Leveraging the 5G architecture to mitigate amplification attacks

Matteo Repetto
CNR - IMATI
matteo.repetto@ge.imati.cnr.it

Alessandro Carrega
CNIT - S3ITI
alessandro.carrega@cnit.it

Guerino Lamanna
Infocom Srl
guerino.lamanna@infocomgenova.it

Jaloliddin Yusupov
Politecnico di Torino
jaloliddin.yusupov@polito.it

Orazio Toscano, Gianmarco Bruno, Michele Nuovo, Marco Cappelli
Ericsson Telecomunicazioni
{name.surname}@ericsson.com

***Abstract*—Volumetric (Distributed) Denial of Service attacks remain one of the major threats for any organization, capable of saturating most Internet access links through the usage of botnets and amplification techniques. The only effective mitigation mechanism today is the redirection of the network traffic towards scrubbing centers; this protects the Internet pipe of the victim, but does not prevent wasting resources in other parts of the network.**

In this paper, we leverage the cloud-native design of the 5G architecture to monitor traffic statistics at the edge of the network, which are then processed by a powerful Analytics ToolKit (ATk). Our work is based on the framework designed by the ASTRID project, which allows to automatically change the inspection probes while chasing a better balance between the granularity of the collected data and the overhead. We demonstrate our approach for an NTP amplification attack; the ATk is first trained with historical data and then used to detect deviations from the expected traffic profile, by switching between normal/warning/alert states. Our results show that it can correctly distinguish between periodical fluctuations of requests and attacks.

***Index Terms*—eBPF, syscall tracing, stegomalware, covert channels, detection.**

I. Introduction

Distributed Denial of Service (DDoS) attacks have often hit the headlines in the last years because they were able to saturate the Internet pipe of even larger organizations (e.g., GitHub).[1] To reach the necessary amount of traffic, "amplification" techniques are often used that exploit hundreds or thousands of buggy or misconfigured servers on the Internet. The typical attack pattern is shown in Fig. 1. It starts from a botnet of compromised nodes, which query a large number of servers while spoofing the IP address of the victim; such servers send to the victim responses that are far larger in size than the original messages that triggered them, hence generating the amplification effect.

There are many Internet protocols where small queries may trigger larger responses. Those exploited for DDoS amplification attacks use the UDP transport protocol, because this makes easier to spoof the IP address of the victim. Well-known examples include the Network Time Protocol (NTP) and the Domain Name System (DNS); however, the problem extends to other servers and protocols as well (Memcache, SIP, LDAP, RIP, SNMP). The relevant parameter is the "amplification factor," namely how many times the response is bigger than the original query; it may range from a few to thousand times, as shown in Table I.

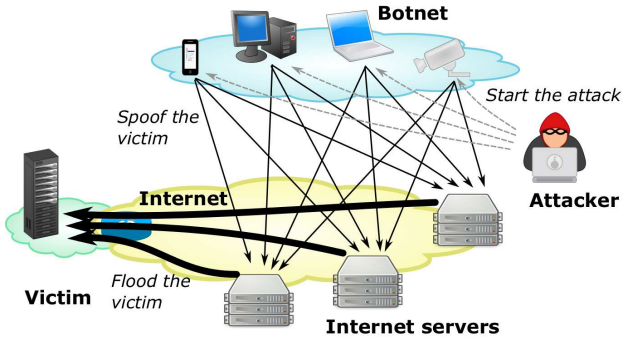


Fig. 1: Typical attack pattern for volumetric DDoS with amplification.

While the detection of a volumetric DDoS is trivial, effective mitigation is almost impossible for the victim, because these attacks saturate its Internet link. Today, the most effective defensive mechanism consists in diverting all traffic from the Internet to an external scrubbing center in case of attack [1]. This limits the impact to the time needed to detect the saturation and divert packets; typically a few dozens of minutes are required at most. However, such approach needs to model the capacity of the scrubbing center some times more than the biggest expected attack (e.g., four to ten), hence resulting in large resource overprovisioning, which should be continuously increased as the attackers can increase the

The final publication will be available at IEEE Xplore.
[1]https://www.wired.com/story/github-ddos-memcached/.

978-1-6654-0522-5/21/$31.00

Protocol/Server	Amplification factor
Memcached	10,000 to 51,000
NTP	556.9
CharGEN	358.8
QOTD	140.3
RIPv1	131.24
CLDAP	56 to 70
LDAP	46 to 70
DNS	28 to 54
Quake Network Protocol	63.9
TFTP	60
SSDP	30.8
MSSQL	25
Kad (P2P)	16.3
Portmap (RPCbind)	7 to 28
SNMP	6.3
Steam Protocol	5.5
NetBIOS	3.8
BitTorrent	3.8
Multicast DNS (mDNS)	2 to 10

TABLE I: Amplification factor for some Internet protocols and servers.

volume of their attacks.[2]

Mitigation of an amplification attack is challenging, because packets come from legitimate sources and carry valid data. Stopping it at its root would be possible in theory, by applying safe configurations to servers, blocking unnecessary ports, activating anti-spoofing filters; however, this is difficult to achieve in practice, because it depends on each organization that connects servers and devices to the Internet. With the advent of 5G technology, the number of (vulnerable) connected devices will increase, giving attackers more opportunities to create large botnets and to find vulnerable servers for amplifying their attacks. However, the same 5G architecture offers unprecedented opportunities to integrate monitoring and inspection functions at the edge, which can be used to detect and mitigate DoS attacks before they are amplified.

In this paper, we describe our solution for detecting amplification attacks through an Analytics Toolkit (ATk). Our approach builds on the framework developed by the ASTRID project, which was conceived a few years ago [2]. Our solutions is based on the identification of anomalies in the traffic statistics with respect to historical patterns; we combine coarse-grained traffic measurements and deep-packet inspection, which are used as preliminary indicators and for confirmation, respectively. The main innovations of our work consist in the following aspects:

- we leverage the cloud-native design of the 5G architecture to deploy the monitoring probes and analytics engine without any modification to the original standard;
- we switch between three different states (business-as-usual/warning/alert) that correspond to different measurement sets, pursuing a better balance between the granularity of the inspection process and the computing overhead;
- measurements are collected from multiple access networks and correlated, in order to improve the likelihood of detection and reduce the number of false positives.

Our results show that the ATk can distinguish between periodic fluctuations of requests and anomalies. The overhead for collecting the measurements is also quite limited, and the overall framework does not introduce relevant delays in the overall detection process.

The rest of this paper is organized as follows. We briefly introduce the ASTRID framework in Section II, and explain how the ATk operates in a closed control loop. In Section III we give a quick understanding of the 5G architecture and how the ASTRID framework is applied to it; then, we discuss the scenario for the NTP amplification attack in Section IV. We describe the implementation of the ATk and its business logic in Section V. We provide preliminary functional validation and performance analysis in Section VI, and then compare our approach to existing work in Section VII. Finally, we give our conclusion in Section VIII.

II. The ASTRID framework

The ASTRID framework was born to decouple detection and analytics services from the implementation of virtualized services (both hosted in virtual machines and containers), and to allow more flexibility in the design and operation of security processing pipelines.

Fig. 2 shows a simplified view of the ASTRID software architecture, tailored to the specific processing pipeline created for the ATk. In the ASTRID framework, an external software orchestrator is used to deploy a virtual service, starting from a descriptive template; in our work, we used Kubernetes for this purpose. The only requirement for the service deployment process is to include ASTRID agents within each virtual function; this is quite easy for Kubernetes, since ASTRID agents are already delivered as Docker containers. Several monitoring and inspection agents have been integrated in the architecture, including plain Elastic beats, Logstash, and purpose-specific agents developed by the project (mostly consisting in inspection tools within the Polycube[3] framework that run eBPF programs). A special kind of agent, named *Local Control Plane* (LCP), acts as common control point to all other

[2]Less than two years after the unprecedented 1.35 Tbps DDoS attack experienced by GitHub, AWS reported to have defeated against a 2.3 Tbps attack in February 2020, which almost doubled the previous volume. During the same period, Imperva reported one of its client to have experienced a 500 million packets-per-second attack, which represents the most intensive DDoS attack against network infrastructure in the history of the Internet.

[3]https://polycube.network/.

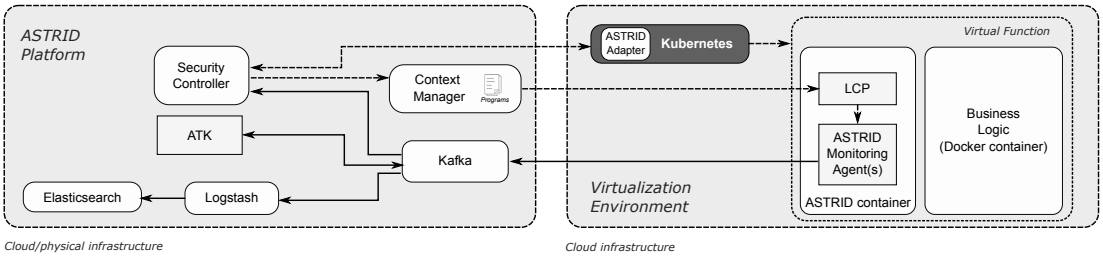


Fig. 2: Simplified view of the ASTRID software architecture.

agents, including support for different configuration methods (yaml files, REST APIs, command line).

The framework also assumes that a management API is exposed by the service orchestrator; this interface is used to collect information about the current service topology and its configuration.[4] In this case, we developed an ASTRID adapter for retrieving all the necessary information which is not provided by the plain Kubernetes management APIs.

It is worth noting that ASTRID does not envision any interaction with the underlying infrastructure: this is a specific design choice, that allows to support multi- and cross-cloud deployments in a transparent way, as well as externalization of security services.[5]

The architecture builds on and extends the well-known and proven Elastic Stack, by collecting events and measurements on a Kafka bus and delivering them for real-time processing (the ATk in our work) and internal storage (Elasticsearch). Events generated by the ATk are again published on Kafka, but using a different topic, and consumed by the dashboard (not shown in the picture, since it is not relevant in this work), Elasticsearch, and the Security Controller. The latter is a rule management system based on Drools that takes control and management actions based on internal policies associated to the specific detection service. The definition of policies follows an Event-Condition-Action pattern, which triggers one or more action when an event occurs, if some context conditions are satisfied. Finally, a Context Manager provides the abstraction of the whole system, including the description of the service topology and its configuration, the list of available agents and network parameters of the LCP, parameters that can be changed for each agent, eBPF programs available, etc. It exposes a common REST interface for changing the configuration of local agents deployed within the virtual service, independent of specific protocols and syntax.

Differently from existing commercial and open-source tools, the ASTRID architecture allows more flexibility in combining together different agents and analytics engines, following a programming model that supports both streaming and off-line analysis. This allow the creation of custom processing pipelines, as the one for the ATk used in this work. The detailed description is available in technical documents [3].

[4]The main target for ASTRID are elastic cloud services that can change at run-time according to orchestration policies (e.g., scaling, backup, redundancy).

[5]The lack of visibility over the physical infrastructure of course affects the threat model for the framework.

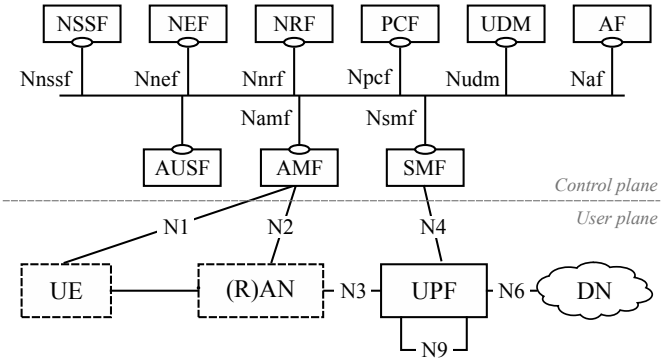
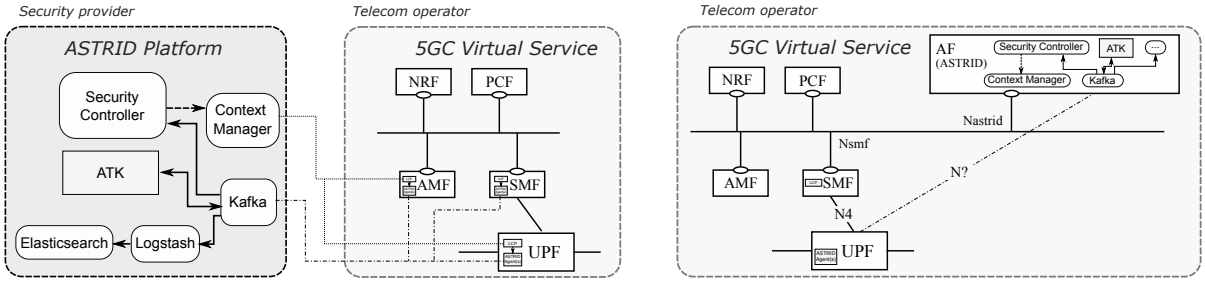


Fig. 3: ETSI architecture for the 5G core (5GC). The UE and AN are shown for reference, but they are not part of the 5GC.

III. Inspecting packets in the 5GC

Beyond the delivery of faster and broader air interfaces, 5G introduces a ground-breaking approach in the internal network architecture [4]. Even if a similar distinction to 4G between the access and core parts still exists, the new standard reflects the main advances in software-defined networking and moves to a cloud-native approach for the core, where each functional entity is now modeled as a (virtual) Network Function (NF). This evolution is not limited to the nomenclature and the delivery of software instances instead of hardware appliances, but it also encompasses a sharper distinction between the control and user plane and a large transition from reference points (N[0-9]) towards service-based interfaces (*Nxxx*), at least in the control plane (see Fig. 3).

The service-based architecture facilitates both planning and management of the whole infrastructure. On the one hand, it allows more flexibility in placing NFs across centralized offices and computing/storage resources at the edge, with better support for latency-sensitive applications and user mobility. On the other hand, multiple NFs instances can be deploed to create different network "slices", dedicated to vertical applications or user groups. The description of the 5GC architecture falls outside the scope of this paper; here, we only briefly elaborate on the user plane, especially the User Plane Function (UPF), which plays a crucial role in our approach.



(a) The ASTRID platform monitors the virtual service.

(b) The ASTRID platform integrated in the 5GC.

Fig. 4: Alternative options for creating detection and analytics services for the 5GC.

In the 5G architecture, the user plane carries Packet Data Units (PDUs) – Ethernet, IPv4 or IPv6 frames – from the User Equipment (UE) to an external Data Network (DN), and viceversa. A PDU Session is created between the UE and the UPF, which encapsulates PDUs and carries them through the Access Network (AN); indeed, different tunnels are created on the air and wired interfaces, with the AN relaying packets in between. The UPF is therefore the outermost NF in the 5G network that processes IP packets generated from or intended to the UE; it is mostly conceived for packet steering to/from multiple data networks, for anchoring in case of mobility, and for QoS and traffic policies enforcement.

Even if this is not explicitly considered by the 3GPP specification (not yet, at least), the UPF appears a perfect point for traffic monitoring at the edge. As a matter of fact, UPFs are software functions that can be easily replicated for scalability, therefore they probably represent the only point in the 5GC (and also the Internet) where packets can be inspected with fine granularity.

A. Monitoring the 5GC virtual service

Overall, the 5GC can be described by a service template as any other cloud service, and indeed there are already Kubernetes implementations available.[6] Therefore, the ASTRID framework described in Section II can be used for attack detection, by embedding agents in one or more VFs and connecting them to the centralized platform. This is transparent to the 5G architecture, because these agents do not affect the internal protocols and are controlled by an external entity. Further, the ASTRID platform can be an external standalone component or deployed as additional function of the 5GC template.

Fig. 4a depicts the expected architecture in this scenario, limiting to a few functions for the sake of brevity. ASTRID probes can be deployed both in the user plane and in the control plane. The former can be used for packet inspection, whereas the latter for log and event collection, software tracing, and integrity verification. This brings the opportunity for a large number of detection and analytics engines, addressing the need for both trustworthy operation of the 5G network itself and protection of the UE/DN.

[6]See, for example, Open5GS: https://open5gs.org/.

This deployment option best fits the original concept behind the design of the ASTRID framework, so we follow it in our work.

B. Integrating detection services in the 5GC

An alternative integration option is to deploy the ASTRID platform as NF. This approach leverages the 5G service-based architecture, and the possibility to integrate external Application Functions (AFs) delivered by third parties for specific control and management tasks. Without claiming to define a complete and fully compliant solution, we sketch in Fig. 4b an indicative example of how this can be implemented. In this case, the ASTRID platform becomes part of the 5G architecture, which should be extended with additional service interfaces and reference points.

Detection and analytics services may therefore become an AF, managed by either directly the telecom operator or an external security provider. The monitoring and inspection capabilities of ASTRID agents may be exposed by an additional service in the SMF (e.g., *Nsmf_Monitoring*); in this case, the ASTRID LCP would be deployed in the SMF rather than in the same VF as the monitoring agents. An additional reference point would be needed to collect data from agents with a streaming pattern. As for all other NFs and services, also the availability and location of the *Nsmf_Monitoring* can be discovered through the NEF.

At a preliminary analysis, it seems that this approach fits well the need for monitoring the UE traffic in the UPF, but its application to monitoring of other NFs is not straightforward. Anyway, we do not further consider it in our work.

IV. NTP AMPLIFICATION USE CASE

One relevant use case for 5G is the Internet of Things (IoT), which is expected to drastically increase the number of connected devices to the Internet. However, we argue that this will also enlarge the amount of potentially vulnerable nodes, because of the typical weak security posture of IoT devices, hence offering attackers the opportunity to create huge botnets. In this respect, packet inspection at the network edge is an effective strategy to mitigate the attack before they propagate to and are amplified on the Internet, hence saving resources both in the 5GC as well as on connected data networks.

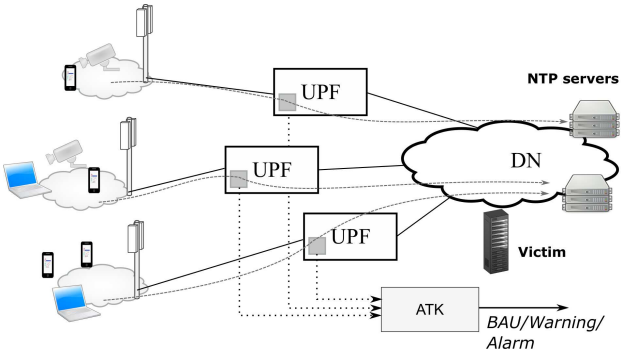


Fig. 5: The use case for NTP amplification attack.

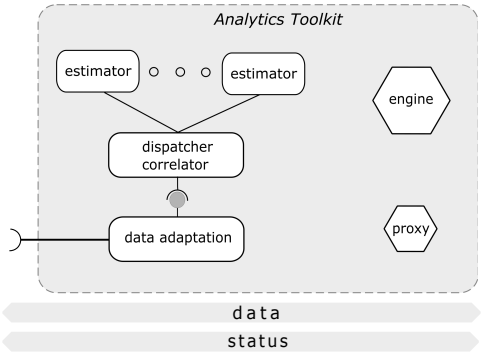


Fig. 6: ATk architecture.

Among the broad set of Internet servers that can be exploited for amplification, we select the Network Time Protocol (NTP) because it has one of the largest amplification factors and a huge number of public servers; indeed, NTP is among the top emerging network attack vectors.[7] However, our approach can be applied by other protocols as well, by simply changing the configuration of the monitoring agent.

The NTP attack is based on sending a command called *monlist* to an NTP server; the server returns the addresses of up to the last 600 machines that it has interacted with. The request packet is only 234 bytes long, but the response may sum up to several dozens of kilobytes, depending on the number of returned addresses.

Fig. 5 shows a simplified representation of the use case investigated in this paper. We monitor NTP packets in the UPF of the 5GC network, as described in Section III. We switch between two kinds of measurements, depending on the current context:

- a *coarse-grained* indication: the number of NTP packets seen;
- a *fine-grained* indication: the number of NTP packets containing the *monlist* command.

The second parameter is more suited for the purpose of detecting the amplification attack; however, it needs deep packet inspection and that usually entails slower processing.[8]

The next Section describes how the ATk processes the two measures and switches between them at run-time, chasing an optimal balance between the overhead of the inspection and accuracy of the detection.

V. The Analytics ToolKit

The Analytics ToolKit (ATk) is the component responsible for analyzing quasi real-time data, performing estimations based on historical time-series and finding anomalies. It is therefore a general-purpose tool, which can be applied to a plurality of attacks. Its behavior can also be easily adapted to multiple use cases, through the rich REST-based configuration API.[9]

The implementation of the ATk is compliant with the ASTRID framework. Its internal architecture is shown in Fig. 6 and based on two microservices: the *proxy* and the *engine*.

The *proxy* interfaces the internal logic with external components, by reading/writing from/to the common message bus and by exposing the REST configuration API. Therefore, it implements a *data adaptation* function. In particular, the proxy aggregates the data published asynchronously by the each monitoring agent j (*probe*) located in the set of network nodes $\mathcal{N}$ at each time instant t: coarse grained measurements (namely, volume of NTP packets) denoted by f_t^j, and fine-grained measurements (namely, volume of *monlist* requests) denoted by d_t^j. In general, f_t^j and d_t^j can be arrays, though in this use-case they are scalars. Nodes are grouped into a set of A areas $\mathcal{A}_i$, based on their geographical location ($\mathcal{A}_i \subseteq \mathcal{N}, \quad \bigcup_i \mathcal{A}_i = \mathcal{N} \ \ \forall i = 1, \ldots, A$).

The *engine* includes a battery of $N = |\mathcal{N}|$ *estimators*, each one processing the data from a different node, which work under the control of the *Estimator Manager*. The latter implements the main detection logic. It forwards f_t^j collected by the *proxy* to the j-th *estimator*, which computes the forecast based on what it learned in the past; the result is then returned to the *Estimator Manager* for changing the current status and performing correlations between different nodes (see below). Finally, both the current status and correlations are pushed to the *proxy* and published on the Kafka bus.

Each *estimator* j is responsible to predict the value $\hat{f}_t^j$ expected at time t, by using the previous T values (from $t-1$ downto $t-T$). Their implementation makes use of standard ARIMA (AutoRegressive Integrated Moving Average) models provided by Python *statsmodel* library. The larger T, the more daily, weekly and seasonal patterns can be captured by the *estimators*. A benefit of the current architecture is that the number of *estimators*, which are the most computationally heavy functions, can scale horizontally with the number of nodes and/or use-cases to keep the ATk responsive without consuming unnecessary cluster resources.

[7] See https://blog.cloudflare.com/network-layer-ddos-attack-trends-for-q3-2020.

[8] The number of NTP packets can be easily computed by hardware appliances based on the destination port of UDP packets. The detection of *monlist* packets requires a software approach, since it cannot be generalized for any protocol.

[9] Astrid analytic toolkit API: https://github.com/astrid-project/analytic-toolkit-apis.

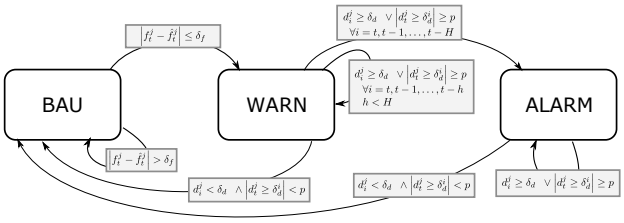


Fig. 7: ATk state diagram.

The behavior of the ATk is governed by the state machine shown in Fig. 7. The three states reflect normal (*BAU*), suspicious (*WARN*) and attack conditions (*ALARM*). The state machine is then re-evaluated every Δt, by comparing the current measurement f_t^j with the estimation $\hat{f}_t^j$ computed from the previous T values.

The ATk persists in *BAU* as long as $\left|f_t^j - \hat{f}_t^j\right| \leq \delta_f$, where δ_f is a fixed threshold, otherwise it switches to the *WARN* state. To start the process, a time-series with T "clean" measurements must be available, i.e. coming from nodes which are not under attack, in order to learn the periodic behavior. During operation, the new measurements f_t^j are continuously added to the time-series to continue training; however, real values are replaced by their estimation $\hat{f}_t^j$ when the state is not *BAU*, to prevent overfitting.

In the *WARN* state, fine-grained measurements d_t^j are also collected to improve the accuracy of the detection. These values are compared with two thresholds δ_d and δ_d^i that account for individual deviations and correlated variations in area $\mathcal{A}_i$, respectively. The ATk switches and remains in the *ALARM* state when either one node exceeds the threshold δ_d or p nodes in the same area $\mathcal{A}_i$ exceed the threshold δ_d^i, in both cases this should happen for H consecutive evaluations. This can be formally described as:

$$\exists j \in \mathcal{N} : d_i^j \geq \delta_d \ \vee \\ \exists \mathcal{A}_i \ (i = 1, \ldots, A) : \left|\{j \in \mathcal{A}_i\} : d_t^j \geq \delta_d^i\right| \geq p \\ \forall i = t, t-1, \ldots, t-H \quad (1)$$

The first equation allows to identify attacks that go through a single node; the second equation is more suitable for very distributed attacks, which affect a plurality of nodes of the same area with smaller deviations.

VI. Numerical Evaluation

We set up an experimental testbed to carry out functional and performance validation. We used the last releases of the ASTRID framework and ATk available from the project repository.[10] All components are available as Docker images and Helm charts are available for automatic deployment with Kubernetes.

For packet inspection we used the *Dynmon* agent[11] (a service of the Polycube framework), which provides a common control plane for running different eBPF programs and collecting their measurements. We also developed policies for the Security Controller; they are used to push two different eBPF programs to *Dynmon*: one collects coarse-grained measurements, and the other one also includes fine-grained measurements, as described in Section IV.

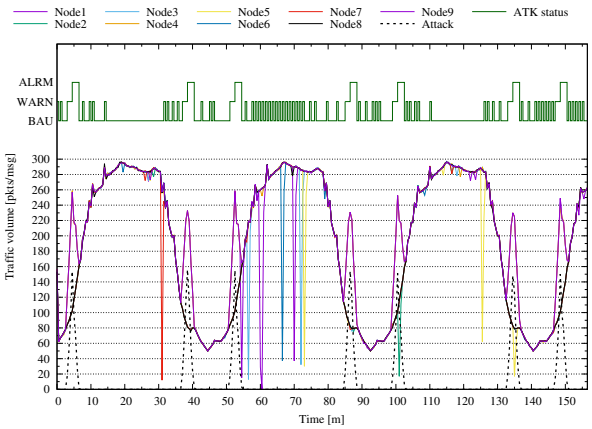


Fig. 8: Measurements reported by the network probes and ATk state.

The testbed implements the scenario depicted in Fig. 5, with 9 UPFs connected to the same DN. These nodes are grouped into 3 non-overlapping areas. The ATk was trained offline with a wave profile, which is representative of typical daily variations.[12]

At this stage, we only focus on the ATk and the ASTRID framework, so we didn't deploy a working 5G network but replaced UEs with software Traffic Generators (TGs). Each TG creates NTP packets according to different traffic profiles. A subset of 3 TGs emulated a small botnet that generates *monlist* packets; they can be placed in the same or different areas.

Fig. 8 shows the measurements reported to the ATk and how its internal state evolves during the simulation. The same generation profile was used for all nodes, so the graphs largely overlap. Even if the attack was quite limited in size (the number of queries was below those generated during peak periods), the ATk correctly generated a warning only when the anomaly occurred.

Beyond the operation of the ATk engine itself, some delays can be introduced by the ASTRID framework, while changing the configurations of the network agents. We therefore investigated in details the composition of this delay, by splitting it into two components: processing introduced by the Context Manager (CM) and configuration of the local agent (Polycube). Table II shows that the delay is rather limited, around a couple of seconds, and have a marginal impact on the timescale of the ATk. Overall, the major impact is due to loading the eBPF program into the kernel, because this operation implies

[10] https://github.com/astrid-project/astrid-framework.

[11] https://polycube-network.readthedocs.io/en/latest/services/pcn-dynmon/dynmon.html.

[12] The whole profile is generated in around one hour, to keep the simulation time acceptable.

Operation	CM	Polycube	Total
Add coarse-grained meas.	0.0221	2.3792	2.4013
Remove coarse-grained meas.	0.1628	0.0296	0.1924
Add fine-grained meas.	0.0502	2.4033	2.4535
Remove fine-grained meas.	0.1726	0.0302	0.2028

TABLE II: Breakdown of the delay (in seconds) to push an eBPF program to the network agent.

compilation and code verification. Oddly, the CM is slower in the removal process; this is due to the internal implementation that stores the current configuration in Elasticsearch and needs more time to remove an element. Finally, each message sent on the kafka bus is rather small (around 100 bytes) and sent every few minutes, so the communication overhead is negligible.

VII. Related work

After the amplification DDoS attacks hit the headlines around 10 years ago, several researchers have started investigating the pattern and impact over the global Internet [5], [6]. Due to the difficulty to monitor the public Internet, the interest has often focused on the prevention of these attacks [7]. However, some authors tried to identify relevant features in historical data [8].

More recently, the advent of Software-Defined Networking has revived the interest in detection, especially by applying machine learning techniques that use generic network features like the size of packets, duration of the connection, etc. [9]–[13]. Since the training is not trivial with real traffic, reinforcement learning was also investigated that learns directly from traffic [14].

Some authors also explicitly considered the geographical dimension of the problem [15]. However, we argue that SDN is not capillary used in practice, especially in larger telco's networks. Finally, beyond the analysis of network traffic, there are also authors that consider logs for the configuration of firewall rules [16].

VIII. Conclusion

In this paper, we have proposed a concrete application of the ASTRID framework to detect amplification attacks in virtualized 5G networks. Our approach does not require any modification to the 5G architecture and protocols, but it only needs the addition of simple agents. This could be easily done in Kubernetes-based deployments by including an additional container within the same POD of the UPF function. This simplifies the portability across following releases and upcoming architectures, hence realizing a good separation of concerns between network management and security processes.

Preliminary performance analysis shows that small delays are introduced by the framework itself, which have a negligible impact on the specific detection process. By combining volumetric measures with deep packet inspection we can easily monitor large distributed infrastructures, and scale well with the number of users and access networks. Future work will investigate more in detail the effectiveness and accuracy of the detection with different traffic profiles and attack patterns, also including the identification of correlations.

Acknowledgment

This work has received funding from the European Commission, Grant Numbers: 786922 (ASTRID) and 833456 (GUARD).

References

[1] Radware, "Cloud DDoS protection service: attack lifecycle under the hood," Technology Overview Whitepaper,, 2016. [Online]. Available: https://www.radware.com/assets/0/314/6442477977/2c6454b4-403b-45b1-ac58-dc628bc210b3.pdf

[2] S. Covaci, R. Rapuzzi, M. Repetto, and F. Risso, "A new paradigm to address threats for virtualized services," in *IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, Tokyo, Japan, Jul. 23rd-27th, 2018, pp. 689–694.

[3] M. Repetto, "Final astrid architecture," The ASTRID Consortium, Tech. Rep. D1.3, October 2020, v. 1.0. [Online]. Available: https://private.astrid-project.eu/Documents/PublicDownload/79

[4] "5g; system architecture for the 5g system," ETSI, Tech. Rep. TS 123 501 V15.3.0, September 2018, 3GPP TS 23.501 version 15.3.0 Release 15.

[5] J. Czyz, M. G. Kallitsis, M. Gharaibeh, C. Papadopoulos, M. D. Bailey, and M. Karir, "Taming the 800 pound gorilla: The rise and decline of NTP DDoS attacks," in *Proceedings of the 2014 Conference on Internet Measurement Conference (IMC '14)*, Vancouver, BC – Canada, November 2014, pp. 435–448.

[6] A. Wang, W. Chang, S. Chen, and A. Mohaisen, "Delving into internet DDoS attacks by botnets: Characterization and analysis," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2843–2855, December 2018.

[7] C. Rossow, "Amplification hell: Revisiting network protocols for ddos abuse," in *In Proceedings of the 2014 Network and Distributed System Security Symposium, NDSS*, 2014.

[8] L. Cai, Y. Feng, J. Kawamoto, and K. Sakurai, "A behavior-based method for detecting DNS amplification attacks," in *10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, Fukuoka, Japan, Jul., 6th–8th, 2016, pp. 608–613.

[9] M. E. Ahmed, H. Kim, and M. Park, "Mitigating dns query-based ddos attacks with machine learning on software-defined networking," in *IEEE Military Communications Conference (MILCOM)*, Baltimore, MD – USA, Oct., 23rd–25th, 2017, pp. 11–16.

[10] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *International Conference on Wireless Networks and Mobile Communications (WINCOM)*, Fez, Morocco, Oct., 26th–29th, 2016.

[11] Y. Zhauniarovich and P. Dodia, "Sorting the garbage: Filtering out DRDoS amplification traffic in ISP networks," in *IEEE Conference on Network Softwarization (NetSoft)*, Paris, France, Jun., 24th–28th 2019.

[12] M. Han, T. N. Canh, S. C. Noh, J. Yi, and M. Park, "DAAD: DNS amplification attack defender in SDN," in *International Conference on Information and Communication Technology Convergence (ICTC)*, Oct., 16th–18th 2019.

[13] K. Özdin cer and H. A. Mantar, "SDN-based detection and mitigation system for DNS amplification attacks," in *3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, Ankara, Turkey,, Oct., 11th–13th 2019.

[14] Y. Zhang and Y. Cheng, "An amplification DDoS attack defence mechanism using reinforcement learning," in *IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Leicester, United Kingdom, Aug., 19th–23rd 2019.

[15] V. Gupta and E. Sharma, "Mitigating DNS amplification attacks using a set of geographically distributed SDN routers," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Bangalore, India, Sep., 19th–22nd, 2018.

[16] A. S. Jose and A. Binu, "Automatic detection and rectification of DNS reflection amplification attacks with hadoop mapreduce and chukwa," in *Fourth International Conference on Advances in Computing and Communications*, Cochin, India, Aug., 27th–29th, 2014.