Computers & Security 140 (2024) 103768

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

MONDEO-Tactics5G: Multistage botnet detection and tactics for 5G/6G networks

Bruno Sousa [a,*], Duarte Dias [a], Nuno Antunes [a], Javier Cámara [b], Ryan Wagner [c], Bradley Schmerl [c], David Garlan [c], Pedro Fidalgo [d]

[a] University of Coimbra, CISUC, Department of Informatics Engineering, Pinhal de Marrocos, Coimbra, 3030-290, Portugal
[b] ITIS Software, Universidad de Málaga, Spain
[c] S3D, Carnegie Mellon University, United States of America
[d] Mobileum, ISCTE-IUL, Portugal

ARTICLE INFO

Keywords:
Botnet
DDoS
Command Control server
Tactics

ABSTRACT

Mobile malware is a malicious code specifically designed to target mobile devices to perform multiple types of fraud. The number of attacks reported each day is increasing constantly and is causing an impact not only at the end-user level but also at the network operator level. Malware like FluBot contributes to identity theft and data loss but also enables remote Command & Control (C2) operations, which can instrument infected devices to conduct Distributed Denial of Service (DDoS) attacks. Current mobile device-installed solutions are not effective, as the end user can ignore security warnings or install malicious software. This article designs and evaluates **MONDEO-Tactics5G** - a multistage botnet detection mechanism that does not require software installation on end-user devices, together with tactics for 5G network operators to manage infected devices. We conducted an evaluation that demonstrates high accuracy in detecting FluBot malware, and in the different adaptation strategies to reduce the risk of DDoS while minimising the impact on the clients' satisfaction by avoiding disrupting established sessions.

1. Introduction

Mobile malware is an artifact with malicious code designed specifically to target mobile devices with the goal of undertaking multiple types of fraud, either by gaining access to private data or disrupting organizations and businesses.

Statistics show that around 437 million malware attacks were detected in 2023 (Kaspersky, 2023). Although there is a general decline in the number of attacks, mobile malware continues to evolve, including new capabilities. Attacks are becoming more complicated and difficult to detect. Malware can now be easily downloaded from official app stores, teasing users more often (drWeb, 2023). Malware causes financial losses (Association, 2021) and affects 5G and future networks (Ahmed et al., 2023). These attacks have a large impact on end users, in terms of financial damages, identity theft and loss of privacy. Mobile Malware also has an impact on the services provided by Mobile Network Operators (MNOs), where 5G and beyond enable an increased number of connected devices, along with a higher bandwidth that facilitates data exfiltration.

FluBot is one of the threats with more impact in terms of malware that uses the Domain Name System (DNS) to interact with Command and Control (C2) servers, having an impact on end devices and network operators. Through FluBot, malware deployed on mobile devices can be instrumented via the C2 channel to act as a botnet, enabling coordinated attacks such as distributed denial of service (DDoS). Other types of botnets exist, such as Mirai, Reaper, Android.Pandora.2 and variants that appear from time to time and can be instrumented to carry out DDoS attacks (Wazzan et al., 2021; drWeb, 2023).

Several botnet detection techniques are available, including honeypot analysis, communication signatures (e.g., using whitelists and blacklists), deep learning techniques based on neural networks, reinforcement learning, convolutional neural networks, statistical analysis, distributed approaches, and also using combination methods (Xing et al., 2021; Ahmed et al., 2023). Or they can also take advantage of defense approaches like Moving Target Defense (MTD) (Almutairi et al., 2020; Wang et al., 2014). These approaches use multiple techniques to detect botnets but require the deployment of agents on the mobile de-

[*] Corresponding author.
E-mail address: bmsousa@dei.uc.pt (B. Sousa).

https://doi.org/10.1016/j.cose.2024.103768
Received 30 November 2023; Received in revised form 28 January 2024; Accepted 9 February 2024
Available online 21 February 2024

vices and on the network side like honeypot agents. Accuracy in botnet detection depends on the combination of multiple techniques (Wang et al., 2020). Botnet detection can take advantage of DNS information such as the number of packets per second (PPS), the average payload size, or the type of request. Using these features, clustering techniques (e.g., kNN) can be applied to identify abnormalities in communication flows (i.e., excessive requests).

Network operators cannot rely on the installation of software on mobile devices or IoT devices, to guarantee the security of their mobile networks. In fact, the solution to install software on the end devices would not work for roaming users. Additionally, detection and mitigation processes must also be synchronized, so efficient tactics can be devised according to the level of security risk (e.g., considering the number of infected devices).

This article proposes MONDEO-Tactics5G, which is a Multistage Botnet Detection and Tactics for 5G and beyond networks. The approach mitigates malware due to its high impact through an efficient detection technique and through custom tactics that can be integrated into the existing controls of mobile network operators. MONDEO-Tactics5G is based on three design principles: First, allow for full control of operators without the need to deploy agents/software on end users' devices. Second, facilitate integration with core services, such as DNS servers or User Plane Function (UPF), in the infrastructures of network operators. Third, support custom policies/tactics that can be integrated into the security mechanisms employed by network operators, like Access Control Lists (ACL).

The MONDEO-Tactics5G architecture includes two main components: (a) detection and (b) tactics, both designed to be integrated in 5G network architectures. The detection component consists of four steps: (1) blacklist/whitelist analysis; (2) query rate analysis; (3) Domain Generated Algorithm (DGA) analysis; (4) machine learning. The output of botnet detection is the ability to detect infected devices and limit their footprint within the network. Among a high volume of requests, the C2 server(s) need to be identified after the detection of a botnet.

The tactics component is concerned with policies to mitigate/eliminate the malware threat. The tactics in MONDEO-Tactics5G can be diverse, including the isolation and quarantining of affected servers and devices and the introduction of CAPTCHAs to distinguish humans from bots. The choice of tactics is sensitive to both the context of the system, including load on customer service personnel who may need to interact with affected users, and the overall utility and priorities of the business goals of clients.

The remainder of this paper is organized as follows. Section 2 provides the contributions of MONDEO-Tactics5G, Section 3 introduces the background and motivation for this work, Section 4 details the design of MONDEO-Tactics5G. Section 5 delineates the evaluation methodology, while Section 6 provides evaluation results and Section 7 concludes the paper.

2. Contributions of MONDEO-Tactics5G

There is a need for solutions to detect mobile malware in 5G networks and beyond (Salahdine et al., 2023), as it greatly affects end users (Association, 2021) and network operators, leading to inefficient resource utilization (Ahmed et al., 2023; drWeb, 2023).

MONDEO-Tactics5G establishes a foundational knowledge on botnet detection and the application of tactics for botnet mitigation in 5G networks and beyond. The use case validation scenario includes a 5G network with DNS server(s), User Plane Function (UPF) for traffic analysis and connected to the MONDEO Detection component, and the Policy Control Function (PCF) connected to the MONDEO Tactics component, at the core of the network. The access network includes benign and infected mobile devices.

The contributions of MONDEO-Tactics5G are: (1) efficient botnet detection mechanisms that can be integrated with the User Plane Function (UPF) in 5G networks, or DNS servers; (3) MONDEO-Tactics5G supports the identification of C2 server(s), often obfuscated in the high volume of DNS requests; (3) MONDEO-Tactics5G includes three tactics like quarantining mobile devices, blackholing IP addresses of C2 servers, and CAPTCHAS working at connection level. Such tactics are formulated according to a utility function, considering operator and customer interests, and can be instrumented in 5G networks through the Policy Control Function (PCF).

3. Mobile malware in telecom fraud

5G rethinks and redesigns how the network is built and managed by introducing emerging use cases and business models, affecting not only consumers, but also enterprises and industries. In 5G, most subscribers will not be consumers as before – the bulk of 5G will consist of IoT devices with different needs from human subscribers (e.g., Smartmeters, environmental sensors). In fact, even in terms of IoT, and depending on the use case, IoT devices can actually have totally different behaviours (e.g., a Smartmeter and a connected car). 5G can be split into 3 different use cases, two of them more related to IoT like Ultra Reliable Low Latency (URLLC) and massive Machine-Type Communications (mMTC), while the evolved Mobile Broadband (eMBB) is more related with generic services and is intended for human subscribers. This mixture of patterns, as well as different use cases, changes the scope of suspicious behaviour, leading to new fraud models. Banking and Gaming Trojans are good examples of how malware targets specific business areas and how future Trojans can evolve to specific 5G use cases.

3.1. 5G impact on malicious activities

5G technology will impact fraud, as already outlined in reports with the participation of Mobile Network Operators (Association, 2021; Ahmed et al., 2023). The 5G impact is promoted by the use of Artificial Intelligence to perpetrate fraud and avoid detection, termed Smart Fraud; the massive increase in the number of connected devices also contributes to the effectiveness of DDoS attacks. New Radio (NR) enhancements introduced in 5G enable high-density connections, allowing more connections per unit area with higher data rates, compared to previous versions like 4G (Tian and Lin, 2022).

An increase in fraud activity, besides being facilitated by 5G networks, is also associated with other factors, like the increase in cross-industry-targeted social engineering schemes, the increase in financial services impersonation frauds using stolen credentials from data breaches, and the facilitated use of faceless transaction portals (e.g., appear as benign websites) to commit Subscription Fraud and Account Takeovers.

The common factor in all of the above is compromised credentials and sensitive personal data obtained through data breaches and mobile malware that leads to identity theft. While the financial impact is extremely high, there are other major impacts to the MNOs, such as customer complaints, low customer loyalty and trust, churn, dispute costs, artificial increase of traffic (SMS/Voice/Data), impact on network performance, and interconnect costs, among others.

Another concerning fact is the multitude of other frauds that can be perpetrated through infected phones that so far have not been exploited by the fraudsters and the volume of scam messages that can infect phones. Specific to the telecom domain, most malware already has enough privileges on infected devices to make calls or send SMS's, which allows them to perform International Revenue Share Fraud (IRSF), as well as other types of fraud which are monetized through voice calls or SMS's (Flinders, 2021).

3.2. Infection and spreading

SMS frequently serves as the main attack vector (smishing). Often a text message serves as bait and replicates legitimate aid programs designed by governments, such as COVID vaccination programs. Other typical types of smishing are gifts, missed deliveries, bank fraud warnings, invoices, order confirmations, or customer support issues. The message contains a call-to-action and a malicious URL to follow, which will drive the victim to a recent and convincing look-alike page for the entity the message claimed to be from. This website will then ask for personal information, such as credit card information, credentials, date of birth, or to download malicious software, typically disguised as a legitimate application, allowing multiple types of attacks once installed. Once a device is infected, it will join a botnet and execute commands sent by C2 server(s). Initial targets are obtained by leaked information, such as Facebook leaked data. Then contact lists are extracted from infected devices to the C2 Servers, enabling them to unleash further attacks.

Although blocking access to known fraudulent websites seems like a common sense approach to protect subscribers, it is in many ways ineffective. Not only is it easy to avoid blacklists, cybercriminals also simply need their malicious URL running for less than 13 hours, for an effective bulk campaign (Elie Bursztein, 2019). Attacks occur quickly and use cheap domains on websites that were registered the same day or within days of the attack itself, allowing enough time to verify if a site is safe or dangerous.

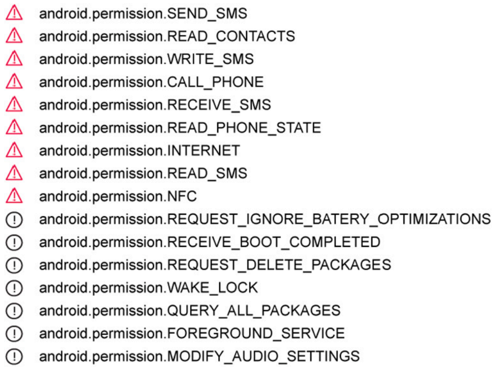
5G and beyond facilitate smishing campaigns and other types of mobile malware due to reduced network latency and increased bandwidth to exfiltrate data from mobile devices (Salahdine et al., 2023).

3.3. Botnets and DNS

A botnet is a network of hijacked devices compromised by various forms of remote code installation and controlled remotely by a hacker. Typically, these devices execute commands sent by a botmaster either through a distributed or centralized C2 server, a peer-to-peer network, or any other management channel (e.g., IRC, HTTPS) which allows them to transfer commands to the bot. Botmasters will often hide their identity via proxies, TOR, to disguise their IP address from detection by investigators and law enforcement. These botnets can have tens of millions of devices (e.g., Zeus, Storm, or Mariposa) and belong to different botnet families associated with click fraud, banking fraud, DDoS, crypto miners, or ransomware (Ahmed et al., 2023).

Although control of infected devices can be done using different protocols, 85% of malware uses the DNS protocol for malware delivery, Command and Control (C2), or data exfiltration (paloalto Networks, 2022; Lyu et al., 2022; Principi et al., 2023). The ubiquitous nature of DNS, the high traffic volume, and often overlooked attack surfaces are the primary reasons that malware uses DNS to hide malicious activity. Three different attacks stand out:

- **Malware Using DNS for C2**. Once a device is infected, the system sends a DNS request back to the attacker's control server. In this way, the infected device becomes part of the botnet. Depending on the malware installed on the device, it will receive and execute commands associated with fraud or cyberattacks.
- **Malware Using Domain Generation Algorithms (DGAs)**. DGAs randomly generate a large number of distinct domain names, which do not need to be registered. In these cases, attackers may use only one to bypass traditional security controls, such as block lists or Web reputation filtering.
- **DNS Tunneling** attackers encode their payloads (e.g., data theft or C2) into small chunks within DNS requests to bypass security controls. Once a device is compromised, it sends a request inside the DNS traffic to a DNS server (controlled by the cybercriminal), which is instructed to connect to the cybercriminal server, opening a channel through which data is transmitted.



⚠ android.permission.SEND_SMS
⚠ android.permission.READ_CONTACTS
⚠ android.permission.WRITE_SMS
⚠ android.permission.CALL_PHONE
⚠ android.permission.RECEIVE_SMS
⚠ android.permission.READ_PHONE_STATE
⚠ android.permission.INTERNET
⚠ android.permission.READ_SMS
⚠ android.permission.NFC
(!) android.permission.REQUEST_IGNORE_BATERY_OPTIMIZATIONS
(!) android.permission.RECEIVE_BOOT_COMPLETED
(!) android.permission.REQUEST_DELETE_PACKAGES
(!) android.permission.WAKE_LOCK
(!) android.permission.QUERY_ALL_PACKAGES
(!) android.permission.FOREGROUND_SERVICE
(!) android.permission.MODIFY_AUDIO_SETTINGS

Fig. 1. FluBot permissions on Android phones.

3.4. Fraud realization

Malware on infected devices acquires permissions that allow them to have complete control over most of the system features and to perform almost any task on behalf of cybercriminals. Fig. 1 illustrates the case of FluBot, a Banking Trojan, and the associated permissions once installed, as well as the possible fraud scenarios that can be performed with the associated permissions.

From the above, it is possible to observe multiple fraud scenarios that can be executed by the fraudster, once the mobile malware is installed and the infected device is part of a botnet. With all the permissions, the apps on the infected device allow the frausters to detect crypto wallets and banking apps, but also to remove any apps that detect and prevent them from running, like antivirus software, and can even run unnoticed to the end user.

In a 5G scenario, as depicted in Fig. 2, mobile devices are compromised through a smishing campaign ***(1)***, in this case targeting users with specific banking apps in some countries. The SMS message would require the user to install an app ***(2)*** belonging to a package delivery provider (e.g. FedEx, DHL), to track or reschedule a delivery. Once installed, the device is infected with FluBot malware and communicates with its C2 server via DNS Tunelling over HTTPS (DoH). FluBot ***(3)*** uses a domain generation algorithm (DGA) to enable it to communicate with its C2 server. In the latest versions, additional seeds are downloaded from the C2 server to generate more domains. Malware is commonly spread through SMS messages to contacts on an infected device ***(4)*** and executes fraud on behalf of the fraudster.

4. Multistage botnet detection and tactics for 5G

This section describes MONDEO-Tactics5G, a multistage botnet detection and tactic for 5G and beyond networks.

4.1. Use case

The use case includes a 5G network with DNS server(s), User Plane Function (UPF) for traffic analysis and connected to the MONDEO Detection component, and the Policy Control Function (PCF) connected to the MONDEO Tactics component, at the core of the network. The access network includes benign and infected mobile devices. Infected mobile devices are those that have malware running, which was installed by SMS phishing, as described in Section 3.2.

Given this use case, the following requirements, for an efficient detection of malware in 5G and beyond networks are formulated (Zhan et al., 2022):

R1 Perform flexible and scalable data collection in existing systems of mobile network operators;
R2 Avoid the installation of agents, software in mobile devices;

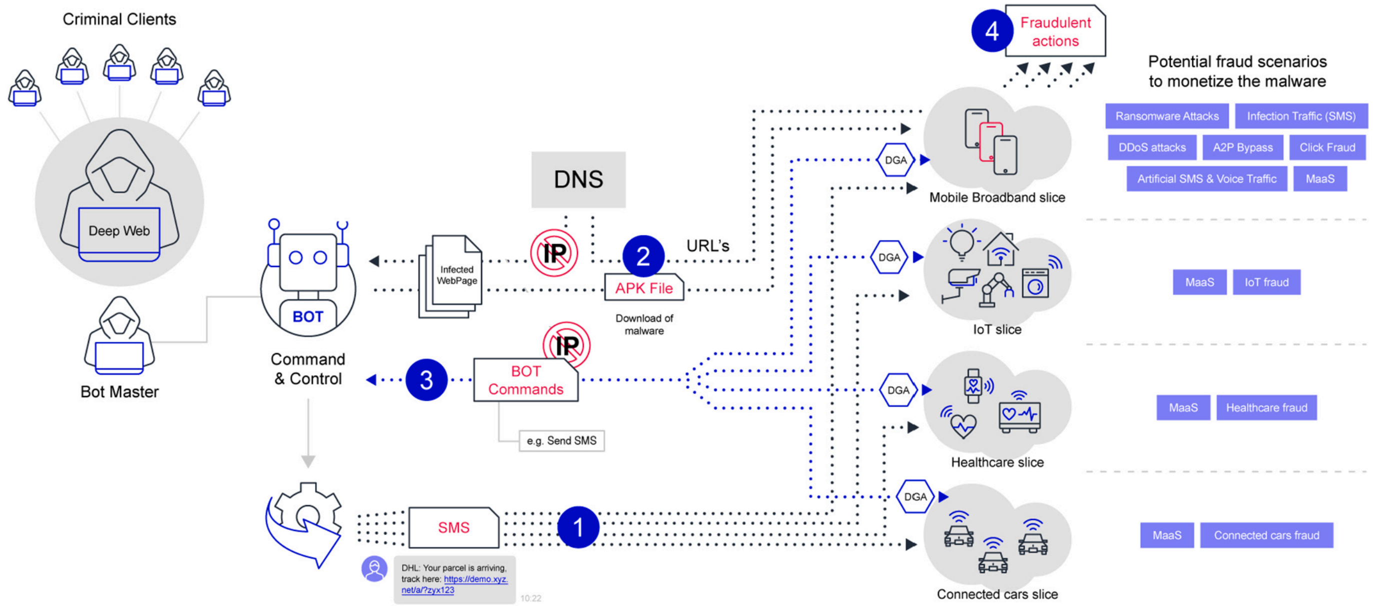


Fig. 2. Mobile Malware affecting different network slices in a 5G network.

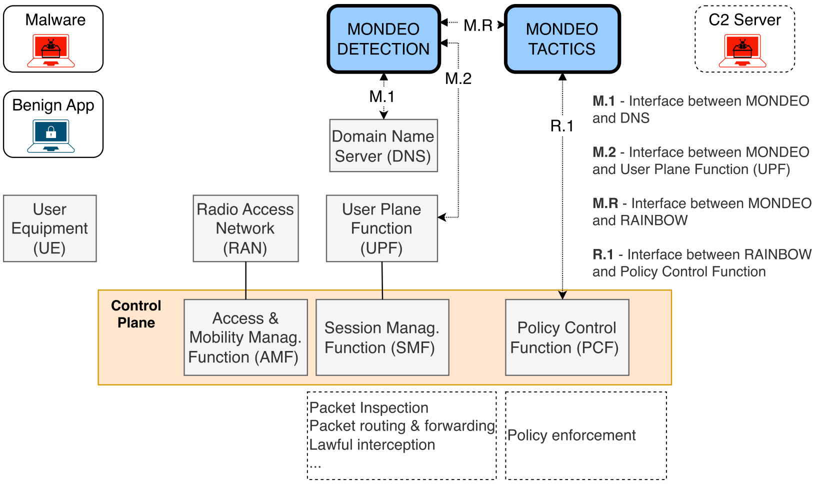

Malware
Benign App
MONDEO DETECTION
M.R
MONDEO TACTICS
C2 Server
M.2
M.1
Domain Name Server (DNS)
R.1
M.1 - Interface between MONDEO and DNS
M.2 - Interface between MONDEO and User Plane Function (UPF)
M.R - Interface between MONDEO and RAINBOW
R.1 - Interface between RAINBOW and Policy Control Function
User Equipment (UE)
Radio Access Network (RAN)
User Plane Function (UPF)
Control Plane
Access & Mobility Manag. Function (AMF)
Session Manag. Function (SMF)
Policy Control Function (PCF)
Packet Inspection
Packet routing & forwarding
Lawful interception
...
Policy enforcement

Fig. 3. MONDEO architecture.

R3 Enable distinct tactics according to the probability of a security event or severity;

R4 Support multiple decision layers considering performance issues. For instance, support blacklist(s) for immediate denial.

4.2. Overall architecture

The architecture depicted in Fig. 3 illustrates the different components of the multistage botnet detection and tactics.

The MONDEO detection component interacts with domain name servers through the *M.1* interface and with the User Plane Function (UPF) through the *M.2* interface, for instance to receive traffic sent by the mobile device. Information regarding DNS queries and replies is provided by the DNS Server to MONDEO, through *M.1*, this is relevant to obtain DNS requests and DNS replies. The interaction with the UPF is required to detect flows communicating with the C2 server, for instance, to detect HTTPS requests. The *M.R* interface allows the exchange of information between MONDEO and TACTICS components, so that upon detection of malware by MONDEO, the required information is provided to implement mitigation tactics. Such information can include the identification of the source (i.e., User Equipment) and information about the C2 server that has been detected, within a certain confidence level.

4.3. MONDEO - botnet detection component

The overall perspective of MONDEO includes details regarding the detection mechanism, and practical aspects that were chosen to implement a valid and deployable proof of concept.

MONDEO detection component runs on a microservice-based architecture, providing the detection service through a RESTful API. Through the API, it is possible to exchange the required data, where each request is treated individually and processed efficiently through parallelization, leveraging the support of multiple threads in Python Flask.

Botnet detection in the multistage MONDEO pipeline is condensed into four phases (Fig. 4): 1- Whitelisting/Blacklisting; 2- Query Rate Analysis; 3- DGA Evaluation; and 4- Machine Learning Evaluation. In each phase the output can be a classification of Benign (passed) or Infected (flagged), the following phase occurs if no classification is performed. Phase 2 - Query Rate Analysis flags as Infected the requests that may have anomalies in the ratios of the DNS requests. The output of DGA and ML evaluation in the last phases allows the implementa-

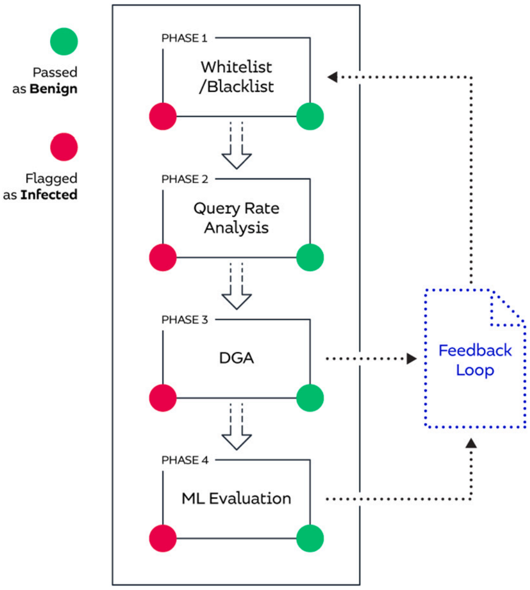


Fig. 4. MONDEO overall stages and feedback loop.

tion of a feedback loop. This feedback can be used to manage data in whitelists and blacklists, depending on the classification output, benign for whitelists and infected for blacklists. A 5G UPF function uses the information on the whitelist/blacklists to accept or deny traffic flows, accordingly.

The design of MONDEO has considered deployment, accuracy, and efficiency concerns. Regarding deployment, the choice for multistage analysis in MONDEO is for DNS requests, whereas other approaches rely on DNS replies/responses (Singh et al., 2019). This is relevant for mobile network operators, or ISPs, as they control the DNS infrastructure in their networks. This allows for parallelization of botnet detection with core services like DNS. The concern of efficiency leads to optimization in the analysis process, in terms of maximizing the speed of analysis, making it so that packets can be either discarded or accepted in the early phases of the pipeline (preferably in phase 1).

4.3.1. Phase 1 - whitelisting/blacklisting

Whitelists or blacklists are simple lists that can hold any basic data structure to represent information. The options described above take into account a complete (1-1) direct match. One can, however, implement a similar technique, but only using partial matches. Using the concept of Free Level Domain (FLD) and at the cost of low precision and processing time, if only part of the domain is analyzed, the lists will be shorter. For example, instead of having multiple entries for each of the Google applications, one can just whitelist all the "*.google.com*" domains.

In our proof-of-concept implementation we took a simpler approach: as the whitelist is short, we used a traditional linear search with a complexity of $O(n)$.

The Whitelist and Blacklist can be enhanced in MONDEO by leveraging the feedback loop, which uses the output of phase 3 - DGA and phase 4 - ML model. The feedback loop provides the required information for whitelists by indicating benign DNS requests and clients, and blacklists indicating DNS requests that are not resolvable. Thus, the feedback loop allows phase 1 to be approved by retroactively adding or removing domains from the lists. This could be achieved by disseminating the events of the feedback loop in Security Information Event Managers (SIEMs) or in dashboards with security events for further analysis and approval for the whitelists or blacklists.

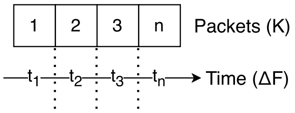


Fig. 5. Query analysis ratio.

4.3.2. Phase 2 - query rate analysis

The real address of the C2 server(s) is often masqueraded, as the infected device spams thousands of queries. Just as the attacker can exploit this for its own gain, the defenders can also use this information to defend the system.

Implementing a mechanism to detect high rates might seem easy at first, but one must take into consideration the scalability implications of the approach, as this pipeline would most benefit from being implemented at the core networks of network operators. Therefore, one possible approach would be to implement an event-driven architecture as a doubly linked-list-like data structure. When new packets are added to the structure, and at the other end, the packets are removed, so that the structure only contains a predefined window of time w. As mentioned, a doubly linked-list data structure is required, in order to efficiently add and remove elements.

In the implementation of the proof of concept, we followed an effective approach. Instead of keeping a list of all packets that circulate on the network for a given time window, we measure the time difference between every DNS query for each individual device, as depicted in Fig. 5. For example, if packet 1 arrived at the timestamp $t1 = 1$ and packet 2 arrived at $t2 = 2$ then they differ in 1 time unit. With this approach, we can use parameters, to calibrate the sensitivity of the algorithm:

- ΔF which details the maximum allowed time unit interval;
- K which specifies how many packets can disrespect ΔF before being quarantined.

ΔF is used to calibrate the interval sensitivity, where $\Delta F = 0$ is the shortest interval possible, and therefore the smallest possible number of packets are caught. K is used to limit packet capture even if ΔF fails. For example, in situations where a legitimate service makes 5 queries in the designated time ΔF, nothing will be reported if $K > 5$.

4.3.3. Phase 3 - DGA detection

Any packet that may not be detected by the query rate analyzer, or is simply a regular packet, will be parsed through a detector for queries issued by DGA approaches. In the implementation of the proof-of-concept, we used the Intel DGA solution, available, as open source on github (Mallarapu, 2020).

The outcomes of the evaluation performed in this phase include a value between 0 and 1 indicating whether a domain could be or not be DGA generated, where 0 is a regular domain and 1 is a DGA-Generated Domain. With this phase, it should be once again possible to calibrate the acceptance/rejection criteria from the DGA-generator. In experiments, we defined the lower bound as 0.1, which meant immediate acceptance of the packet, and the upper bound as 0.9, which meant immediate packet discard. All the other $0.1 \le x \le 0.9$ will be evaluated in the next step of the pipeline. These threshold values were based on the sensitivity analysis conducted when gathering knowledge of the FluBot malware, as documented in this work (Dias et al., 2023).

4.3.4. Phase 4 - machine learning detection

The last stage of the pipeline is the machine learning evaluation, where a minimum number of packets should arrive. They represent the hardest DNS requests to classify and, therefore, take the longest time in the pipeline. This stage produces a binary output, where 0 is not infected, and 1 is an infected package. The design of the model encompasses feature selection and model training, as detailed below. One

Table 1
Selected feature set.

Feature ID	Description	ML data type
IP Src	Source IP performing the request	Bit Conversion
IP Dst	Destination of DNS request	Bit Conversion
Length	Size of the Payload	Integer
DNS Flag	Info Regarding Flags	Boolean
DNS Questions	Requests in a DNS message	Integer
Query Type	Query Type (A, AAAA, CNAME, PTR)	Integer
Query Name Null	If DNS name is NULL or not	Boolean
Timestamp	When the packet was created	Integer

of the first steps in designing the ML model is related to the selection of features. This process was performed considering the preliminary analysis of FluBot, as documented in Section 3. These features are summarized in Table 1 and are mainly based on fields present in DNS requests.

It should be mentioned that some of the fields were converted to numeric values for efficiency concerns. The IP addresses used bit conversion for each decimal octet of an IP version 4 address. Note that the features can be applicable to types of malware other than FluBot, as long as they use DNS packets to identify and initiate communications with the C2 server. As summarised in Table 1, the chosen features for botnet detection are narrowed to FluBot. Instead, they are based on the fields of standard IPv4 packets and DNS protocol, such as the *DNS QueryType* and *DNS Questions*, which can be employed for other types of malware (Salahdine et al., 2023).

Each model follows a simple, yet effective, training strategy. The training data included a total of 10.000 data points, from which there is a 50/50 split between a fabricated packet set using the Alexa Top 1 million domain list (Target, 2020), and lab-generated malware samples. As such, a balanced data set is structured, with a 50/50 split between examples of infected and non-infected packets. Finally, the trained AI is subject to an 80/20 test/train split, where 80% of the data is used to train the ML model and the remaining 20% is used to test the accuracy of the AI. The model is implemented in Python using the scikit-learn library (Team, 2007), with the available *RandomForest* and *IsolationForest* classifiers, as these presented the best performance in terms of accuracy and speeds, compared to others such as KNN. Detailed results of the comparison between the different ML models are provided in this work (Dias et al., 2023).

4.4. C2 server detection

The detection of the Command and Control – C2 server – needs to consider the behaviour of the malware. Infected devices are detected with a high volume of DNS queries, where the goal is to mask communications with the C2 server.

In the proof of concept with Flubot malware instances, the behaviour of the infected device indicates that a HTTP handshake is performed between the C2 server and the device to establish a connection. This happens after DNS queries start, immediately stopping the DNS flood after a successful connection. If this handshake is captured, we can filter from the several thousands of possible domains (DGA generated) to the ones that are registered and currently active. The C2 server detection mainly involves the capture of the handshake in a efficient fashion, after the trigger of infected traffic has been detected.

4.4.1. Capturing the handshake

To successfully capture the handshake with the C2 server, a packet filter was first deployed into the network, capturing HTTP connections that match a common format. The endpoint(s) with possible C2 server(s) identified by the filter can correspond to the names employed in previous DNS requests. To achieve high accuracy, the uniform resource identifier (URI) in HTTP packets is analysed (Mallarapu, 2020). If the prediction is higher than a user-defined threshold, it is considered infected. Fig. 6 shows the request URI present in a filtered packet to the C2 server `tofelumrdyrgwhg.ru` and it can be easily identified as infected. This simple assertion greatly decreases the probability of a false positive. Other approaches could also consider the analysis of the HTTP reply status code, which with a value of 200 means a successful request.

4.4.2. Implementation of detection mechanisms in MONDEO

Considering a microservice architecture, and that packet capture occurs within other processes/approaches, for instance using the UPF in a 5G network, the mechanism works as follows:

1. Based on DNS analysis (Section 4.3), the system establishes a list of possibly infected devices (based on time, so that false positives are not triggered).
2. Whenever a packet arrives, it is analyzed with respect to the source, if it is present in the infected list - *phase 1* the observed query ratio (Section 4.3.1), and with respect to the name / URI requested in the DNS request (i.e., if DGA based or not). If it meets any of the criteria in the various phases, it is considered infected (Section 4.3.3).
3. Otherwise, the final evaluation is carried out considering:
   - Infected, if conditions in phase 3 are true;
   - Not infected, otherwise.
4. Perform C2 server detection.
5. On the basis of this evaluation, a JSON response is produced and provided through the REST API. This response is used by the MONDEO Tactics component.

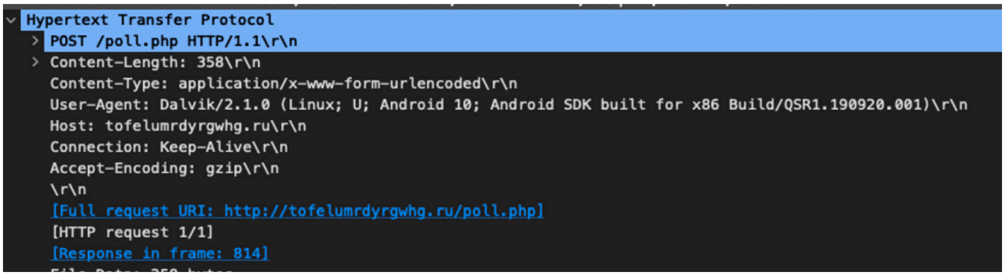
The implementation follows a microservice architecture, promoting efficiency by parallelising the detection of botnets and retrieving relevant information to apply tactics. This also facilitates the deployment in 5G/6G architectures, where network functions can be implemented as virtual network functions in virtual machines or as microservices. The identification of the C2 server is related with the FluBot malware patterns, other types of malware will have different patterns (drWeb, 2023). The detection mechanism in MONDEO has considered 6 distinct FluBot malware samples performing high volume of DNS requests towards C2 server identification, to enhance the detection accuracy, as detailed in Section 5.

The detection of the C2 server is communicated to the tactics component with an associated probability, to devise the appropriate tactics for risk mitigation, as detailed in the next section.

4.5. Tactics component

Once malware is detected and detection is communicated through the *M.R* interface in the MONDEO architecture (Fig. 3), we need to decide which tactics must be applied to mitigate associated risk. For this, the MONDEO Tactics component uses Rainbow (Cheng et al., 2006), a self-adaptation framework that can react to the information provided by MONDEO Detection. Briefly, self-adaptation introduces a control loop that comprises four main activities: **M**onitoring, **A**nalysis, **P**lanning, and **E**xecution, which use shared **K**nowledge about the state of the system and the environment in which it is running. Collectively, these activities are known as the MAPE-K loop (Arcaini et al., 2015). The monitoring activity updates the Knowledge with information about the system and the environment, represented in a model. The model is updated using data from the MONDEO detection component on current knowledge of infected devices and malicious C2 servers. Analysis evaluates these data to create higher-order knowledge. In this case, MONDEO Detection also provides information about probabilities associated with suspicious activities. Analysis within MONDEO tactics determines whether a response is needed.

MONDEO Tactics uses the information from the MONDEO detection component to maintain a model of the system and its environment. From this information, it can decide if the system is not behaving as



```
⌄ Hypertext Transfer Protocol
  > POST /poll.php HTTP/1.1\r\n
  > Content-Length: 358\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
    User-Agent: Dalvik/2.1.0 (Linux; U; Android 10; Android SDK built for x86 Build/QSR1.190920.001)\r\n
    Host: tofelumrdyrgwhg.ru\r\n
    Connection: Keep-Alive\r\n
    Accept-Encoding: gzip\r\n
    \r\n
    [Full request URI: http://tofelumrdyrgwhg.ru/poll.php]
    [HTTP request 1/1]
    [Response in frame: 814]
```

Fig. 6. Wireshark detail on possibly infected packet.

desired and develop or choose a plan to fix/mitigate identified problems. For this purpose, MONDEO Tactics uses utility theory (Cheng et al., 2006). A planner within MONDEO Tactics uses this information and predicts the effects of various tactics on the future utility of the system. The tactic(s) that maximize the utility are the ones that Rainbow then selects and executes. These tactics are discussed below.

The interface between MONDEO Detection and MONDEO Tactics components occurs through the *M.R* interface, as shown in Fig. 3. The data that is exchanged in a JSON format includes:

- Timestamp of the identification event;
- Identification of the infected device (IPv4 address);
- Identification of the C2 server (IPv4 address and FQDN);
- Level of certainty in the detection.

The integration between MONDEO Detection and MONDEO Tactics components is via *probes*. MONDEO Detection acts as a probe (monitor) in MONDEO Tactics, periodically reporting information that is used to update the model. MONDEO Tactics with Rainbow, when receiving information of infected devices and the C2 server identification, configures the required policies to mitigate the attack. We consider three representative tactics to respond to malware:

T1 quarantining the mobile device;
T2 "blackholing" the C2 server (i.e., resolving its DNS to a benign IP address, like a honeypot for further analysis);
T3 requiring the mobile device user to complete a CAPTCHA upon the detection of a suspicious connection.

Quarantine approaches are per mobile device basis, where each mobile device is quarantined separately. Tactic **T1** does not scale well and is prone to false positives by isolating benign devices. Blackholing the IP address of the C2 server is something that is done in a centralized way by a mobile network operator, and it simultaneously affects all mobile devices that use the mobile network operator's DNS servers. Thus, being more efficient in terms of scale, as well as policy enforcement (**T2**). Note that the introduction of DNS security measures like DNSSEC can make it more difficult to effectively blackhole IP addresses, but we do not address those difficulties here. While blackholes are network-level, and quarantines are device-level, CAPTCHAs are connection-level tactics (**T3**). They are used to temporarily block a connection until a human user is able to prove that they are the one making the connection request rather than malware or a bot.

At the same time, we model three impacts that affect overall utility: (I1) There is an impact on the utilization of the customer service department for the mobile network operator; (I2) intrusiveness to the customer experience; and (I3) effectiveness in containing the attack.

Details of tactic implementation can affect impact on utility. For example, if quarantine uses a list of known bad sites to prevent malicious connections (**T1**), the impact on the end user will be minimized (I2), while effectiveness will also be diminished (I3) compared to a quarantine approach that uses a list of known good sites. The known-good-sites approach is more limiting and will result in higher impacts to the mobile user and mobile network operator while also likely being more effective.

4.6. Tactics for 5G networks and beyond

The tactics T1, T2 and T3 can be implemented differently in a 5G or beyond network. The most straightforward tactic to implement is the blackhole (**T2**). In this case, an update is sent to the DNS server to either not resolve an address or resolves it to a benign address like a honeypot. This can be done with the collaboration of 5G functions such as UPF or PCF, as shown in Fig. 3.

Quarantine of a device (**T1**) can be implemented using technologies, such as Network Function Virtualization (NFV) to create a network slice for quarantined devices. These devices can have a separate DNS service or have their connections mediated through a user plane function (UPF) that limits to known-good (or not known-bad) connections depending on the specific implementation.

The CAPTCHA tactic (**T3**) is the most complex to implement. It requires the implementation of devices on the 5G core network that can detect the network flows to be temporarily blocked, serve up the CAPTCHA, and – if the CAPTCHA is correctly completed – maintain state to open the flow and allow the flow to reopen with subsequent reconnections. UPF and Session Management Function (SMF) are the feasible 5G core functions to implement this connection tracking and management of the CAPTCHA mechanism. Nevertheless, this approach can also have associated scalability issues that can be associated with the high number of simultaneous connected devices and with their heterogeneous characteristics (e.g., without support for HTTP protocol).

From the perspective of a network operator, botnet detection must be integrated with the 5G core functions that are responsible for managing mobile device traffic in the data plane. This is associated with the requirement R2 of the use case, described in Section 4.1. UPF is a feasible candidate, mainly to implement packet capture or deep packet inspection functionalities to detect botnets, as described in Section 4.3. Such a traffic analysis is required not only for the detection of botnets, but is also crucial to identify the C2 servers. Traffic inspection can consider 5G mechanisms that can be in place such as network slicing, where a UPF function may exist in each slice. Additionally, considering only the detection of botnet traffic, only the packet capture of DNS requests is required. Other types of malware may require full packet analysis. Botnet detection can be facilitated if operators manage the DNS server(s).

5. MONDEO-Tactics5G evaluation

This section describes the evaluation methodology.

5.1. Datasets

The overall scenario for data collection is represented in Fig. 7.

To be as realistic as possible, we configured a DNS server using ISC BIND, where we collected information regarding DNS queries. The collection included DNS logs and the capture of DNS packets with the

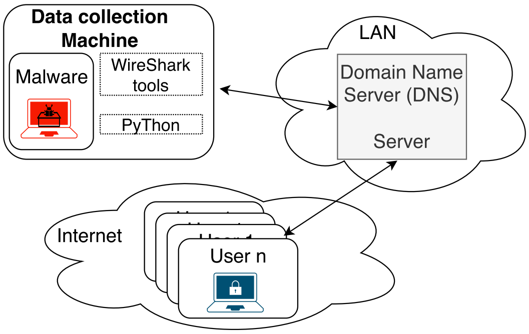


Fig. 7. Data collection setup.

Table 2
FluBot Malware sample information.

Name	Malware	File(s)	Description
Correos	FluBot	Correos1	Application similar to Correos app
FedEx	FluBot	FedEx1, FedEx2	Application similar to FedEx app
UPS	FluBot	UPS1	Application that mimics UPS app
DHL	FluBot	DHL1, DHL2	Application similar to DHL app
VoiceMail	FluBot	VoiceMail1	Application similar to VoiceMail app

Table 3
Tests information in the evaluation of data pipeline.

Test	Type	File(s)	Description
#1	Infected	FedEx1, FedEx2, UPS1, Correos1, DHL1, DHL2, VoiceMail Lab	With malware samples
#2	Benign	23	Only with regular DNS requests

tshark tool. The information in the datasets included the DNS packets that were captured from regular DNS clients of several volunteer participants that configured their devices to use the configured DNS server.

To generate infected traffic, we set the data collection machine as an isolated machine, where we could safely deploy hidden malware applications (APK), as summarized in Table 2. These malware samples were obtained from online repositories such as MalwareBazar (Abuse.ch, 2022) and Koodous (Koodous, 2022). The profile for the emulated/virtual device was based on the Pixel 4 running Android API 29. This device was chosen for its computing characteristics, which we found to be representative of the mobile phones used worldwide. In addition, this device is also used in related work to evaluate the behaviour of mobile devices (Rybakov et al., 2020; Trotta et al., 2021). The emulated mobile devices were activated in different time periods and thus the malware traffic - infected - is recorded in specific tshark capture files. It should be noted that while malign requests were being made, Benign traffic associated with regular DNS requests was also running.

5.2. Botnet detection

The evaluation of MONDEO-Tactics5G includes the performance characterization of each phase in the MONDEO data pipeline, as documented in Section 4.3. Performance is assessed in terms of the time required to process a packet in each phase and the overall number of packets that are processed.

All tests are based on the DNS samples collected in the DNS Experimental Setup, summarized in Table 3 and collected using the methodology documented in Section 5.1. In this evaluation we focus on testing the impact of the malware, especially with regards to the HTTP handshake. From this list, a data set with the features summarized in Table 1 was built to assess the performance of MONDEO and the application of tactics. The metrics used to assess the performance of the MONDEO Data Pipeline are summarized in Table 4.

Table 4
Performance metrics in the MONDEO data pipeline.

Metric	Unit	Description
Packets Processed	%	Ratio of packets processed in each phase, considering the total of captured packets
Processing Time	ms	Time required to process a packet in each phase
Classification	n/a	Final classification of MONDEO, if packet is flagged as infected or benign.

5.3. C2 server detection

To evaluate the performance of the C2 detection, we consider the ratio of analyzed HTTP requests, in terms of passed or flagged as infected (recall Fig. 4). Resource consumption is also considered, in terms of CPU usage, memory usage ratios, and the amount of information that is exchanged and in bytes.

5.4. Tactics

To evaluate the effectiveness of **T1**, **T2** and **T3** tactics, we model the system and then use statistical model checking to show which combinations of tactics would be most effective in each state. We consider the diverse impact (I1, I2, I3), previously identified in Section 4.5.

We took this approach because we did not have access to a real 5G network or a simulator of it. In this evaluation approach, however, we show that in all cases, tactics would improve the overall utility of the network, considering the dimensions of utilization of customer service (I1), the intrusiveness for customer service (I2) and effectiveness in containing the attack (I3).

5.4.1. Model

We model our system as a Discrete-time Markov Chain (DTMC) using the modelling language of the probabilistic model checker PRISM (Kwiatkowska et al., 2011). Given the complexity and scalability issues that would entail representing devices and servers individually in our model, we group and represent them as device and server *tiers*, as pictured in Fig. 8.

Devices and servers within the same tier are considered to have a similar likelihood of being compromised by malware, and hence the level of granularity at which tactics are applied is tiers, and not individual devices/servers. For the sake of clarity, rather than introducing the PRISM code, we describe in Fig. 8 the dynamics of interaction among processes (*modules* in the PRISM nomenclature) that capture the behaviour of the main components in our system (i.e., device and server tiers, connectors, and customer service). Device tiers contain a state variable compromised, which ranges between 0 and 100 and captures the likelihood that the device tier is compromised by malware. Server tiers contain a similar state variable c2, which captures the probability that a server level is compromised by malware. The customer service module contains a variable utilization, which ranges between 0% and 100% to capture the level of utilization of the customer service (i.e., whether agents are busy answering calls from clients experiencing service disruption) *I1*.

Each device tier is assigned a maximum number of attempts - MAX_REQUEST_ATTEMPTS to communicate with servers. For each attempt, if the device is not quarantined - **T1**, it is allowed to send a request to the server through the connector. Once the request is received at the connector, the request is automatically forwarded to the corresponding server tier if the CAPTCHA for the device tier sending the request is not enabled. If CAPTCHA tactic is enabled **T3**, the request is forwarded only with a given probability pCaptcha, which is inversely proportional to the value of compromised at the device level that sent the request (i.e., the higher the likelihood of being compromised by malware, the lower the probability of passing CAPTCHA). After receiving the response from the server, the connector checks if the server tier

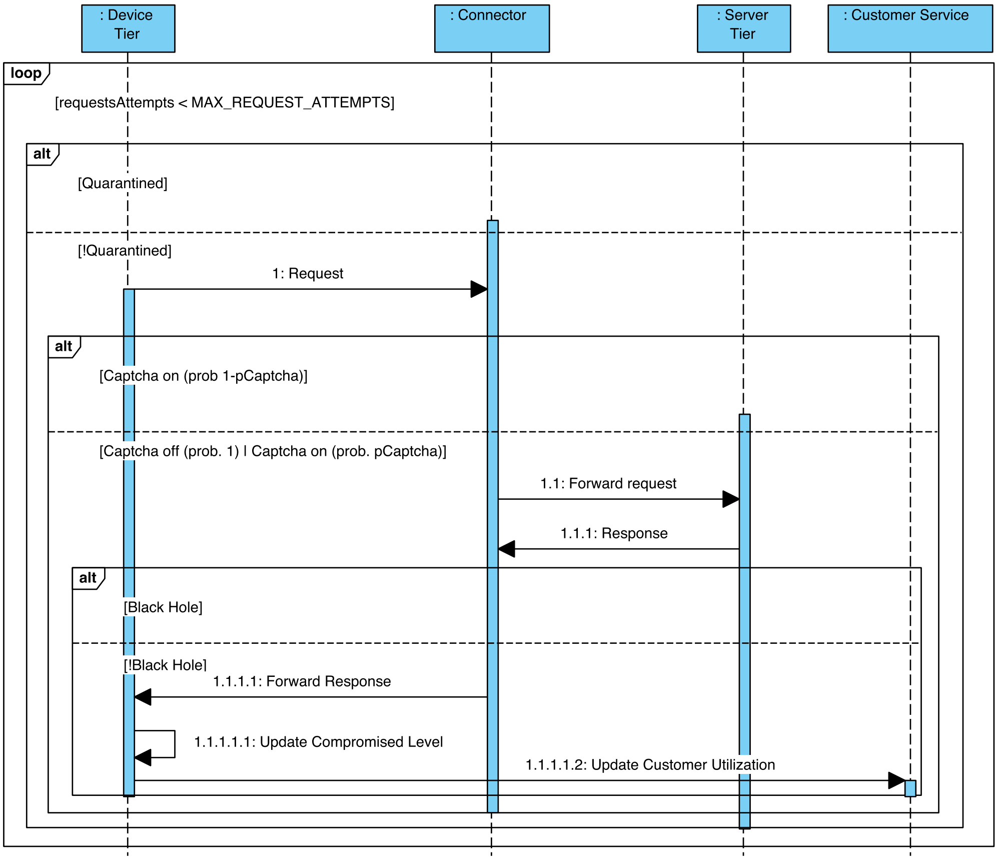

: Device Tier
: Connector
: Server Tier
: Customer Service
loop
[requestsAttempts < MAX_REQUEST_ATTEMPTS]
alt
[Quarantined]
[!Quarantined]
1: Request
alt
[Captcha on (prob 1-pCaptcha)]
[Captcha off (prob. 1) I Captcha on (prob. pCaptcha)]
1.1: Forward request
1.1.1: Response
alt
[Black Hole]
[!Black Hole]
1.1.1.1: Forward Response
1.1.1.1.1: Update Compromised Level
1.1.1.1.2: Update Customer Utilization

Fig. 8. Sequence diagram describing interactions of the PRISM model.

Table 5
Terminology.

Term	Description
α	Malware detection accuracy
T	Set of available tactics, $T1-quarantine, T2-blacklist, T3-captcha$
m_t	Multiplicative factor for the disruption of different tactics
$e_t i$	Indicates if t is enabled in the device tier i
$u_u(*)$	Utilization utility function
$u_e(*)$	Effectiveness utility function
u_{mau}	Mean accrued utility
R{r}	Reward Operator
ϕ	Reachability predicate

is blacklisted and, if that is not the case, it forwards the response back to the device tier. Once the response is received on the device tier, the following updates take place:

- The value of compromised is updated by adding to it the c2 of the server that sent the response.
- The value of utilization is updated in customer service to reflect the new levels of customer service utilization. Updates are:
  - Directly proportional to compromised and c2 levels.
  - Inversely proportional to the accuracy of the malware detection parameter designated by α, as reported by the MONDEO detection component.

The magnitude of the updated value depends on the tactics that are activated. In practice, blacklisting, captcha and quarantining have multiplicative factors that capture the increasing level of disruption in services that these tactics can introduce (with quarantining being the most disruptive), as per Eq. (1):

$$utilization = \sum_{t \in T}(1-\alpha)m_t * \sum_{j=1}^{n} compromised_j * e_t i \tag{1}$$

In Eq. (1), $T = \{T1-quarantine, T2-blacklist, T3-captcha\}$ is the set of available tactics, m_t is a multiplicative factor that models the disruption of different tactics, n is the number of device tiers, and $e_t i$ is 1 if tactic t is enabled in device tier i, and 0 otherwise. The terminology is summarized in Table 5.

To measure the value provided by the system during execution, we consider a utility function $U : \mathbb{R} \times \mathbb{R} \to [0,1]$ defined as a linear combination of two terms that correspond to the level of utilization of the system and the effectiveness of the adaptation tactics (i.e., in terms of minimizing the likelihood of devices being compromised):

$$U(u,e) = w_u * u_u(u) + w_e * u_e(e) \tag{2}$$

In Eq. (2), the utilization utility function $u_u(*)$ returns an output between 0 and 1 that is inversely proportional to the utilization value provided as input, while the effectiveness utility function $u_e(*)$ takes as input an effectiveness value that corresponds to the mean of the compromised values across all device levels and returns a value between 0 and 1 that is inversely proportional to it.

Our model incorporates a reward structure that enables the storage of information about the accrued utility. During execution, an amount of utility equivalent to the result of Equation (2) is accrued in the reward structure at the end of every cycle of the loop depicted in Fig. 8. We designate the amount of utility accrued during the execution of a scenario as u_{mau} (mean accrued utility).

5.4.2. *Analysis*

To evaluate the effectiveness of the tactics, we take an approach similar to the evaluation in Cámara et al. (2016), where we analyze the system using the statistical model checking engine of the PRISM

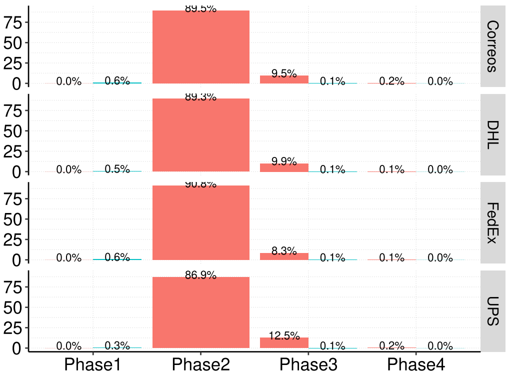


Fig. 9. Detection rate.

probabilistic model checker. We model check the system to understand the effect of various strategies on the utility of the system, comparing versions of the system with and without adaptation.

To quantify the utility that each strategy yields, we make use of Probabilistic Reward CTL (sPCTL) (Andova et al., 2004), which extends the probabilistic temporal logic PCTL (Bianco and de Alfaro, 1995) with reward-specific operators aimed at the specification of performability measures over DTMC models. Specifically, our technique enables us to statically analyze a particular region of the state space, which first has to be discretized to check the PRCTL properties. Obtaining the results of the analysis for each state in the discrete set requires an independent run of the model checker, in which the model parameters are instantiated with variable values that correspond to that state. In our case, the discrete set we consider corresponds to pairs (α, w_u) in the range $[0, 1]$, where the discretization step for accuracy is $\mu_\alpha = 0.1$ and the one for w_u is $\mu_{w_u} = 0.05$. These values are defined according to the evaluation performed in Cámara et al. (2016).

For each independent run of the model checker, we analyze a PRCTL property that employs the reward operator $\mathsf{R}\{\mathsf{r}\}_{=?}[\mathsf{F}\phi]$, which enables the quantification of the accrued reward r along paths in a model that eventually reach states that satisfy the reachability predicate ϕ. Concretely, we analyze the property $\mathsf{R}\{\mathsf{utility}\}_{\mathsf{max}=?}[\mathsf{F}\ \mathsf{done}]$, where utility is the name of the reward structure in our PRISM model, and done is a label that corresponds to an expression over state variables that captures states in which devices can no longer perform requests (because MAX_REQUEST_ATTEMPTS has been reached).

6. Evaluation results

This section summarizes the evaluation results.

6.1. Botnet detection

MONDEO-Tactics5G uses a multistage feedback loop to detect FluBot malware packets.

As illustrated in Fig. 9 most of the detection, for the samples used, is performed in phase 2, which assesses the query rate. In this phase, the majority of requests is flagged as malware due to the high number of requests per second. In the evaluation results, the feedback loop was not used, as one can see in Fig. 10 since there is no flagged time in phase1 and the passed is almost zero.

The phases with more impact, in terms of processing time, are phases 3 and 4, which use DGA algorithms and the ML models (i.e., *RandomForest* and *IsolationForest*), respectively. In these phases, the processing time is on the order of 400 ms, either to flag or to allow a packet to pass.

6.2. C2 server detection

The C2 server detection is assessed in terms of the detection accuracy regarding the HTTP requests towards the C2 server.

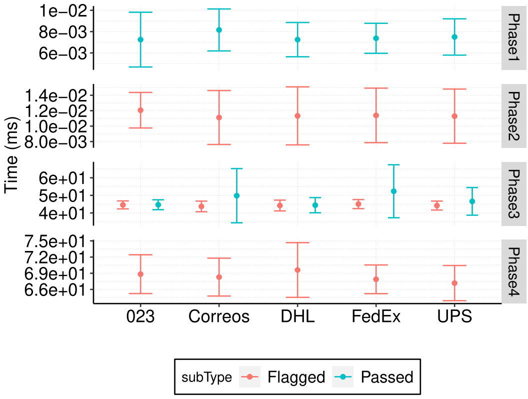


Fig. 10. Detection time.

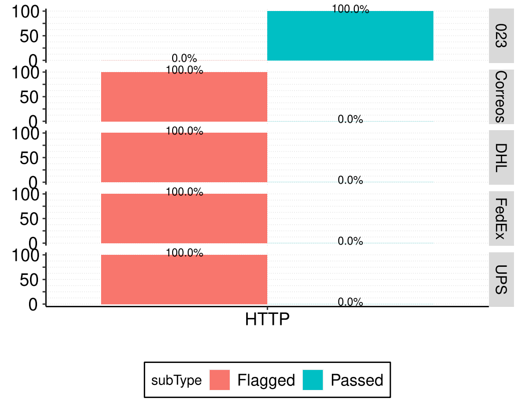
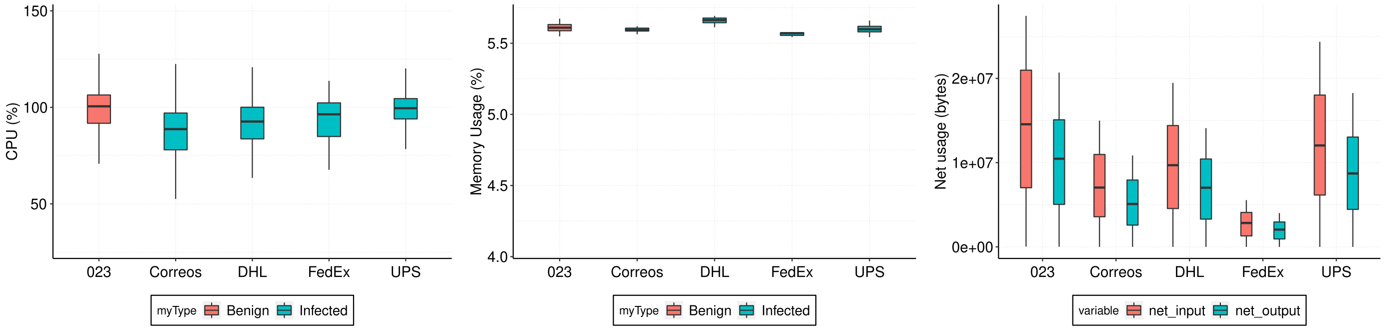


Fig. 11. Successful identification of HTTP requests to the C2 server(s).

Fig. 11 depicts the results of HTTP identifying the HTTP requests towards the C2 server. In the *023* dataset, all requests correspond to legitimate HTTP requests, while in FluBot-infected datasets there are HTTP requests to the C2 server(s) and benign HTTP requests, which correspond to browser requests. Most of the HTTP requests in the datasets with malign samples are for the C2 server(s). Detection of botnets and detection of C2 servers lead to resource consumption, as illustrated in Fig. 12, in terms of CPU, memory, and network I/O. The results report the resources consumed by the MONDEO detection component, which is implemented as a microservice with APIs for botnet detection and HTTP analysis. The MONDEO detection component has an impact in terms of CPU usage due to the required analysis; nonetheless, the impact on memory usage is low. In addition, the microservice exchanges small amounts of data, where the input is higher, since it contains information about the DNS and HTTP packets that are received. The output is provided in JSON format with a significantly lower volume. The amount of information exchanged under normal conditions - case *023* is greater, since DNS packets contain legitimate requests.

6.3. Tactics

Fig. 13 displays the experimental results for a scenario in which we compare the overall utility that the system can accrue during execution with and without adaptation. The system contains four device tiers and four server tiers, arranged in increasing order of likelihood of being compromised by malware (e.g., the first tier of devices is between 0 and 25%, the second one ranges between 25 and 50%, and so on). The horizontal axes of the graphs range between 0 and 1 and represent the precision of the system in detecting the likelihood that a device will be compromised (α, equal across all levels of the device in this scenario) and the weight given to the term of utilization of the utility function



(a) CPU usage ratio

(b) Memory usage ratio

(c) Network I/O

Fig. 12. Resource usage ratio.

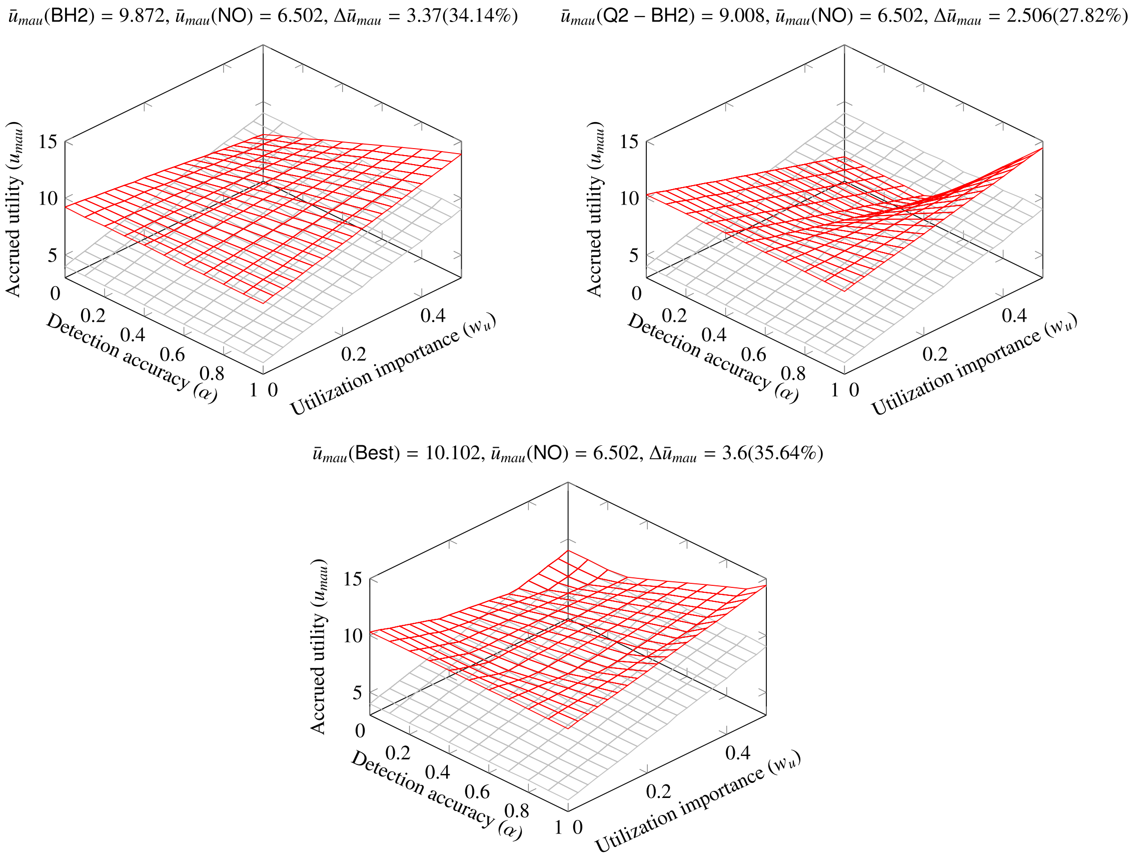


Fig. 13. Experimental results comparing accrued utility with and without adaptation strategies: Blacklisting (top, left), blacklisting combined with quarantining (top, right), and best strategy (bottom).

w_u. The vertical axis corresponds to the accrued utility u_{mau} for a given combination of values of α and w_u.

The grids in each plot show the performance of the system without adaptation (gray) and with adaptation (red). Plots in the Figure correspond to strategies that combine tactics from the following set:

- Q2 Quarantines (T1) the two top tiers of devices, i.e., those with a likelihood of being compromised by malware ranging between 50 and 100%.
- C2 Captchas (T3) the two top tiers of devices.
- BH2 Blacklists (T2) the two top tiers of servers.
- Q1 Quarantines (T1) the top tier of devices (75-100%).
- C1 Captchas (T3) the second highest device tier (50-75%).

Each strategy is labelled by the combination of tactics it uses. In the figure, NO refers to a strategy that does nothing (that is, it does not execute any tactics), and Best corresponds to a strategy that picks at each point (α, w_u) the best of all available strategies (including NO). The set of strategies analyzed corresponds to C2-BH2, C2, Q2, BH2, Q2-BH2, Q1-C1, Q1-C1-BH2. However, we only represent in the Figure strategies that are optimal at some point of the space (Q2-BH2 and BH2).

All adaptation strategies (even those not illustrated in the figure) outperform on average the version without adaptation, with a delta in the mean accrued utility ($\Delta\bar{u}_{mau}$) that is always positive and ranges between 1.62% (Q1-C1) and 34.14% (BH2), and goes all the way up to 35.64% for Best.

If we focus on the version without adaptation (gray grid, which is the same across all plots) and low values of utilization importance, we

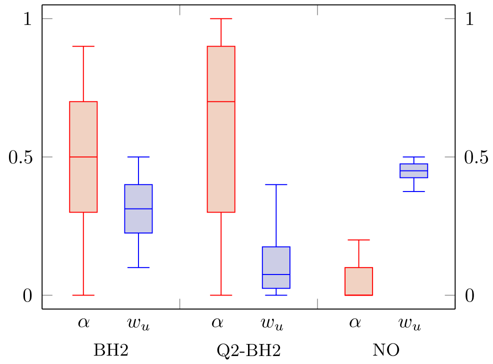


Fig. 14. Use context of best adaptation strategies: accuracy and utilization importance conditions.

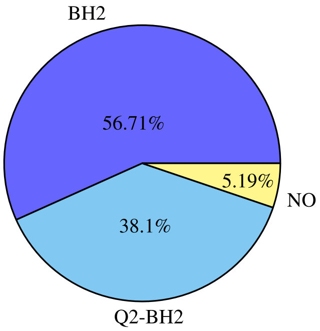


Fig. 15. Use context of best adaptation strategies: coverage.

can observe that there are lower values of accrued utility, compared to areas with higher values of utilization importance. This is to be expected because in the latter all utility is derived from the effectiveness term of the utility function, and highly compromised devices yield little utility on the effectiveness term of the utility function.

If we focus on adaptation versions (red grids), we can see that the general trend across all plots is that higher detection accuracy values tend to yield higher accrued utility values when we combine them with high values of w_u. This is because a higher detection accuracy results in less disruption to legitimate clients, and therefore, customer service utilization is less affected (I1) than it would be with lower accuracy, and this yields more utility coming from the utilization term of the function. It is also worth observing that high values of utilization importance with low detection accuracy tend to yield low utility because it creates situations in which more legitimate clients are disrupted (I2) and customer service gets overloaded. In fact, focusing on Best (bottom plot) reveals that the best strategy for very high utilization importance values ($w_u \simeq 0.5$) and very low α values is not to execute adaptation tactics at all to avoid disrupting customers.

Fig. 14 illustrates the context in which it is best to use different adaptation strategies. The figure shows a boxplot chart that illustrates the average, maximum, and minimum values of accuracy and utilization importance under which a given strategy is better than the rest. We can observe that the strategies BH2 and Q2-BH2 are best across a broad range of accuracy and importance of utilization. However, for very low utilization importance values, Q2-BH2 performs better than Q2. This is aligned with the fact that when all utility comes from the effectiveness term of the utility function, incorporating the quarantining tactic in addition to blacklisting servers is more effective at keeping the likelihood of devices being compromised at low values. Moreover, the plot also shows that not using any tactics is the best option for high values of utilization importance and low values of accuracy. This is consistent with our discussion of Fig. 13.

Fig. 15 displays a pie chart that corresponds to the coverage of the best strategies in the region of the space studied. The plot shows how in $\simeq 5\%$ of the situations, not executing any tactics is actually the best choice (this corresponds to the region of the state space of high utilization and low accuracy described previously). Moreover, we can see that the strategies that employ BH2 are the best among the set considered for our analysis, covering more than 94% of the space.

7. Conclusions

Mobile Malware is malicious code specifically designed to infect mobile devices with the goal of performing multiple types of fraud and is increasing rapidly. Efficient mechanisms are required to detect and mitigate their impact, in particular, in 5G and beyond networks. MONDEO-Tactics5G operates as a multistage botnet detection approach, allowing a system to automatically identify FluBot malware in different phases. The feedback loop supports the configuration of the blacklists as soon as samples of malware are identified.

MONDEO-Tactics5G also involves the application of tactics to respond to the malware detection event. Tactics provide mitigation to reduce the impact of Botnet attacks. In this paper, we discussed three tactics and a multidimensional utility space that allows us to balance the concerns of degree with which a 5G operator can adapt to attacks with various costs to the business, including overloading customer service departments. While the paper concentrates on these particular tactics and utilities, the approach is by no means limited to just these concerns, but can be tailored to different contexts and different tactics. Evaluation of the approach showed that our mitigation responses are effective. Even though for scalability purposes, we needed to consider buckets of devices and C2 servers rather than individual ones, it would guide companies to choose which tactics would be most beneficial in which contexts.

MONDEO-Tactics5G focused on the FluBot threat and established a foundational knowledge for detection of malware relying on the DNS protocol, and also established a set of tactics that mitigate such a type of thread efficiently. A key advantage of MONDEO-Tactics5G is the possibility of being integrated in 5G networks, through the interaction with specific 5G network functions, like the User Plane Function for traffic capture or in the Session Management Function for the application of specific controls.

MONDEO-Tactics5G relies mainly on DNS and HTTP packets to detect attacks and the C2 servers, respectively. Malware uses HTTPS or DNS over HTTPS to further make it more difficult to detect C2 servers. Our next steps will consider other types of malware that use DNS over HTTPS and other types of tactics fully integrated with 5G networks and exploiting the potential of the feedback loop.

CRediT authorship contribution statement

Bruno Sousa: Investigation, Supervision, Writing – original draft, Writing – review & editing. **Duarte Dias:** Investigation, Software, Writing – original draft. **Nuno Antunes:** Investigation, Writing – original draft. **Javier Cámara:** Investigation, Writing – original draft. **Ryan Wagner:** Investigation. **Bradley Schmerl:** Investigation, Writing – original draft. **David Garlan:** Investigation, Methodology, Supervision. **Pedro Fidalgo:** Investigation, Supervision, Writing – original draft.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Bruno Sousa reports financial support was provided by Foundation for Science and Technology. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

Acknowledgements

This work is funded by project AIDA (POCI-01-0247-FEDER045907), cofinanced by the European Regional Development Fund (ERDF) through the Operational Program for Competitiveness and Internationalisation (COMPETE 2020) and by the Portuguese Foundation for Science and Technology (FCT) under CMU Portugal. This work has been supported by Project “Agenda Mobilizadora Sines Nexus”. ref. No. 7113), supported by the Recovery and Resilience Plan (PRR) and by the European Funds Next Generation EU, following Notice No. 02/C05-i01/2022, Component 5 - Capitalization and Business Innovation - Mobilizing Agendas for Business Innovation. This work is funded by the FCT - Foundation for Science and Technology, I.P./MCTES through national funds (PIDDAC), within the scope of CISUC R&D Unit - UIDB/00326/2020 or project code UIDP/00326/2020. This work was also partially funded by the Spanish Government (FEDER/Ministerio de Ciencia e Innovación–Agencia Estatal de Investigación) under projects TED2021-130523B-I00 and PID2021-125527NB-I00.

References

Abuse.ch, 2022. Malwarebazaar database. bazaar.abuse.ch/browse/tag/flubot/, Last visit: 2024-01-28.

Ahmed, I., Anisetti, M., Ahmad, A., Jeon, G., 2023. A multilayer deep learning approach for malware classification in 5g-enabled iiot. IEEE Trans. Ind. Inform. 19 (2), 1495–1503.

Almutairi, S., Mahfoudh, S., Almutairi, S., Alowibdi, J.S., 2020. Hybrid botnet detection based on host and network analysis. J. Comput. Netw. Commun., 2020.

Andova, S., Hermanns, H., Katoen, J.-P., 2004. Discrete-time rewards model-checked. In: Formal Modeling and Analysis of Timed Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 88–104.

Arcaini, P., Riccobene, E., Scandurra, P., 2015. Modeling and analyzing mape-k feedback loops for self-adaptation. In: 2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pp. 13–23.

Association, C.F.C., 2021. Cfca 2021 fraud loss survey. techreport.

Bianco, A., de Alfaro, L. Model checking of probabalistic and nondeterministic systems. In: Foundations of Software Technology and Theoretical Computer Science. 15th Conference Bangalore, India, December 18-20, 1995, Proceedings.

Cámara, J., Garlan, D., Moreno, G.A., Schmerl, B., 2016. Evaluating Trade-Offs of Human Involvement in Self-Adaptive Systems. Elsevier.

Cheng, S.-W., Garlan, D., Schmerl, B., 2006. Architecture-based self-adaptation in the presence of multiple objectives. In: ICSE 2006 Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). Shanghai, China.

Dias, D., Sousa, B., Antunes, N., 2023. Mondeo: multistage botnet detection. arXiv:2308.16570.

Elie Bursztein, D.O., 2019. Deconstructing the phishing campaigns that target gmail users. fastcompany.com/90387855/we-keep-falling-for-phishing-emails-and-google-just-revealed-why, Last visit: 2024-01-28 (Aug. 7).

Flinders, K., 2021. Uk consumer trust in banks, retailers and telcos declines as scams increase. computerweekly.com/news/252507268/UK-consumer-trust-in-banks-retailers-and-telcos-declines-as-scams-increase, Last visit: 2024-01-28 (Sep. 28).

Kaspersky, 2023. Kaspersky security bulletin 2023. securelist.com/ksb-2023-statistics/111156/, Last visit: 2024-01-28.

Koodous, 2022. Collective intelligence against Android malware. koodous.com/, Last visit: 2024-01-28.

Kwiatkowska, M., Norman, G., Parker, D., 2011. PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (Eds.), Proc. 23rd International Conference on Computer Aided Verification (CAV’11). In: LNCS, vol. 6806. Springer, pp. 585–591.

Lyu, M., Gharakheili, H.H., Sivaraman, V., 2022. A survey on dns encryption: current development, malware misuse, and inference techniques. ACM Comput. Surv. 55 (8).

Mallarapu, R., 2020. github.com/sudo-rushil/dgaintel, Last visit: 2024-01-28.

October 2023 review of virus activity on mobile devices. news.drweb.com/show/review/?lng = en&i = 14775, Last visit: 2024-01-28.

paloalto Networks, 2022. Stop attackers from using dns against you. techreport.

Principi, L., Baldi, M., Cucchiarelli, A., Spalazzi, L., 2023. Efficiency of malware detection based on dns packet analysis over real network traffic. In: 2023 IEEE International Conference on Cyber Security and Resilience (CSR), pp. 42–47.

Rybakov, O., Kononenko, N., Subrahmanya, N., Visontai, M., Laurenzo, S., 2020. Streaming keyword spotting on mobile devices. In: Interspeech 2020, ISCA.

Salahdine, F., Han, T., Zhang, N., 2023. Security in 5g and beyond recent advances and future challenges. Secur. Priv. 6 (1), e271.

Singh, M., Singh, M., Kaur, S., 2019. Issues and challenges in DNS based botnet detection: a survey. Comput. Secur. 86, 28–52.

Target, H., 2020. https://hackertarget.com/top-million-site-list-download/, Last visit 2021-09-08.

Team, S., 2007. scikit-learn.org/stable/, Last visit: 2024-01-28.

Tian, W., Lin, K., 2022. Chapter 2 - requirements and scenarios of 5g system. In: Shen, J., Du, Z., Zhang, Z., Yang, N., Tang, H. (Eds.), 5G NR and Enhancements. Elsevier, pp. 41–52.

Trotta, S., Weber, D., Jungmaier, R.W., Baheti, A., Lien, J., Noppeney, D., Tabesh, M., Rumpler, C., Aichner, M., Albel, S., Bal, J.S., Poupyrev, I., 2021. 2.3 soli: a tiny device for a new human machine interface. In: 2021 IEEE International Solid- State Circuits Conference (ISSCC), vol. 64, pp. 42–44.

Wang, K., Huang, C.-Y., Tsai, L.-Y., Lin, Y.-D., 2014. Behavior-based botnet detection in parallel. Secur. Commun. Netw. 7 (11), 1849–1859.

Wang, W., Shang, Y., He, Y., Li, Y., Liu, J., 2020. BotMark: automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviours. Inf. Sci. 511, 284–296.

Wazzan, M., Algazzawi, D., Bamasaq, O., Albeshri, A., Cheng, L., 2021. Internet of things botnet detection approaches: analysis and recommendations for future research. Appl. Sci. (Switzerland) 11 (12).

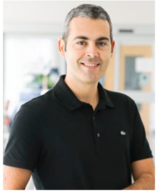
Xing, Y., Shu, H., Zhao, H., Li, D., Guo, L., 2021. Survey on botnet detection techniques: classification, methods, and evaluation. Math. Probl. Eng. 2021, 6640499.

Zhan, M., Li, Y., Yu, G., Li, B., Wang, W., 2022. Detecting dns over https based data exfiltration. Comput. Netw. 209, 108919.

Bruno Sousa is an Assistant Professor in the Department of Informatics Engineering of the University of Coimbra, Portugal, since December 2018, from where he got a PhD in Informatics Engineering on the subject of Multihoming for IP-based networks, in December 2014. He is a senior researcher in the Centre for Informatics and Systems of UC (CISUC), where he initiated his activities in 2006. He is the author of several book chapters, several publications in journals and international conferences. He has participated in the TPC of several conferences. He has participated in several European and national research projects, such as IST FP6 Integrated Projects, EuQoS and WEIRD, ICT FP7, MobiTRust, SALUS, Mobile Cloud Networking, LiveCity and FI-WARE, and H2020 EMPATIA. His research interests include resilience mechanisms in networks and applications/services, and intrusion detection and prevention in 5G networks and for Internet of Things (IoT).

Duarte Dias is a former MSc Student in cybersecurity at the University Coimbra, where he got his degree in 2023. His research interests include ethical hacking, penetration testing and network security.

Nuno Antunes is an Assistant Professor at the Department of Informatics Engineering of the University of Coimbra. Since 2008 he has been actively researching on secure and dependable software with the Software and Systems Engineering group of CISUC. His research interests include testing, fault injection, vulnerability injection and benchmarking, which are applied in the assessment of the dependability and security of cloud applications, intelligent systems, virtualized environments, web services, web and mobile applications, and data management systems (see H2020 ATMOSPHERE, EUBra-BIGSEA, FP7 CECRIS). He has been working with helping software engineers to create secure systems by proposing automated tools to find vulnerabilities and detect intrusions (see CMU|PT AIDA), and also with benchmarking approaches that allow selecting the most adequate alternatives (see SPEC RG Security).

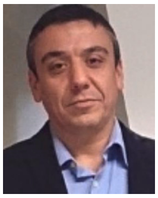


Javier Cámara is an Associate Professor at the Department of Computer Science, and a researcher at the Institute for Software Technology and Software Engineering (ITIS) in the University of Málaga. He is also an Honorary Visiting Fellow in the Department of Computer Science, University of York, where he was a Lecturer during 2018-2021. Prior to that, he was a Senior Systems Scientist at the Institute for Software Research, Carnegie Mellon University. He received his Ph.D. in Computer Science from the University of Málaga in Spain and has held postdoctoral appointments at INRIA Rhône-Alpes and the Centre for Informatics and Systems of the University of Coimbra. He works at the intersection of software engineering, formal methods, and artificial intelligence. Among others, his research interests include: software engineering for autonomous and self-adaptive systems, software architecture, applied formal methods, and cyber-physical systems.

Ryan Wagner is a Ph.D student at the Carnegie Mellon University.

Bradley Schmerl is a principal systems scientist in the Institute for Software Research at Carnegie Mellon University, Pittsburgh. His research interests include software architecture, self-adaptive systems, and software engineering tools. Schmerl received a Ph.D. in computer science from Flinders University, Adelaide, Australia.

David Garlan is a Professor of Computer Science in the School of Computer Science at Carnegie Mellon University. He received his Ph.D. from Carnegie Mellon in 1987 and worked as a software architect in industry between 1987 and 1990. His research interests include software architecture, self-adaptive systems, formal methods, and cyber-physical systems. He is recognized as one of the founders of the field of software architecture, and, in particular, formal representation and analysis of architectural designs. He is a co-author of two books on software architecture: "Software Architecture: Perspectives on an Emerging Discipline", and "Documenting Software Architecture: Views and Beyond." In 2005 he received a Stevens Award Citation for "fundamental contributions to the development and understanding of software architecture as a discipline in software engineering." In 2011 he received the Outstanding Research award from ACM SIGSOFT for "significant and lasting software engineering research contributions through the development and promotion of software architecture." In 2016 he received the Allen Newell Award for Research Excellence. In 2017 he received the IEEE TCSE Distinguished Education Award and also the Nancy Mead Award for Excellence in Software Engineering Education He is a Fellow of the IEEE and ACM.



Pedro Fidalgo is the Engineering Director of Risk Management for R&D at Mobileum. He has a Msc in Computer Science from the University of Liverpool, UK and he is a PhD candidate in Complexity Sciences from (ISCTE-IUL), Portugal. His research interests include large-scale data mining with graphs, fraud, anomaly detection and telecom signalling protocols.