DEPLOYMENT OF THE FED4FIRE+ TESTBED FOR FORENSICS VISUALIZATION PURPOSES

Leonidas Kallipolitis, Panagiotis Katrakazas and Ilias Spais
Zelus P.C., Athens, Greece, {l.kallipolitis, p.katrakazas, ilias.spais}@zelus.gr

NOTE: Corresponding author: Panagiotis Katrakazas, p.katrakazas@zelus.gr

Abstract *– A security incident or rule violation can be detected and documented using forensic analysis, which is made easier by preconfigured views that are enhanced with crucial data. In this paper, we present an advanced visualization mechanism for digital forensics that increases the situational awareness of a security expert by analysing and presenting security events, alarms and critical performance indicators. Using testbeds made available by Fed4FIRE+, we demonstrate an experimentation setup that simulates genuine client settings, including their varying needs and differences in size and requirements. These tests allowed for the parameterization of the variables, which led to rapid and well-documented results that could only be reached by trial and error with potential financial repercussions.*

Keywords – Cybersecurity, digital forensics toolkit, Fed4FIRE+ testbed, network topologies, scenarios visualization

1. INTRODUCTION

Handling cybersecurity risks and vulnerabilities can be particularly inefficient and time-consuming due to budget restrictions and a lack of security expert personnel [1]. At the same time security data is rapidly growing, leaving organizations overwhelmed by the scaling needs for collecting, storing and analysing digital evidence while maintaining the security and integrity of data [2]. This results in severe consequences, legal and/or business value related, for organizations being hit by a security incident.

One of the biggest challenges for organizations and researchers is the correlation of forensic data collected by disparate cyber-centric security procedures and technologies, i.e., Firewalls (FWs), Intrusion Detection Systems [IDSs], and Intrusion Prevention Systems, (IPSs), with device and control systems logging data [3]. This must take place in a reliable and at the same time affordable manner for the financially limited Small and Medium-sized Enterprises (SMEs) [4].

The challenges identified earlier have been the topic of research in recent attempts, including data extraction from mobile devices [5], [6], the continuously increasing realm of the Internet of Things [7] and the computer forensic processes in industrial control systems and automotive IT [8]. As the field of digital evidence is constantly growing [9], the security forensics tools which are available on the market cover different perspectives of digital forensics. Some of them are quite specific, like the CapAnalysis tool[1] used for network forensics, while others include a large set of tools, like the CAINE[2] and the EnCase platforms[3].

Investing in a strong visualization functionality that relies on preconfigured views of visual components and temporal analysis of digital evidence will increase the situational awareness of a security expert, thus resulting in faster investigations of security incidents and even the prevention of them. The accuracy and effectiveness of the preconfigured views relies on the expertise of the security specialist, as well as the already successfully conducted forensics analyses.

Experiments, like the one offered by Fed4FIRE+ (F4F+)[4], assist with testing and assessing solutions like the one described in the previous paragraph. If F4F+ was not available, probably commercial testbeds would have been used to allow executions of such experiments. This would require extra budget and time allocation of human resources that would have to exert significant effort to set up the experiment. Moreover, deciding on the proper testbed would also take some time and desktop research to find an appropriate offering. Usual problems that can occur when dealing with commercial services could include inadequate support and documentation, missing features or difficulty in management of the experiment.

[1] https://www.capanalysis.net/ca/
[2] https://www.caine-live.net/
[3] https://security.opentext.com/encase-forensic
[4] https://www.fed4fire.eu/

The rest of the paper is structured as such: Section 2 briefly presents the Forensics Visualization Toolkit (FVT) and its functionalities, while Section 3 refers to the F4F+ experimentation and testbed setup. Section 4 presents the results of our approach and we discuss some points and outcomes in Section 5.

2. THE FORENSICS VISUALIZATION TOOLKIT

The Forensics Visualization Toolkit (hereinafter, FVT) of the digital forensics toolkit aims to improve the speed and accuracy of collecting, monitoring, recovering and protecting digital evidence in the event of a security incident being committed. To achieve these goals, FVT invests in strong visualization functionalities that enable operators to quickly gain situational awareness and minimize the time needed to identify the root cause of security incidents. Employing a set of agents deployed in a customer's local network, FVT not only allows digital forensic analysis but it can be also used for monitoring purposes, complementary to existing security solutions (e.g. Security Information and Event Management (SIEM), antivirus, etc). Therefore, visual exploration of data, adaptability to various network setups and scalability potential need to be verified soon on the road to commercialization.

The conceptual architecture that realizes this concept is depicted in Fig. 1. Presentation functionalities include timeline analysis (e.g., time inspection of system metrics and events in different levels of granularity and comparison of system status at different times), preconfigured views (predefined setups of dashboards according to prior usage statistics) and advanced visualizations

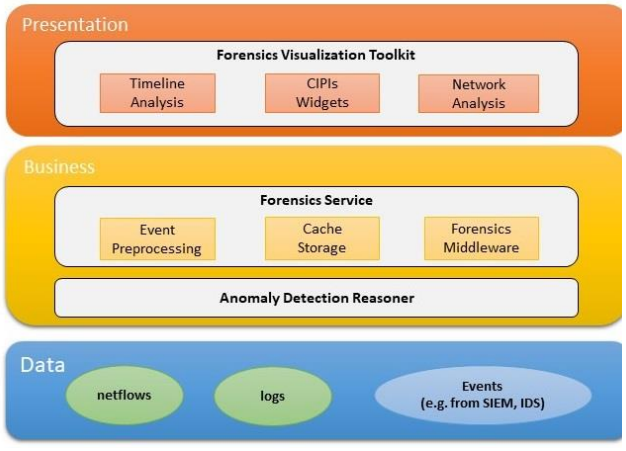


Fig. 1 – High-level architecture of the FVT tool

with multiple filtering/grouping options. All these allow for the analysis and the monitoring of Company Infrastructure Performance Indicators (CIPIs) (e.g., internal network traffic) as well as for detected events (e.g., security related or system events).

Events are handled by the anomaly detection reasoner, which enables an FVT's forensics service layer to consume the ones coming from existing event detection systems and complement them with others detected using its own rule-based event detection methods which are tailor-made to every installation. Data to support these functionalities is collected by existing security systems such as SIEM or IDS systems, log files and other utilities or monitoring software via a set of agents. These agents can be deployed in a customer's local network, and they enable the FVT to not only allow digital forensic analysis but to be also used for monitoring purposes, complementary to existing security solutions.

3. FED4FIRE+ EXPERIMENTATION DESCRIPTION AND SETUP

The goal of the Fed4FIRE+ (F4F+) experimentation was to fine-tune the existing functionality and further enhance FVTs in order to serve potential customers with different needs regarding resources, network topologies and generated data volumes. The main objectives of the experiment were: (i) to validate if the tool works as intended; (ii) if it can handle the foreseen data load; (iii) if the designed functionalities are easy to use and fit-for-purpose; and (iv) whether there should be any changes or additions to strengthen the tool. In addition, running F4F+ experiments were explored to give the opportunity to run and test the security scenarios we have designed and enrich the DFT product offering with tested and valid functionalities.

In particular, we first experimented with up to 10-15 nodes, simulating various conditions and business needs regarding the frequency of node communication that generate different amounts of data and have different security needs and network infrastructures. In addition, we simulated security scenarios and conditions that may trigger security alarms to be analysed by the proposed forensics services, e.g., unexpected network traffic and high CPU loads.

The objective was to use the results of experiments so as to get vital feedback regarding the following

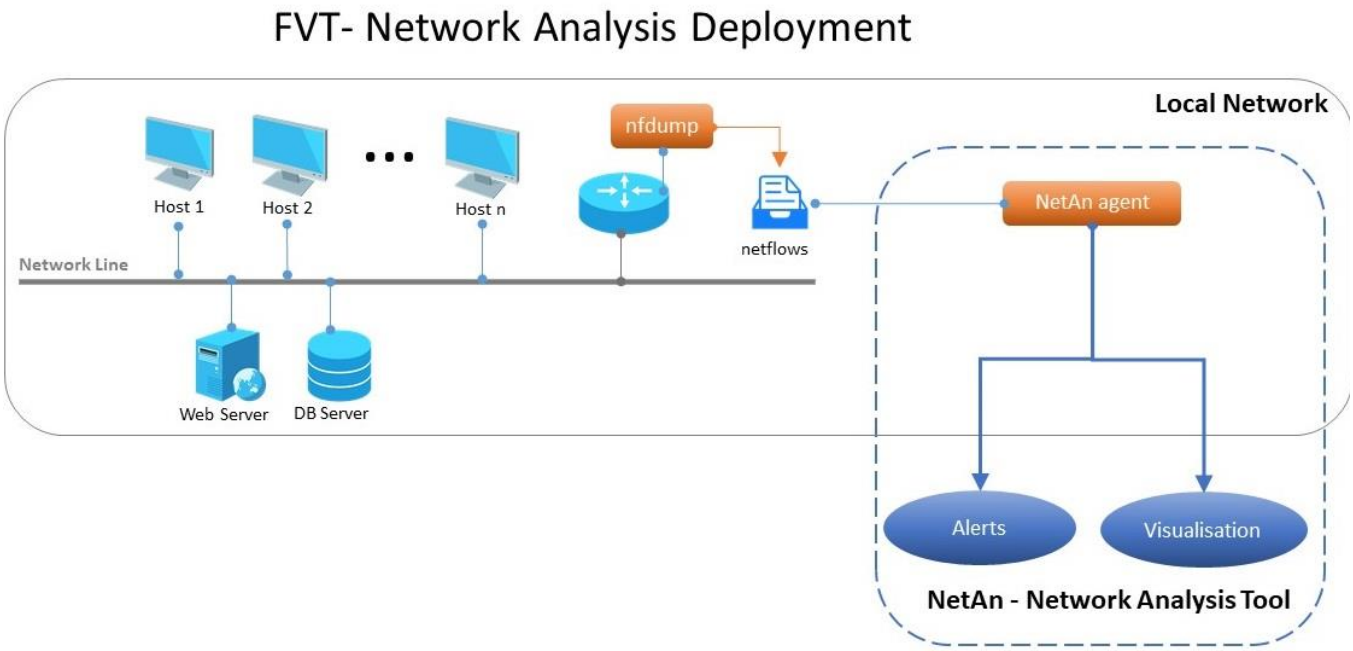


Fig. 2 - Network analysis deployment schema

questions: (i) Does the application as a whole work as intended? (ii) Do all features/visualization components function as expected? (iii) Can the application withstand the demands of a heavy load? (iv) Is the application reasonably easy to use, and (v) What changes should be made to existing components and what new visualizations or preconfigured views should be added?

Although the whole FVT was deployed in order to conduct the experiment with F4F+ facilities, special focus was given to the NetFlow Analysis (NetAn) tool which provides a visualization of a local network's activity together with a set of filters and grouping functions that enable the easy exploration of the network traffic and minimizes the time needed to perform various network-related operations, e.g. identification of nodes with highest traffic or of external IPs that communicate with the network.

NetAn processes and visualizes NetFlow[5] data that is pulled from nfdump[6] files including information about the network flows between a monitored machine and other hosts (of the internal developed by Cisco for collecting IP traffic information and monitoring network traffic. It includes information about network flows between two hosts, e.g., start/end times of the flow, duration, used protocol, source/destination IPs, etc. Nfdump is a toolset to collect and process NetFlows. It stores the monitored flows in binary format or externally). NetFlow is a network protocol which files and provides a command-line-based facility to query these binary files for flows using a number of filters, e.g., all the flows of a specified day or all the flows originating from a specific IP.

NetAn employs an agent that monitors nfdump files and generates alerts based on a set of dynamic rules that can be adapted to the specificities of the monitored network. This way alerts can be tailored according to network usage properties of any given organization and provide business-specific notifications to end users. NetAn's agent is deployed locally inside the monitored networks. On the other hand, the visualization part of NetAn can be either locally deployed also or in the cloud. Information is sent by the agent using a secure communication method based on web services. Fig. 2 presents a high-level deployment setup.

The main idea is that FVT is deployed on a machine inside the local network. This serves as the FVT central logs and events collector (FVT agent) and also hosts the front-end part of the tool that serves the visualizations. The NetAn agent depicted in the figure is responsible for querying the NetFlows collected from all network nodes and serve them in the appropriate format to the visualization layer.

[5] https://www.cisco.com/c/en/us/products/ios-nx-os-software/netflow-version-9/index.html,

[6] https://github.com/phaag/nfdump

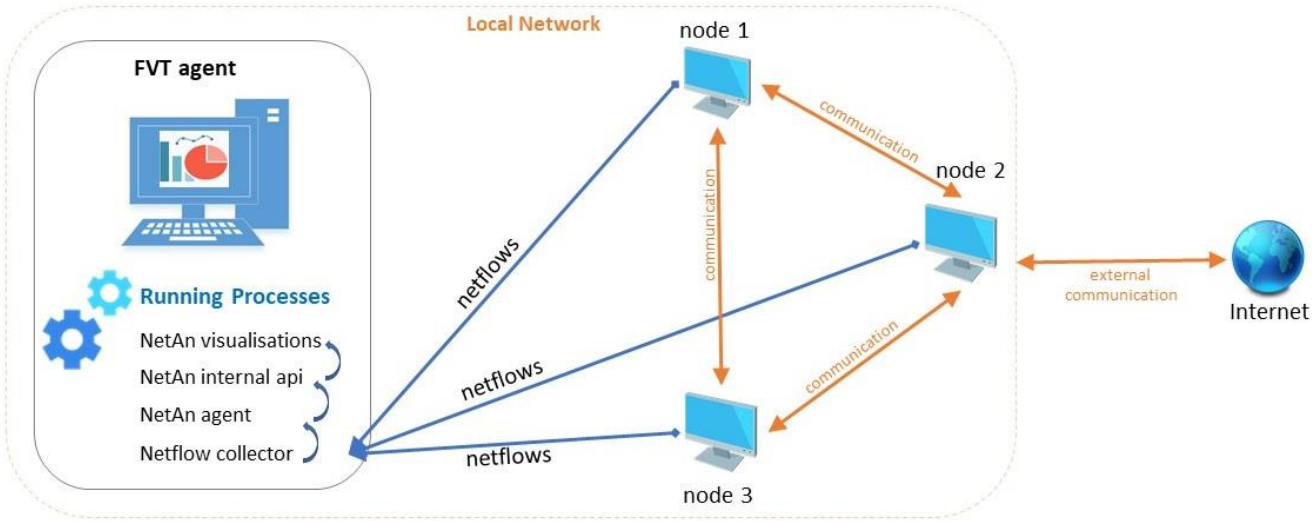


Fig. 3 - NetFlow connection

Alerts are also issued according to rules that have been set up according to the network's nominal activity.

The conducted experiments simulated networks of 3-15 nodes. One of the nodes serves as the FVT agent and the rest of them are connected to each other and send various messages inside or outside the network. Probes have been set up to every node so that the inbound and outbound traffic of each node is sent to the NetFlow collector running at the FVT agent. This results in a full record of NetFlows that allows the analysis of network traffic by the operator. Fig. 3 shows the deployed utilities and

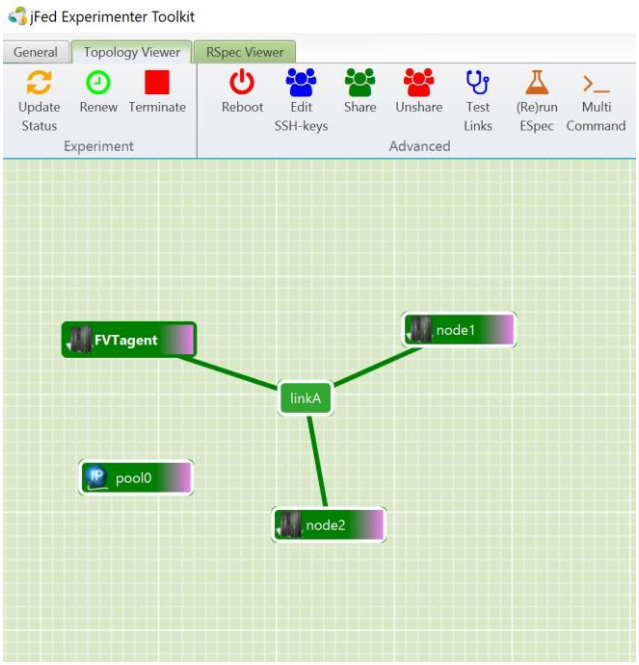


Fig. 4 - Experiment with 3 nodes

collection of NetFlows from all nodes by the FVT agent.

As shown in Fig. 3, all nodes of the experiment communicate with each other and externally to the Internet. Every node sends its NetFlows to the NetFlow collector residing at the FVT agent. The NetAn agent then takes the flows and serves them via an internal API to the visualizations. Fig. 4 is a snapshot of the topology used in the experiment, the one with the minimum nodes (3), while another one was created with the maximum number (15). It should be noted that we experimented with the 'Shared LAN" option to have two different running experiments talk to each other.

4. RESULTS

The technical results of the experiment revealed a number of actions that we had to take in order to better set up the installation and configuration procedures but also exposed the need for updates to existing or the addition of new features at the visualization that the end user interacts with. In this section we initially present a series of screenshots of the deployed tool during an experiment to briefly describe the offered functionality and we then analyse the technical results of the experiment and how they affected our tool.

4.1 FVT – NetFlow Analysis (NetAn) tool overview

Fig. 5 presents the region overview of the monitored network. We see the monitored nodes and get some initial charts of CIPIs for these nodes.

Fig. 5- FVT - region overview

Fig. 6 depicts the initial view of the NetFlow analysis tool, NetAn. The core visualization is based on a chord diagram where edges represent IP addresses and chords the number of NetFlows exchanged between them. A series of filters are available to help operators execute various operations like inspecting a specific time period, filtering IP addresses or defining the direction of the NetFlows that they want to explore. Filters work in an 'AND' logic, meaning that results satisfy all the filters that have been given a value.

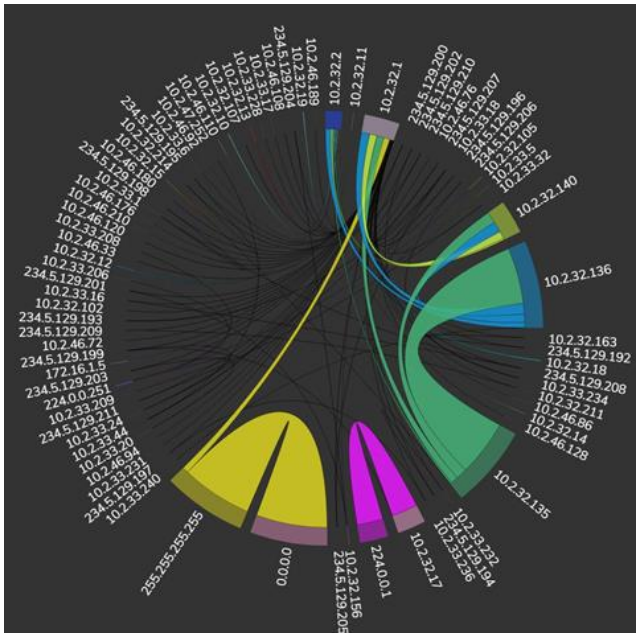


Fig. 6- Zoom-in of the FVT - NetFlow analysis tool

One of the presumably most useful filters is the one that allows operators to select the IP addresses to be shown on the diagram. The filter allows the selection of internal IPs only and it can also group the external ones by country. Fig. 7 shows how the filter can be applied by simply checking the relevant checkboxes in a tree-structured format.

After selecting some specific IPs, in this example the IPs of the experiment nodes, the diagram gets updated to include those and their relevant NetFlows. Fig. 8 shows the NetFlows of one specific IP. The node 10.2.32.136 has been selected so the operator can see the inbound and outbound traffic for this specific node at the specified date and time interval.

Fig. 9 shows two instances of the diagram where the results of such filtering are highlighted: traffic going from node 10.3.32.135 to 10.2.32.136 and 10.3.32.1 to 10.2.32.140 respectively. The whole diagram only includes traffic between the selected internal IPs, namely the experiment nodes: 10.2.32.135, 10.2.32.136, 10.2.32.140 and other internal nodes:10.2.32.1, 10.2.32.2, 10.2.32.15 and 10.2.33.240.

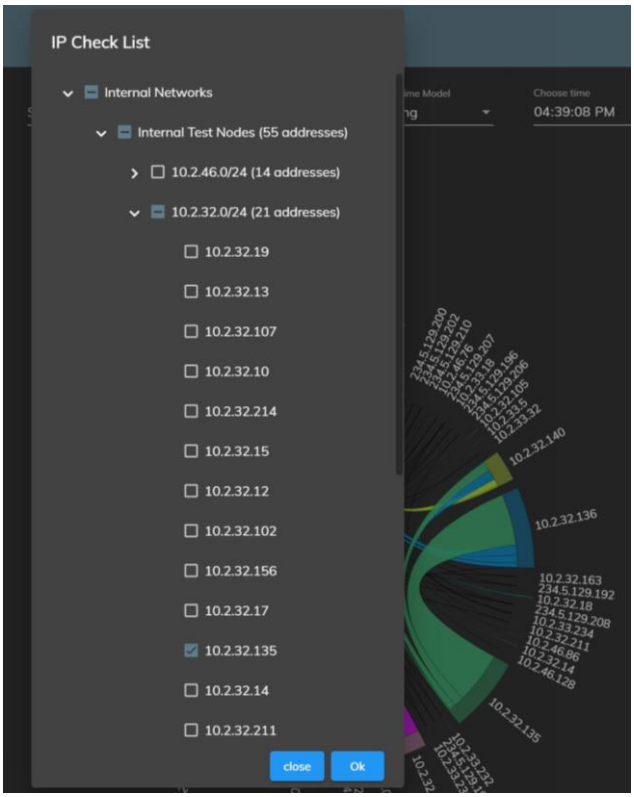


Fig. 7- FVT - IP filtering

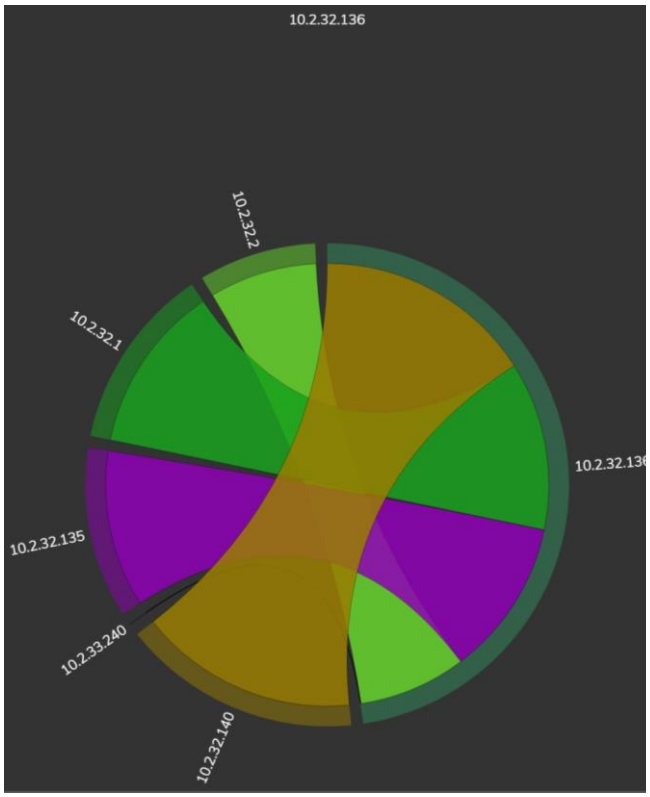
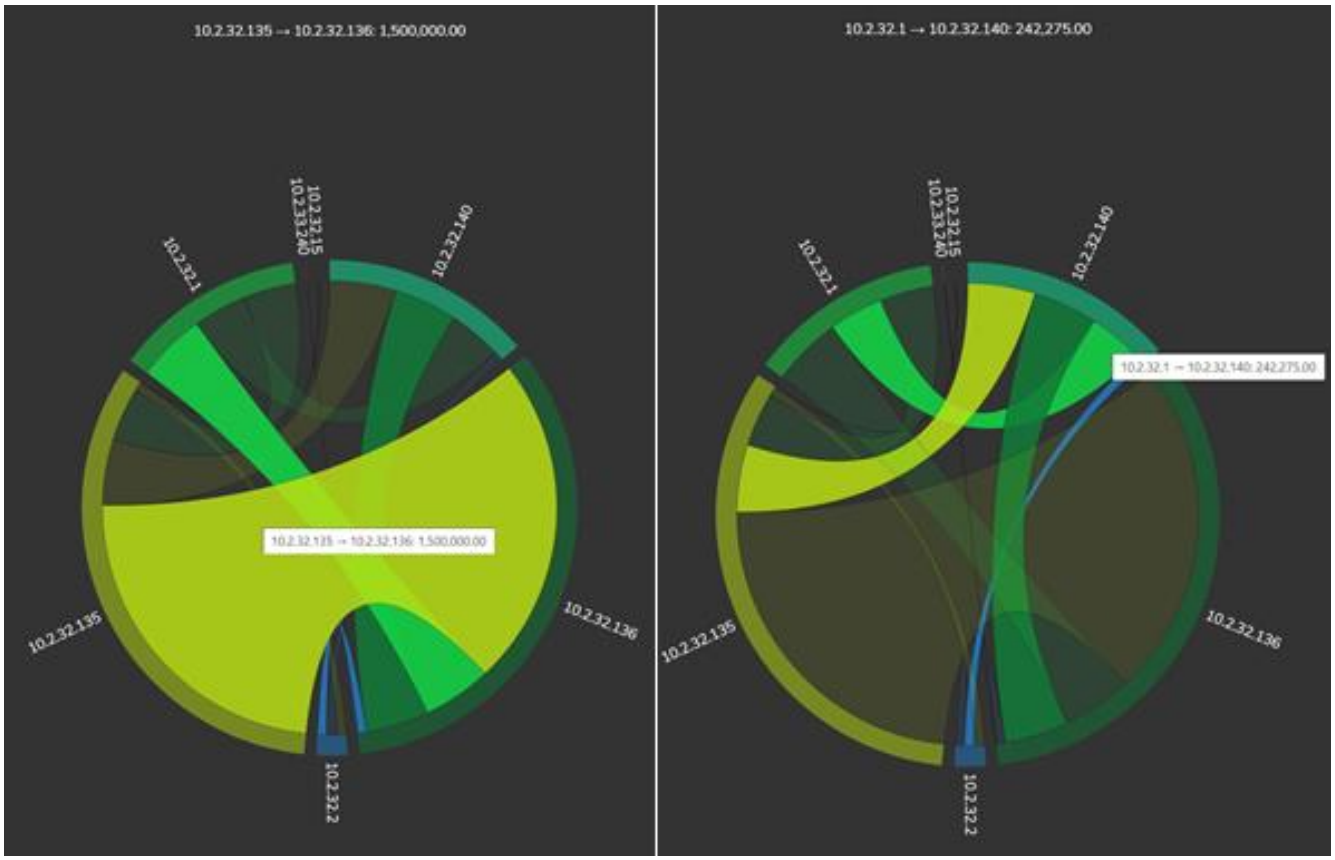


Fig. 8- NetFlows of one IP



Fig, 9- FVT - NetFlows between internal IPs

The web application supporting all these visualizations and functionalities is built using the Angular framework [7]. As already mentioned, the application is hosted by the machine serving as the FVT agent and consumes data served in a proper format by the NetAn internal API. In the following paragraphs, a set of technical results and the impact on the technological offering of the tool are presented.

In terms of the ratio between the time reserved vs the time actually used for each deployed resource, resources were approximately used 20% of the time that they were reserved. This is due to changes to our software that we wanted to test on the same setup. To do this, we took advantage of the duration extension capability. Extending an existing experiment is a very simple action so we found it much easier than setting up a new experiment from scratch. However, most of the time we checked the availability of the resources before actually extending the experiment duration with the aim to be 'fair' towards other potential experimenters. Reality proved the resources were always at a high rate of availability so we proceeded with the extension most of the time. Therefore, a quite high rate of idle time was the result of this approach.

The use of the JFed tool was as expected at almost all times. We have been using it to allocate 3-15 physical nodes and one address pool. In our assessment, the following positive aspects were identified:

- easy drag & drop feature
- useful OS image selection
- easy one-click Secure Shell (SSH) access to nodes
- easy way to extend experiment duration
- recovery option always worked as expected
- bug reporting mechanism was simple and responses were of high speed and valuable information.

On the other hand, some of the issues we faced are described hereinafter:

- Allocating nodes did not work on some times during the whole testing period although the availability was presented at a healthy (green) level.
- SSH access was not possible via putty. This was eventually due to the putty itself and the way sessions are handled by it. The support provided helped to identify the issue and the means to resolve it.
- Connecting the nodes in the design area can be sometimes tricky, click and drag of the connection line is sometimes misleading.

4.2 Technical results and lessons learned

To test the tool, we followed a test-case execution approach which included several predefined steps to be followed using the tool functionalities. Such a test case to check the usability of the tool included the following steps:

- Bring up the NetFlow analysis tool and select a one (1)-hour interval on the previous day and time in relation to the current time.
- Check the internal traffic and find the node with the highest traffic load for the specified hour.
- Determine the external IP that received most of the traffic by the internal network for the specified hour.

Following the steps above, we measured the time needed to reach the end results, the efficiency of the provided UI controls and the user perception on the usability of the tool.

One of the first technical results during the experiment execution was derived from this test case. It was about the functionality of the NetAn visualization and the emerged need to add new features that increase the visualization usefulness. In the original version of the tool that we started the experiments with, there was no provision for IP range filtering, meaning that one could not select to see traffic for internal only or external IP addresses. This resulted in a difficulty to inspect the traffic of an internal network like the one we have in the experiment. Therefore, we enhanced the IP filtering functionality to be able to select internal/external IPs using a tree-structured selection widget. This control has undergone several updates to include internal/external network distinction, internal addresses grouped in IP blocks using the same network mask and finally counts of addresses

[7] https://angular.io/

belonging in every group (seen in parenthesis beside every tree leaf).

Additionally, we introduced a grouping filter which has similar options (internal, external networks) but it differs in that it groups together IP addresses as a single edge. Therefore, the operators can have one chord including, for example, all the traffic between the internal network and the external addresses. This provides additional analysis and exploratory options to operators while they deal with a big number of NetFlows that can otherwise produce a hard to read diagram.

Another technical result was the need for a configuration file that would enable adaptation of the visualizations according to the underlying network topology. Such a configuration file would allow an easy way to define the network nodes of the monitored network so as to have the relevant filters (e.g., IP filtering and grouping) behave according to the given topology. The necessary file and the respective code changes to load properties in the visualization elements were implemented and tested as evident from the screenshots presented in the previous subsections.

Finally, the last technical result extracted from the experimentation was the need to better structure the installation process. This includes better documentation of the required steps as well as the definition of the alternative options with regard to the OS of the machine that the installation is taking place on. The latter applies mostly on the utilities that have to be installed on the existing network nodes for which we seek a minimum footprint for our components. For the FVT agent, the installation and initial configuration of the various required components (NetFlow collector, NetAn agent, web app) were reviewed and fine-tuned while executing the experiments. The final goal is to eventually package the required components as a docker container which will highly increase the integrability of the tool and decrease the time needed to have it up and running on a production environment.

The technical results and relevant actions described in the previous paragraphs offered us the opportunity to gain some knowledge summarised in the following lessons:

- The installation and the configuration of the tool must be a streamlined process adaptable to multiple variations of network topologies (e.g. configuration files have been created and ‘dockerization’ is forthcoming).
- The tool must maintain a modular architecture that allows extensions to be implemented in a fast and robust manner, thus enhancing its expandability and constant improvement of usability and effectiveness (e.g. additional visual elements in filters and supporting internal API functions were implemented to satisfy emerging needs).

Scalability in terms of data and network sizes will reveal unforeseen problems that will have to be accounted for at a fast-paced rate (e.g. a huge number of NetFlows led to the introduction of an IP grouping feature).

5. CONCLUSION

The major reason to consider F4F+ for the experiment was the compatibility of infrastructure with what we wanted to achieve as well as the ease of access to it (e.g. graphical setup of a network, SSH access to the machines). Thanks to the experiment conducted within the F4F+ testbed, we could optimize the internal processes, discover technical needs, and refine the FVT value proposition before entering the market. As a next step, we intend to experiment with more nodes and complex network topologies, simulating loads of larger companies. Our intention is to achieve the monitoring of 100 or more nodes reproducing more complex security scenarios that can potentially generate vast amounts of data, events and logs to be analysed, e.g. efficient analysis of network communications to identify a specific node’s activity.

Moreover, the experiment results helped us improve the formulation of our business plan by providing valuable information towards:

- Acceleration of the time-to-market process via highlighting the resources and timeline of a production deployment of our tool. We were able to better define an appropriate schedule, human resources needed and managerial processes that will be needed before deploying at a customer’s infrastructure.

- Better definition of target customer groups via assessment of OPEX and CAPEX needs for the supporting of various realistic network topologies of potential clients. The experiment setups provided useful insights on what the needs of customer groups might be and what kind of investments would have to be undertaken by the company to support them. Therefore, our strategic planning will take into account this data before forming the final business plan and market proposition.
- Limited business risk via early identification of potential risks and existing weaknesses that can be handled before getting to the market and therefore ensuring a robust service for early adopters that will foster acceptance by greater groups of users.

Tools like FVT enable investigators to make sense of massive amounts of data, analyse security event data in real time for internal and external threat management, which necessitates collecting, storing, analysing and reporting on log data for forensics and regulatory purposes. Visualization plays a crucial role in the easy and effective identification of crucial information regarding security, forensics (incident response), big data, business productivity and human perception.

REFERENCES

[1] J.-P. A. Yaacoub, O. Salman, H. N. Noura, N. Kaaniche, A. Chehab, and M. Malli, "Cyber-physical systems security: Limitations, issues and future trends," *Microprocess Microsyst*, vol. 77, p. 103201, Sep. 2020, doi: 10.1016/j.micpro.2020.103201.

[2] M. I. Alghamdi, *Digital Forensics in Cyber Security—Recent Trends, Threats, and Opportunities*. IntechOpen, 2021. doi: 10.5772/intechopen.94452.

[3] J. Jang-Jaccard and S. Nepal, "A survey of emerging threats in cybersecurity," *Journal of Computer and System Sciences*, vol. 80, no. 5, Aug. 2014, doi: 10.1016/j.jcss.2014.02.005.

[4] M. Bada and J. R. C. Nurse, "Developing cybersecurity education and awareness programmes for Small and medium-sized enterprises (SMEs)," *ICS*, vol. 27, no. 3, pp. 393–410, Jul. 2019, doi: 10.1108/ICS-07-2018-0080.

[5] A. Fukami, R. Stoykova, and Z. Geradts, "A new model for forensic data extraction from encrypted mobile devices," *Forensic Science International: Digital Investigation*, vol. 38, Sep. 2021, doi: 10.1016/j.fsidi.2021.301169.

[6] J. P. van Zandwijk and A. Boztas, "The iPhone Health App from a forensic perspective: can steps and distances registered during walking and running be used as digital evidence?" *Digital Investigation*, vol. 28, pp. S126–S133, Apr. 2019, doi: 10.1016/j.diin.2019.01.021.

[7] M. Conti, A. Dehghantanha, K. Franke, and S. Watson, "Internet of Things security and forensics: Challenges and opportunities," *Future Generation Computer Systems*, vol. 78, pp. 544–546, Jan. 2018, doi: 10.1016/j.future.2017.07.060.

[8] R. Altschaffel, "Computer Forensics in Cyber-Physical Systems - Applying Existing Forensic Knowledge and Procedures from Classical IT to Automation and Automotive," University of Magdeburg, 2020.

[9] P. Reedy, "Interpol review of digital evidence 2016 - 2019," *Forensic Sci Int Synerg*, vol. 2, Mar. 2020, doi: 10.1016/j.fsisyn.2020.01.015.

AUTHORS

Leonidas Kallipolitis holds a BSc in informatics and telecommunications and an MSc in advanced information systems from the National and Kapodistrian University of Athens, Greece. He has been working as a software engineer since 2008. He has great experience working as a technical coordinator of research projects and contributing to the analysis, design and implementation of complex IT systems. He is particularly involved in web applications development and system integration in the domains of cybersecurity, digital forensics and big data analytics. His research interests include novel data visualization techniques and the latest cybersecurity concepts.

Panagiotis Katrakazas (Ph.D.) has been working as a research area manager in Zelus P.C. since November 2020. He holds a diploma in electrical and computer engineering and a Ph.D. in biomedical engineering (National Technical University of Athens, NTUA). He is also appointed IT Associate in the Department of Transportation Planning and Engineering of the School of Civil Engineering (NTUA). His scientific specialization and research interests include acoustics, noise control, data mining and pattern analysis, system theory and analysis. He has authored or co-authored 11 articles in peer-reviewed journals, 23 articles in peer-reviewed conference and two chapters in peer-reviewed books.

Ilias Spais (Ph.D.) received a diploma in electrical and computer engineering from the University of Patras in 2000, and a PhD degree in analysis, design and development of processes, systems and computer engineering from the National Technical University of Athens (NTUA) in 2006. He has been involved in several research projects in the context of the ICT framework as innovation manager, research associate and senior developer for more than 15 years. His research interests include, adaptive big data visualization systems, edge computing applications, natural language processing algorithms, speech recognition and synthesis, distributed SOA-based systems and multimedia e-learning platforms.