75.130.11.154
4
dig SHA3("zeus"+date).checkdns.aegean.gr +TXT
10
5
dig SHA3("zeus"+date)1.checkdns.aegean.gr +TXT
IP1;IP2; … ; IP10
5
dig SHA3("zeus"+date)10.checkdns.aegean.gr +TXT
IP91;IP92; … ; IP100
checkdns.aegean.gr
Zone File (with attack parameters)
IN NS ns1.checkdns.aegean.gr
ns1 IN A 195.251.161.155
1
SHA3("zeus"+date) IN A 75.130.11.153
SHA3("zeus"+date) IN TXT "10"
SHA3("zeus"+date)1 IN TXT "IP1;IP2; … ; IP10;"
SHA3("zeus"+date)2 IN TXT "IP11;IP12; … ; IP20;"
SHA3("zeus"+date)10 IN TXT "IP91;IP92; … ; IP100;"
SHA3("artemis"+date) IN A 75.130.11.154

Fig. 8 Scenario 3—amplification attack

targeted machine. More importantly, since mobile devices are used for accessing Web sites or other network resources by their owners, the portion of DNS traffic originating from the botnet coordination is minimal compared with the whole traffic.

For creating the botnet and implementing the attacks, 12 Sony Xperia L Android Jelly Bean mobile devices were utilized in total. Each device has a dual-core 1GHz CPU and 1GB RAM and was connected to a wireless hotspot. On the other side, the victim was a desktop machine equipped with a dual-core 2.8 GHz CPU and 4 GB RAM connected to a 100 Mbps network interface. This machine acts as DNS authoritative NS having the DNSSEC extensions enabled. For each scenario and for each kind of attack, a botnet consisting of 1, 6 and 12 members has been created. In this way, one is able to infer the accumulative impact of the gradual activation of the mobile bots. Our tests demonstrate that every mobile bot is capable of executing three instances of the client attack script simultaneously without increasing the computational burden to a level that is perceptible by the owner of the mobile device. Naturally, the number of attack scripts a device can bear prior its user notices, it depends on the underlying hardware and OS. So, given that each bot reports its performance capacities to the C&C server, the latter may dynamically instruct the bot about the number of attack instances it should initiate. The concurrent execution of three instances of the attack script creates a stream of nearly 34 DNS queries per second on average or 2.33 KBps of outgoing DNS traffic. Also, this implies that in the ideal case each attacking bot is capable of flooding the target with 34 DNS responses per second. However, this data volume is multiplied by the amplification factor, which as it is discussed further down varies from 32.7 to 34.1 and it is solely dependent on the behavior of the chosen DNS forwarders. Consequently, a single attacking bot unleashes on average a stream of 76.9 KBps toward the victim. For compiling the pool of DNS forwarders to be used in the attack, we considered to examine the network IP blocks of Greece (details on this procedure are given in [5]). From the detected open forwarders, we kept about a number of 1.1K because those return back large DNSSEC records.

Figure 9 depicts the level of bandwidth consumption at the victim side during both attack variations for the case of first scenario and for 1, 6 and 12 bots, respectively. The flow of the inbound traffic remains similar for the rest of the scenarios. Furthermore, Table 1 details the average volume of the inbound network traffic for all three scenarios. The case of the single bot is implemented with the intend to accurately calculate the amplification factor of the DNS amplification.

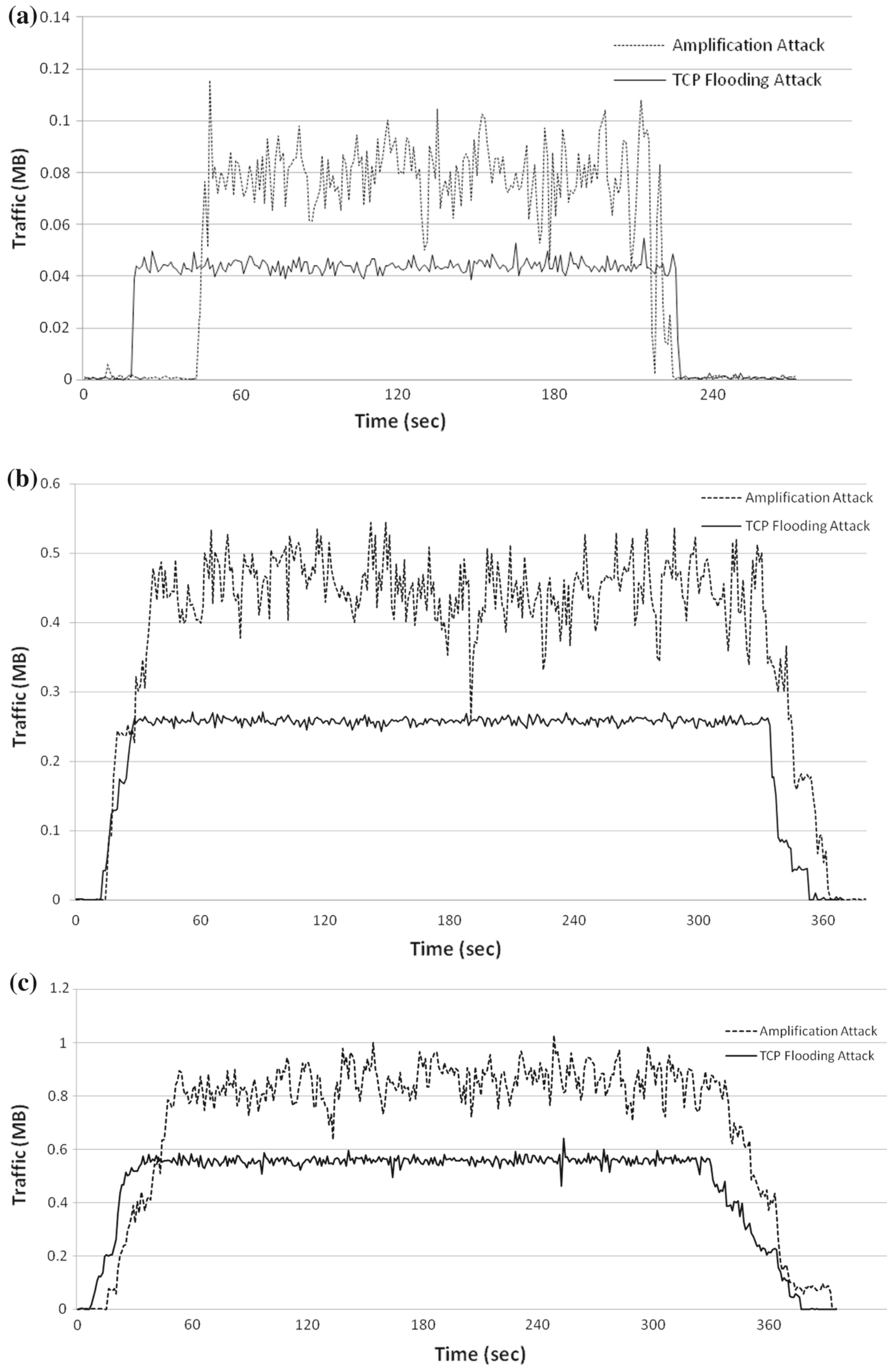


Fig. 9 Inbound traffic in MBps for both attack variations of Scenario 1. Botnet size: **a** 1 bot, **b** 6 bots, **c** 12 bots

Table 1 Inbound traffic (in MBps) per scenario proportional to the number of attacking bots

Number of attacking nodes	Scenario 1		Scenario 2		Scenario 3	
	Amplif	TCP flooding	Amplif	TCP flooding	Amplif	TCP flooding
1 (3 scripts)	0.075	0.042	0.074	0.042	0.074	0.042
6 (18 scripts)	0.45	0.27	0.43	0.25	0.37	0.22
12 (36 scripts)	0.75	0.5	0.85	0.58	0.80	0.57

During this experiment 6,000 DNS queries or 0.4 MB was dispatched by this single bot, given that in average our DNS request had a size of 70 bytes. On the other end, the victim received 13.64 MB of network traffic, which is translated to an amplification factor of 34.1. We can conclude that in total the DNS amplification creates larger volume of traffic toward the target because of the amplification nature of the attack. Also, a smoothness in the incoming traffic of the TCP flooding can be observed due to the direct connection among the bots and the victim. For DNS amplification, a variation with both upward and downward peaks is presented as the forwarders interpose in the communication.

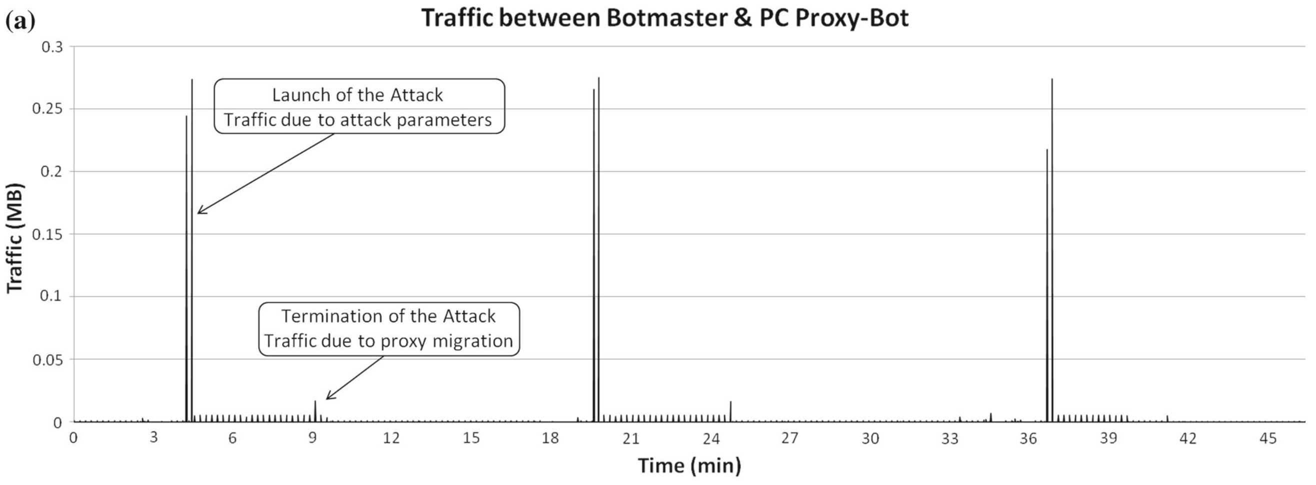
As expected, the second scenario reveals pretty much the same results with the previous one. Similarly to the first scenario, a single bot was able to send about 6,000 DNS queries or 0.4 MB and the target was flooded with 13.1 MB, which is reflected to an amplification factor of 32.75. As the current configuration facilitates the recording of the proxy's network traffic, we analyze the required HTTP transactions for the coordination of the botnet. As such a procedure imposes the usage of a sniffer app on the proxy side, it is normal to affect its operation as a bot too. However, with high confidence, it is asserted that the following analysis is identical for both scenarios, because the proxy operates exactly the same way. Between the botmaster and the current proxy, two types of transactions take place (Fig. 10a). The first one with high spikes happens during the attack phase, where the botmaster sends the attack parameters. The three upward pointing peaks observed in Fig. 10a occur during the time of the launch of the attack, that is, when the master transfers the list with the DNS forwarders. The second type exhibiting low peaks refers to the phase of migration. This traffic is mostly evident during the attack, as the master sends again the list of the forwarders to the new proxy. However, as the bot already posses that list, he instantly drops the connection. Moreover, between the HTTP proxy and a bot the only noticeable traffic occurs the time of the attack phase (Fig. 10b). At that moment, the proxy informs each connecting bot about the available list of DNS forwarders, thus creating a peak of 0.25 MBps. During the attack, there is a tiny amount of traffic as the proxy sends the list regardless if the bot has already receive it or not. This happens because the proxy does not record if a bot is already in possession of the list required for the amplification attack. Nevertheless, in the case the mobile bot holds by then the list, it instantly drops the incoming traffic.

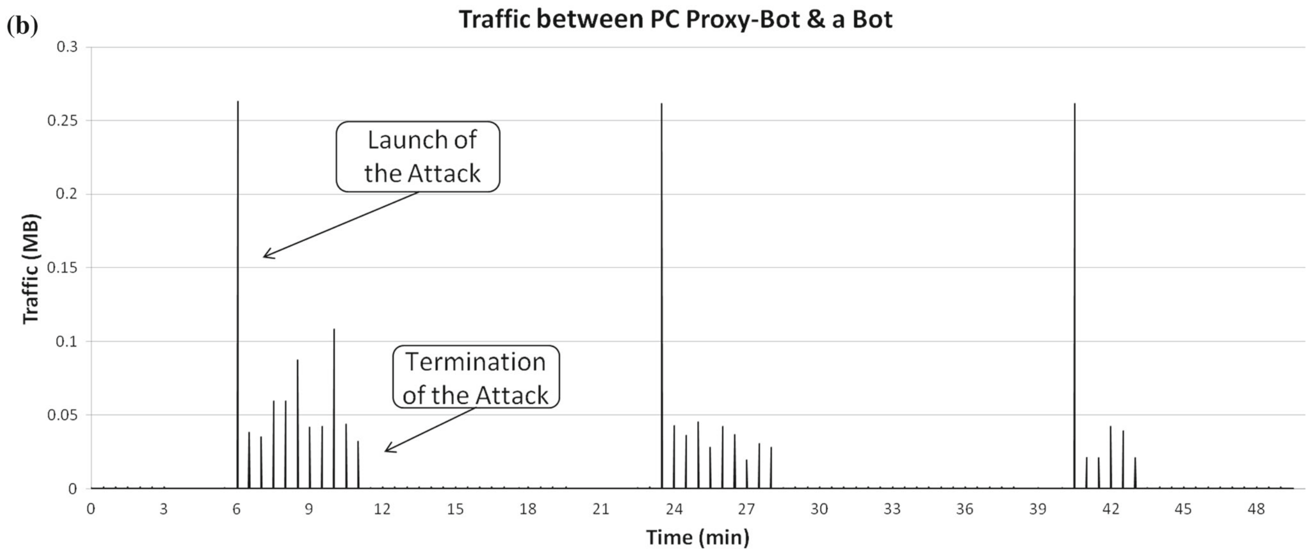
Similarly to the previous ones, the third scenario reveals nearly the same outcome on the victim side. A single bot was capable of sending about 6,000 DNS queries or 0.4 MB, and the target was flooded with 13.4 MB, which corresponds to an amplification factor of 33.5. For this scenario is interesting to examine the burden imposed on the DNS authoritative server for the coordination of the botnet. Specifically, we capture the DNS network traffic using a sniffer app. As it is observed from Fig. 10c, there are three upward pointing peaks in the network traffic which correspond to the launch of the amplification attack. Specifically, the initial bursts originate from the dynamic update of the zone with the attack parameters, while the following spikes are due to the requests by the various bots about the necessary DNS records during the attack phase.

5 Countermeasures

Following the rise of the botnets era, several researchers have investigated various methods and mechanisms for the detection and mitigation of bots. In the literature exist a great number of works studying this issue. However, their overwhelming majority concentrates on PC-based botnets and therefore may not be applicable to mobile ones. Precisely, there are certain impediments that hinder, if not block, the migration of solutions designed for PC-based botnets to the mobile realm [35]. That is, memory and storage constraint, lower processing power that apply to mobile devices in several cases, and mainly the limited battery capacity will hamper the adaptation of traditional countermeasures. Also in contrast to PC-based botnets, in the case of mobile botnets there are exploited a variety of channels for the propagation and coordination of the bots, including SMS/MMS, Bluetooth and Internet. More importantly, smartphone owners often neglect to adopt security measures on their devices, thus leaving them unprotected to ill-motivated entities [36]. In order to evade these intrinsic restrictions, various techniques have been proposed in the literature that aim to relieve the mobile device of a greater load due to the execution of demanding detection tasks.

Considering the above and for reasons of completeness, we deem necessary here to shortly present the most representative ones of them. Mainly, the efforts are focused on reverse engineering the code of the infected application (static [37,38] and dynamic/behavioral analysis [39]), so that the defenders become able to discover traces of the botmaster or the C&C infrastructure and propagation channels. In this setting, Feizollah et al. [40] examine various machine learning classifiers and argue that K-nearest neighbor algorithm gives the best results for anomaly-based mobile botnet detection. In [41], the authors utilize network forensics on the device's traces in order to construct the "normal" and "abnormal" behavior toward unveiling and tracing the source of security incidents pertaining to botnets. The work in [20] proposes to strategically plant special honeypots called honeyphones in crowded places with open Bluetooth or Wi-Fi connections waiting for attacks. This way, the sentinels will be able to analyze on-the-fly malicious code and collect valuable information for the operation and the structure of the botnet. Another approach of detection is to provide malware





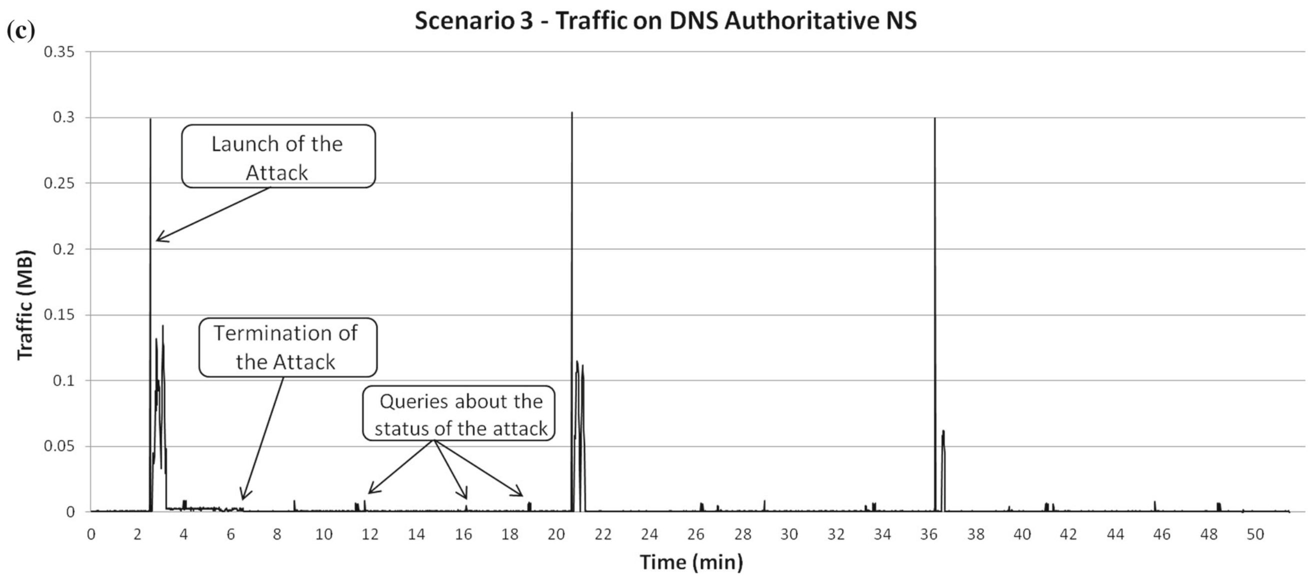


Fig. 10 Network traffic generated due to botnet coordination. **a** Network traffic between botmaster and PC proxy. **b** Network traffic between proxy and a single bot. **c** DNS traffic for the botnet coordination

analysis in the Cloud. For example, Oberheide et al. [42] and Portokalidis et al. [43] examine suspicious executables of the devices in cloud services. Therefore, they avoid to consume the valuable resources of the device and annoy its owner. More recently, Damopoulos et al. [44] present a solution that combines both host- and cloud-based malware analysis for the Android platform.

It is to be stressed that countermeasures and repelling strategies against DNS amplification and other forms of (D)DoS attacks have been intentionally omitted from this section as they are already well explained in other works, including those in [5,33].

6 Conclusions

Undoubtedly, botnets present compelling new challenges for the Internet community in general and security experts in particular. Lately, mobile botnets are rapidly becoming an emerging and increasingly evolving threat mostly due to the difficulty of being detected and neutralized. In this context, this paper introduces a number of novel C&C architectures that may be exploited by potential aggressors to deploy well-hidden arsenals of mobile bots. We detail on the architectural components of each presented scenario and analyze the various phases of their operation. Also, to assess the effectiveness of the introduced C&C architectures, we unleash two real-world attacks and measure their impact on both the target machine and the C&C costs produced. The results reveal catastrophic power combined with negligible volume of C&C traffic. Naturally, the latter quality brings along the advantage of anonymity on the botherder side. As a future work, we shall consider alternative covert channels for bot communication. Specifically, our orientation is toward well-established, text-based signaling protocols such as the session initiation protocol (SIP) one used in Voice over IP (VoIP) [45]. Additionally, our intention is to employ a larger population of mobile devices as bots ensuring a more accurate appraisal of the proposed architectures. This would even allow us to record the accumulative impact of a mobile botnet in DDoS incidents

References

1. Ianelli, N., Hackworth, A.: Botnets as a vehicle for online crime. In: *CERT Coordination Center* (2005)
2. Nazario, J., Holz, T.: As the net churns: fast-flux botnet observations. In: *3rd International Conference on Malicious and Unwanted Software, 2008. MALWARE 2008*, pp. 24–31 Oct (2008)
3. Holz, T., Gorecki, C., Rieck, K., Freiling, F.C.: Measuring and detecting fast-flux service networks (2008). *In: Symposium on Network and Distributed System Security - NDSS*(2008)
4. Antonakakis, M., Perdisci, R., Nadji, Y., Vasiloglou, N., Abu-Nimeh, S., Lee, W., Dagon, D.: From throw-away traffic to bots: detecting the rise of DGA-based Malware. In: *Proceedings of the 21st USENIX Security Symposium (USENIX Security 12)*, pp. 491–506, USENIX, Bellevue, WA (2012)
5. Anagnostopoulos, M., Kambourakis, G., Kopanos, P., Louloudakis, G., Gritzalis, S.: DNS amplification attack revisited. Comput. Secur. **39**(Part B), 475–485 (2013)
6. Gu, G., Perdisci, R., Zhang, J., Lee, W. et al.: BotMiner: clustering analysis of network traffic for protocol-and structure-independent botnet detection. In: *Proceedings of the 17th Conference on Security Symposium, USENIX Association*, pp. 139–154 (2008)
7. Pinto, R.C.G., Silva, S.S.C., Silva, R.M.P., Salles, R.M.: Botnets: a survey. Comput. Netw. **57**(2), 378–403 (2013)
8. Binsalleeh, H., Kara, A.M., Youssef, A., Debbabi, M.: Characterization of covert channels in DNS. In *6th International Conference on New Technologies, Mobility and Security (NTMS), 2014*, pp. 1–5, March (2014)
9. Dietrich, C.J., Rossow, C., Freiling, F.C., Bos, H., van Steen, M., Pohlmann, N.: On botnets that use DNS for command and control. In *Seventh European Conference on Computer Network Defense (EC2ND), 2011*, pp. 9–16, Sept (2011)
10. Wang, P., Wu, L., Aslam, B., Zou, C.C.: A systematic study on peer-to-peer botnets. In *Proceedings of 18th International Conference on Computer Communications and Networks (ICCN2009), IEEE*, pp. 1–8. (2009)
11. Wang, P., Sparks, S., Zou, C.C.: An advanced hybrid peer-to-peer botnet. IEEE Trans. Dependable Secure Comput. **7**(2), 113–127 (2010)
12. Cooke, E., Jahanian, F., McPherson, D.: The zombie roundup: understanding, detecting, and disrupting botnets. In *Proceedings of the USENIX Steps to Reducing Unwanted Traffic on the Internet (SRUTI05) Workshop*, vol. 39, p. 44 (2005)
13. Damopoulos, D., Kambourakis, G., Anagnostopoulos, M., Gritzalis, S., Park, J.H.: User privacy and modern mobile services: are they on the same path? Pers. Ubiquitous Comput. **17**(7), 1437–1448 (2013)
14. Apvrille, A.: Symbian worm yxes: towards mobile botnets? J. Comput. Virol. **8**(4), 117–131 (2012)
15. Porras, P., Saidi, H., Yegneswaran, V.: An analysis of the iKee. b iPhone botnet. In *Security and Privacy in Mobile Information and Communication Systems*, pp. 141–152. Springer (2010)
16. Pieterse, H., Olivier, M.S.: Android botnets on the rise: trends and characteristics. In: *Information Security for South Africa (ISSA)*, 2012, pp. 1–5, Aug (2012)
17. Zhou, Y., Jiang, X.: Dissecting android malware: characterization and evolution. In *IEEE Symposium on Security and Privacy (SP), 2012*, pp. 95–109, May (2012)
18. La Polla, M., Martinelli, F., Sgandurra, D.: A survey on security for mobile devices. IEEE Commun. Surv. Tutor. **15**(1), 446–471 (2013)
19. Singh, K., Sangal, S., Jain, N., Traynor, P., Lee, W.: Evaluating bluetooth as a medium for botnet command and control. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 61–80. Springer (2010)
20. Hua, J., Sakurai, K.: A SMS-based mobile botnet using flooding algorithm. In: *Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication*, pp. 264–279. Springer (2011)
21. Mulliner, C., Seifert, J.P.: Rise of the iBots: owning a telco network. In: *5th International Conference on Malicious and Unwanted Software (MALWARE) 2010, IEEE*, pp. 71–80. (2010)
22. Xiang, C., Binxing, F., Lihua, Y., Xiaoyi, L., Tianning, Z.: Andbot: towards advanced mobile botnets. In: *Proceedings of the 4th USENIX conference on Large-scale exploits and emergent threats*, p. 11. USENIX Association, Berkeley, CA, USA (2011)

23. Faghani, M.R., Nguyen, U.T.: Socellbot: a new botnet design to infect smartphones via online social networking. In: *25th IEEE Canadian Conference on Electrical Computer Engineering (CCECE), 2012*, pp. 1–5, April (2012)
24. Zhao, S., Lee, P.P.C., Lui, J.C.S., Guan, X., Ma, X., Tao, J.: Cloud-based push-styled mobile botnets: a case study of exploiting the cloud to device messaging service. In: *Proceedings of the 28th Annual Computer Security Applications Conference, ACSAC '12*, pp. 119–128, ACM, New York, NY, USA (2012)
25. Hasan, R., Saxena, N., Haleviz, T., Zawoad, S., Rinehart, D.: Sensing-enabled channels for hard-to-detect command and control of mobile devices. In: *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, ASIA CCS '13*, pp. 469–480, ACM, New York, NY, USA (2013)
26. Dagon, D., Zou, C.C., Lee, W.: Modeling botnet propagation using time zones. In *Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS'06)*, vol. 6, pp. 2–13 (2006)
27. Felt, A.P., Finifter, M., Chin, E., Hanna, S., Wagner, D.: A survey of mobile malware in the wild. In: *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM '11*, pp. 3–14, ACM, New York, NY, USA (2011)
28. Knysz, M., Hu, X., Zeng, Y., Shin, K.G.: Open WiFi networks: lethal weapons for botnets? In: *INFOCOM, 2012 Proceedings IEEE*, pp. 2631–2635, March (2012)
29. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: RFC 4033: DNS security introduction and requirements. http://www.ietf.org/rfc/rfc4033.txt (2005)
30. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: RFC 4034: Resource records for the DNS security extensions. http://www.ietf.org/rfc/rfc4034.txt (2005)
31. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: RFC 4035: Protocol modifications for the DNS security extensions. http://www.ietf.org/rfc/rfc4035.txt (2005)
32. Anagnostopoulos, M., Kambourakis, G., Konstantinou, E., Gritzalis, S.: DNSSEC vs. DNSCurve: A Side-by-Side Comparison, p. 201. IGI Global (2012)
33. Rijswijk-Deij, van R., Sperotto, A., Pras, A.: DNSSEC and its potential for DDoS attacks: a comprehensive measurement study. In: *Proceedings of the 2014 Conference on Internet Measurement Conference, IMC '14*, pp. 449–460, ACM, New York, NY, USA (2014)
34. Scapy project. http://www.secdev.org/projects/scapy/
35. Eslahi, M., Salleh, R., Anuar, N.B.: MoBots: a new generation of botnets on mobile devices and networks. In: *IEEE Symposium on Computer Applications and Industrial Electronics (ISCAIE), 2012*, pp. 262–266, Dec (2012)
36. Reinfelder, L., Benenson, Z., Gassmann, F.: Differences between android and iphone users in their security and privacy awareness. In: Eckert, C., Katsikas, S.K., Pernul, G., (eds.), Trust, Privacy, and Security in Digital Business, vol. 8647 of Lecture Notes in Computer Science, pp. 156–167. Springer International Publishing (2014)
37. Schmidt, A.-D., Bye, R., Schmidt, H.-G., Clausen, J., Kiraz, O., Yuksel, K.A., Camtepe, S.A., Albayrak, S.: Static analysis of executables for collaborative malware detection on android. In: *IEEE International Conference on Communications, 2009. ICC '09*, pp. 1–5, June (2009)
38. Abdullah, Z., Saudi, M.M., Anuar, N.B.: Mobile botnet detection: proof of concept. In: *IEEE 5th Control and System Graduate Research Colloquium (ICSGRC), 2014*, pp. 257–262, Aug (2014)
39. Burguera, I., Zurutuza, U., Nadjm-Tehrani, S.: Crowdroid: behavior-based malware detection system for android. In: *Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM '11*, pp. 15–26, ACM, New York, NY, USA (2011)
40. Feizollah, A., Anuar, N.B., Salleh, R., Amalina, F., Maarof, Rauf Ridzuan, Shamshirband, Shahaboddin: A study of machine learning classifiers for anomaly-based mobile botnet detection. Malays. J. Comput. Sci. **26**(4), 251–265 (2014)
41. Vural, I., Venter, H.: Mobile botnet detection using network forensics. In: *ArneJ. Berre, Asuncion Gomez-Perez, Kurt Tutschku, and Dieter Fensel, editors, Future Internet—FIS 2010*, vol. 6369 of Lecture Notes in Computer Science, pp. 57–67. Springer, Berlin (2010)
42. Oberheide, J., Cooke, E., Jahanian, F.: CloudAV: N-version antivirus in the network cloud. In *17th USENIX Security Symposium*, pp. 91–106 (2008)
43. Portokalidis, G., Homburg, P., Anagnostakis, K., Bos, H.: Paranoid android: versatile protection for smartphones. In: *Proceedings of the 26th Annual Computer Security Applications Conference, ACSAC '10*, pp. 347–356, ACM, New York, NY, USA, (2010)
44. Damopoulos, D., Kambourakis, G., Portokalidis, G.: The best of both worlds: a framework for the synergistic operation of host and cloud anomaly-based IDS for smartphones. In: *Proceedings of the Seventh European Workshop on System Security, EuroSec '14*, pp. 6:1–6:6, ACM, New York, NY, USA (2014)
45. Tsiatsikas, Z., Anagnostopoulos, M., Kambourakis, G., Lambrou, S., Geneiatakis, D.: Hidden in plain sight. SDP-Based covert channel for botnet communication. In: Fischer-Hubner, S., Lambrinoudakis, C., Lopez, J. (eds.), Trust, Privacy and Security in Digital Business, vol. 9264 of Lecture Notes in Computer Science, pp. 48–59. Springer International Publishing (2015)