Time Series Data Mining for Network Service Dependency Analysis

Mona Lange and Ralf Möller

Universität zu Lübeck, Germany, email: {lange,moeller}@ifis.uni-luebeck.de

Abstract. In data-communication networks, network reliability is of great concern to both network operators and customers. To provide network reliability it is fundamentally important to know the ongoing tasks in a network. A particular task may depend on multiple network services, spanning many network devices. Unfortunately, dependency details are often not documented and are difficult to discover by relying on human expert knowledge. In monitored networks huge amounts of data are available and by applying data mining techniques, we are able to extract information of ongoing network activities. Hence, we aim to automatically learn network dependencies by analyzing network traffic and derive ongoing tasks in data-communication networks. To automatically learn network dependencies, we propose a methodology based on the normalized form of cross correlation, which is a well-established methodology for detecting similar signals in feature matching applications.

1 Introduction

For deriving how susceptible a network is to software vulnerabilities or attacks, it is essential to understand how ongoing network activities could potentially be affected. A network is built with a higher purpose or mission in mind and this mission leads to interactions of network devices and applications causing network dependencies. A monitored infrastructure's missions can be derived through human labor, however missions are subject to frequent change and often knowledge of how an activity links to network devices and applications is not available. So we are challenged to automatically derive these missions as network activity patterns through network service dependency discovery.

In the context of this work, we introduce a novel framework for Mission Oriented Network Analysis (MONA). To motivate our approach, in Section 2 we provide a background to other network dependency assessment methodologies and illustrate their limitations. We introduce a network model in Section 3 and in Section 4 we uncover network dependencies based on network traffic. In a monitored network large amount of unlabeled data, in form of network traffic, are available for knowledge discovery. This allows us to develop a deeper understanding of network activities. For uncovering network dependencies, we propose a methodology based on normalized cross correlation, which is a well-established methodology for detecting similar signals in feature matching applications. In Section 5, we evaluate MONA based on network traffic traces provided by an energy distribution network.

2 Related Work

Network activities in a data-communication network follow from a network having a higher task. Others refer to a network having a higher task as a mission. The concept of missions is sometimes also referred to as mission-centricity in cyber security.

Multiple distinct mission-centric approaches to cyber security have been proposed [8, 4, 10, 1, 11]. Albanese et al. [8] use ontologies for integrating available data into a common model. Barreto et al. introduced an impact assessment methodology [4] incorporating vulnerability descriptions [12] and other numerical scores acquired through a BPMN model via human input. Another mission-centric approach is introduced by Jakobson [10], who presents an interdependency model representing an infrastructures operational capacity. Albanese et al. [1] present an interdependency model focusing on cost minimization based on information acquired via human input. Another mission-centric approach is the Cyber Attack Modeling and Impact Assessment Framework [11], which supports computational analysis of a monitored infrastructure and allows impact assessment. An example for a mission-centric approach is the framework for cyber attack modeling and impact assessment [11]. They rely on a mission model for generating attack graphs. All these models do not focus on how to acquire the information used to derive interdependency models. Network dependency analysis allows automatically deriving interdependency models based on network traffic. Recent efforts have explored network-based approaches that treat each host as a black box and passively analyze the network traffic between them. For network administrators that are planning to upgrade or reorganize existing applications a dependency discovery approach named Leslie Graph [2] was designed. The approach aims at identifying complex dependencies between network services and components that may potentially be affected and prevent unexpected consequences. NSDMiner [13] addresses the same problem of network service dependency for network stability and automatic manageability. Sherlock [3] is another approach, which learns an inference graph of network service dependency based on co-occurrence within network traffic. A well-known approach is called Orion [6], which was developed to use spike detection analysis in the delay distribution of flow pairs to infer dependencies. All previously mentioned approach require large amounts of network traffic compared to MONA, and where developed to minimize false negative network service dependencies. To our knowledge, MONA is the first stream-based network service dependency analyzer.

3 IT Network Model

Modeling an IT network requires a basic understanding [7] of the Open Systems Interconnection (OSI) model. For understanding network connectivity, the following layers of the OSI model are of particular interest: data link layer, network layer, transport layer and application layer. We define a network device as a physical device on the network. The data link layer physically links network devices using MAC addresses to identify devices. However, the data link layer only provides point-to-point connectivity. For enabling network connectivity beyond a point-to-point communication, a network layer protocol such as the Internet Protocol (IP) is required. IP addresses are used to identify

source and destination of an end-to-end connection. In other words, MAC addresses allow a point-to-point connection, while IP addresses provide an end-to-end connection. Therefore, switches and routers are used to forward packets, i.e. they act as intermediate hosts. Data-communication networks are built with a common higher purpose. This leads to reoccurring interactions between distinct network devices and services, which we call network activity patterns. An example for such a network activity pattern is given in the following example.

Definition 1 (Network Device). *Let MAC and IP be non empty sets of MAC and IP addresses, respectively ($MAC \cap IP = \emptyset$), then*

$$D^{CY} \subseteq \mathcal{P}(MAC) \setminus \{\emptyset\} \times \mathcal{P}(IP) \tag{1}$$

is the set of network devices.

This allows a device to be assigned multiple MAC addresses and IP addresses. Being able to assign multiple MAC addresses to a network device is needed as routers and switches supply multiple point-to-point endpoints. However, switches do not necessarily need to have IP addresses as they work on the data link layer. From this it follows that they are not visible on the network layer. A network captures devices that are communication endpoints and additional intermediate devices, over which endpoints communicate. Network devices that are endpoints can host network services.

Definition 2 (Network Service). *Let S be a set of network services such that a network service $s_i^j \in S$ is hosted by a network device $d_j \in D^{CY}$. Additionally, the network service is associated by a transport protocol $\Psi = \{TCP, UDP\}$ and a port. This allows us to define a relation $SERV$, which links a network device, a transport protocol and a port number to a network service by the follow equation*

$$SERV : D^{CY} \times \Psi \times \mathbb{N} \to S. \tag{2}$$

To derive all network services hosted by a device d_j, we define a relationship $HOSTS(d_j)$, which returns all network services hosted by d_j. In order to derive the device a service s is hosted by, we write $HOSTS^{-1}(s)$, and associate service s_i with device d_j by writing s_i^j. Given a service $s_i^j \in S$, the corresponding device d_j can be derived by

$$d_j = HOSTS^{-1}(s_i^j). \tag{3}$$

Additionally, for a given IP-address and port, we are able to derive the corresponding network device by

$$DEV : \mathcal{P}(IP) \times \mathbb{N}. \tag{4}$$

This allows us to derive all involved network services for a given IP-address and port pair by $HOSTS(DEV(sIP, sPort)) \to \mathcal{P}(S)$. Based on network traffic analysis we will detect network services and determine how they communicate in an end-to-end manner with each other. Aside from intermediate devices (e.g. routers and switches), network devices can be categorized into client and server network devices. In the following we will refer to client network devices as clients and server network devices as servers. Clients and servers are able to send requests. Servers additionally provide network services that answer these requests. Generally, the number of clients by far surpasses the number of servers.

Definition 3 (Network Packet). *The basic building block of our approach are network packets exchanged between directly dependent network services. A network packet is exchanged by a source and destination IP address srcIP and dstIP via source and destination port srcPort and dstPort. In addition, a network packet relies on a specific transport layer protocol. In the context of this paper we distinguish the transport layer protocols TCP and UDP. We define a network packet as a 6-tuple*

$$P = (sIP, sPort, dIP, dPort, \psi, t), \tag{5}$$

for source IP addresses sIP, a source ports $sPort$, destination IP addresses dIP, destination ports $dPort$, a transport protocol $\Psi = \{$UDP, TCP$\}$ and timestamps t.

Requests are often sent through a dynamically assigned port. Dynamically assigned ports are chosen from specifically assigned port ranges [14]. Ephemeral port ranges are available for private, customized or temporary purposes. Although IANA recommends ephemeral port ranges to range from $2^{15} + 2^{14}$ to 2^{16}, the range is highly dependent on the operation system. Microsoft assigns ephemeral ports starting as low as 1025 for some windows versions and a lot of Linux kernels have the ephemeral port range start at 32768. We follow the IANA recommended ephemeral port range for clustering purposes.

Definition 4 (Cluster Network Service). *Let S be a set of services that are hosted by device d_j. All network services communicating through a dynamically assigned port, are grouped by*

$$s^j_* \in S, \tag{6}$$

whereas $*$ represents a dynamically assigned port and j represents the device a network service is hosted on. Known network services have to be linked to ports statically, such that other network services can routinely communicate requests with them. It should also be noted that multiple statically assigned ports could be assigned to the same application.

Based on Equation 5, we conduct a network dependency analysis based on packet headers (e.g. IP, UDP and TCP) and timing data in network traffic. Hence, our approach operates on network flows. To identify network flow boundaries, we look into the definition of TCP and UDP flows. A TCP flow starts with a 3-way handshake (SYN, SYN-ACK, ACK) between a client and a server and terminates with a 4-way handshake (FIN, ACK, FIN, ACK) or RST packet exchange. If network services communicate frequently, they may forgo the cost of repetitive TCP handshakes by using KEEPALIVE messages to maintain a connection in idle periods. In comparison the notion of UDP flows is vague, since UDP is a stateless protocol. This is due to the protocol not having well-defined boundaries for the start and end of a conversation between server and client. In the context of this work, we consider a stream of consecutive UDP packets between server and client as a UDP flow, if the time difference between to consecutive packets is below a predefined threshold. In our analysis we exclude all network packet that are necessary for establishing a communication between server and client. So given that additional data is exchanged between network service s^j_i and s^l_k,

we term these end-to-end interactions between network services as direct dependencies. The direct dependency $SDEP$ between network services s_i^j and s_k^l is denoted as $SDEP = \{SDEP^{rq} \bigcup SDEP^{rsp}\}$. We distinguish requests and responses exchanged between network services based on Equation 6. If a network services uses an ephemeral port to send a network packet to a network service on a static port range, we assume it is a request. Thus, an exchanged request $SDEP^{rq}$ is denoted by

$$SDEP^{rq} = \{(s_*^j, s_k^l) \mid s_*^j \text{ sends a request to } s_k^l \text{ in the period under consideration,}\} \tag{7}$$

where k is in the statically assigned port range. Conversely, this means that a network service using its static port range to answer a network service on an ephemeral port is defined as a response. An exchanged response $SDEP^{rsp}$ is written as

$$SDEP^{rsp} = \{(s_k^l, s_*^j) \mid s_k^l, \text{ sends a response to } s_*^j \text{ in the period under consideration.}\} \tag{8}$$

4 Network Service Dependency Discovery

A data-communication network consists of network devices, which interact due to applications via network services. For example an application "email" uses network services $IMAP$ and $POP3$ to access email messages from a remote network device (i.e. host). From this it follows that an application can rely on multiple network services to fulfill a common goal, which is also referred to as mission. Additionally, we note network activities such as accessing email messages lead to network packets being exchanged by directly dependent network service. The purpose of network service dependency discovery is to abstract network packets in order to detect reoccurring communication pattern. Reoccurring communication patterns indicate that the involved network services are dependent. In order to detect reoccurring communication patterns, we first abstract monitored network traffic into communication histograms. This analysis is done online based on continously captured network traffic.

4.1 Communication Histograms

Let us suppose that we are mirroring network traffic from an initial time point t_{min} to a time point t_{max} within an IT network. We are observing network packets $p \in P$, which are defined as a 6-tuple according to Equation 5. For communicating network services, we build communication histogram with a bin size Δ_t. In the context of work we set Δ_t to 1 second. The number of histogram bins is given by

$$bins = \lfloor \frac{(t_{max} - t_{min})}{\Delta_t} \rfloor, \tag{9}$$

assuming we want to build a communication histogram for network traffic mirrored from time point t_{min} to t_{max} with a bin size Δ_t.

Given that we are monitoring a set of S network services then the data structure for all communication histograms is defined by

$$H : S \times S \times \Psi \rightarrow (\{0, \cdots, bins - 1\} \rightarrow \mathbb{N}_0), \tag{10}$$

where the communication histogram bins $\{0, \cdots, bins-1\}$ are mapped to $\mathbb{N}_0$. Now for every network packet exchanged between directly dependent network services, assuming it was received during the considered time period, the corresponding bin $I(H(s, s', \psi)$ in the communication histogram is incremented. The corresponding bin in the communication histogram is determined by

$$(t_{min} - t) \mod bins, \tag{11}$$

assuming that the network packet p contains the time stamp t.

4.2 Indirect Dependencies

Network services operate on distributed sets of clients and servers and rely on supporting network services, such as Kerberos, Domain Name System (DNS), and Active Directory. To fulfill a network's mission, network services need to interact. Since, engineers use the divide-and-conquer approach to implement a new task, they are able to reuse network services and do not need to re-implement complex customized ones. This leads to multiple network services interacting for a common high-level task. For the purpose of detecting indirect dependencies, we analyze the communication histograms of directly dependent network services in order to derive re-occurring communication patterns. Detecting re-occurring communication patterns requires clustering direct dependencies. Similarly to previous work, we distinguish two different types of remote-remote dependencies and local-remote dependencies [6]. A local-remote (LR) dependency is an indirect dependency, where a system must issue a request to a remote system in order to complete an outstanding request issued to a local service. A remote-remote (RR) dependency is one in which a system must first contact one host before issuing a request to the desired host.

Definition 5 (Candidates for an Indirect Dependency). *Given a direct dependency $\delta(s_i^j, s_k^l)$, all network services hosted by*

$$HOSTS^{-1}(s_k^l) = d_l \text{ or } HOSTS^{-1}(s_i^j) = d_j \tag{12}$$

are candidates for an indirect dependency.

The communication histograms contain the communication pattern of all involved directly dependent network services.

4.3 Measuring Communication Histogram Similarity

Suppose we have two communication histograms r and $s \in H$, which are candidates for being indirectly dependent. The two communication histograms are time series $r = (r_1, r_2 \cdots, r_{bin})$, consist of bin samples.

In statistics, the Pearson product-moment correlation coefficient the linear correlation σ between two variables X and Y by

$$\sigma = \frac{[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}, \tag{13}$$

where μ_Y and σ_Y denotes the mean and standard deviation of Y. The Pearson product-moment correlation coefficient is a measure of the linear correlation between two variables X and Y, giving a value between +1 and -1 inclusive, where

1 is total correlation,
0 is no correlation, and
-1 is total anti-correlation.

To apply the Pearson correlation coefficient as a distance measure on a time series [9], we define the Pearson Distance as

$$d_\varrho(r, s) = \frac{\frac{1}{bins} \sum_{t=0}^{bins} (r_t - \mu_r)(s_t - \mu_s)}{\sigma_r \sigma_s}, \tag{14}$$

where μ_r and σ_r are mean and standard deviation of r, such that $-1 \leq d_\varrho \geq 1$.

An indirect dependency implies that the data received by s_k^l is processed on device d_l. Then, data is sent to another network service s_m^l. Due to the processing of data on device d_l, the request might be sent to s_m^l τ_{delay} time steps later. Network latency can lead to communication patterns being shifted due to the time it takes for a network packet to be transferred. Communication patterns are stored in communication histograms and they contain the same pattern, which thus are also shifted due to network latency. Hence, the communication patterns of both direct dependencies would be similar, although shifted by τ_{delay} time steps. In pattern recognition, normalized cross correlation has been proposed to take a shift, such as τ_{delay}, into account.

To overcome the lack of a perfect alignment between two communication networks, we extend the Pearson distance, introduced in Equation 14, to normalized cross-correlation [5].

$$\varrho_{r,s}(\tau) = \frac{\frac{1}{bins} \sum_{t=0}^{bins} (r_t - \mu_r)(s_{t+\tau} - \mu_s)}{\sigma_r \sigma_s}, \tag{15}$$

The point in time τ_{delay}, where both signals are best aligned can be found by computing

$$\tau_{delay} = \text{argmax}_t \varrho_{r,s}(\tau). \tag{16}$$

If $\varrho_{r,s}(\tau) \geq \theta$, we consider both communication histograms r and s to be correlated and therefore indirectly dependent and shifted by τ_{delay}. Normalized cross-correlation is applied to all indirect dependency candidates and returns a set $ISDEP$, which is derived by applying

$$ISDEP = SDEP \bowtie SDEP \tag{17}$$

on all LR (local-remote) and RR (remote-remote) dependencies.

Algorithm 1 Building communication histograms algorithm

Input:
Network packet $\mathbf{P} = (sIP, sPort, dIP, dPort, \psi, t)$,
start time $\mathbf{t_{min}}$, stop time $\mathbf{t_{max}}$, time step Δ_t
Output: Communication histograms $\mathbf{H}$
▷ compute number of bins for communication histogram $\mathbf{H}$
bins = $(t_{max} - t_{min})$ div Δ_t ▷ all histogram vectors are initialized and filled with zeros
▷ fill histogram bins for every network packet in a network flow
for all p = $(sIP, sPort, dIP, dPort, \psi, t) \in \mathbf{P}$
and $t \geq t_{min}$ **and** $t \leq t_{max}$ **do**
tb = $(t - t_{min})$ mod bins
$\mathbf{H}(SERV(DEV(sIP), sPort),$
$SERV(DEV(dIP), dPort),$
$\psi)[tb]$++
end for
return $\mathbf{H}$

5 Experimental Evaluation

The disaster recovery site of an energy distribution network, provided an Italian water and energy distribution company, was available for network traffic analysis. Based on this network, we are able to deploy MONA for online analysis. In addition, we are collect and analyze real-life network traffic with the help of network operators. Real-life network traffic consists of network services frequently to rarely interacting and we are able to test MONA with typical communication patterns occurring in an operational environment. As this network is a real-life network, absolute knowledge of all existing and non existing network dependencies can only be assumed. Data communication networks are dependent on third party software and operators do not have complete knowledge. However, all identified indirect network service dependencies, which are shown in Figure 1, where classified as true positives.

The node named mferp2 is a communication server for multiple substations, which are identified as TTY-T[116-158]. Also, our evaluation shows, how important Eq. 1 is as mferp2 is one physical device, but is assigned two IP address from two different subnetworks. The communication server mferp2 hosts a network service, defined in Equation 2. Port numbers that identify the network service are appended to the host names. This network service belongs to an application, which sends requests (see Equation 7) to all substations in order to be updated with current measurement information. Therefore, a network service dependency joins these substations to the communication server. Mferp2 communicates via muel2 with Human Machine Interface (HMIs) msoz19 and mso22. Another HMI msoz17 wants to access information about the substations TTY-T[116-158]. For this, first muel1 is contacted, who passes the request on to muel2. As an energy distribution network is a critical infrastructure, it needs to be ensured that all communication pathways are always available. Hence, regularly back up servers and alternate communication pathways are tested, even if no information needs to be transmitted. All these network service dependencies where verified by operators as true positives.

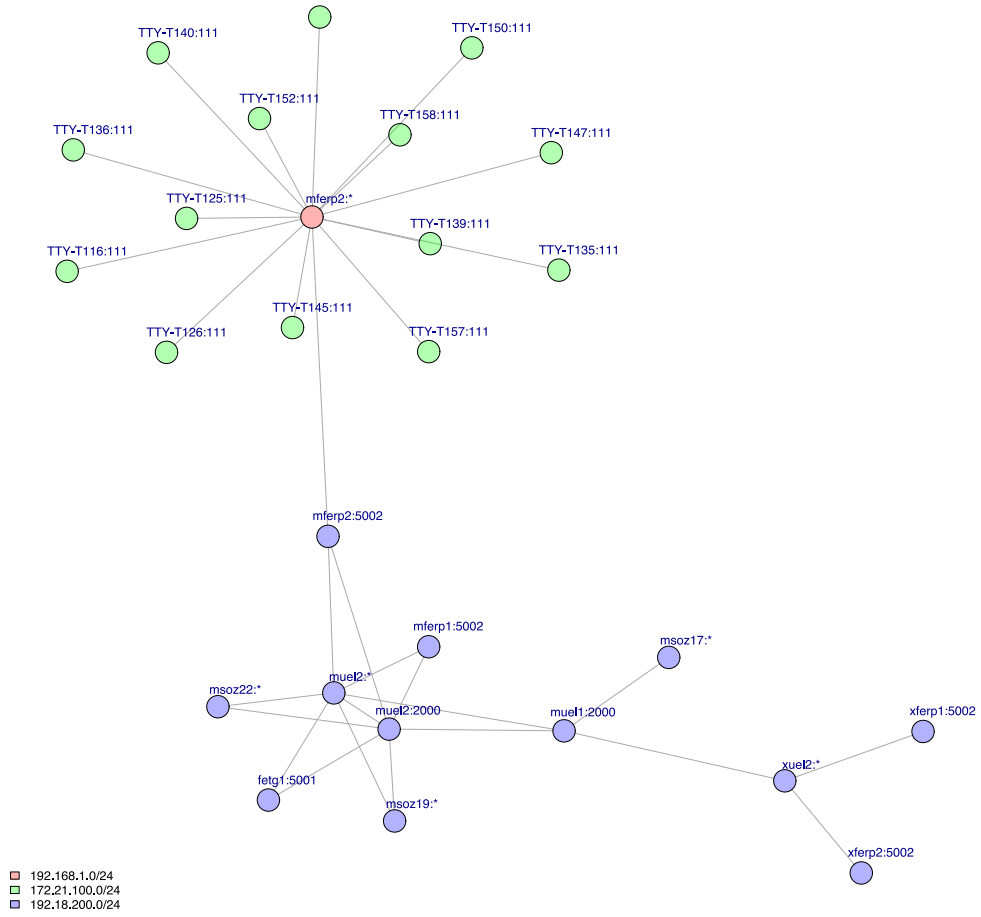

Fig. 1: Network Service Dependency Analysis in an energy distribution network.

6 Conclusion

We have demonstrated a novel network dependency detection approach. During first experiments on online network traffic from the disaster recovery site, we often found new network dependencies that had been previously forgotten by the network operators. Subsequence mining enables deriving a deeper understanding of network activities and leverages well data-communication networks with different numbers of network devices and indirect dependencies.

Acknowledgments

This work has been partially supported by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 610416 (PANOPTESEC). The opinions expressed in this paper are those of the authors and do not necessarily reflect the views of the European Commission.

References

1. Albanese, M., Jajodia, S., Jhawar, R., Piuri, V.: Reliable mission deployment in vulnerable distributed systems. In: Dependable Systems and Networks Workshop (DSN-W), 2013 43rd Annual IEEE/IFIP Conference on. pp. 1–8. IEEE (2013)
2. Bahl, P., Barham, P., Black, R., Chandra, R., Goldszmidt, M., Isaacs, R., Kandula, S., Li, L., MacCormick, J., Maltz, D.A., et al.: Discovering dependencies for network management. In: ACM SIGCOMM 5th Workshop on Hot Topics in Networks (Hotnets-V). pp. 97–102. ACM (2006)
3. Bahl, P., Chandra, R., Greenberg, A., Kandula, S., Maltz, D.A., Zhang, M.: Towards highly reliable enterprise network services via inference of multi-level dependencies. In: ACM SIGCOMM Computer Communication Review. vol. 37, pp. 13–24. ACM (2007)
4. de Barros Barreto, A., Costa, P.C.G., Yano, E.T.: A semantic approach to evaluate the impact of cyber actions on the physical domain (2012)
5. Briechle, K., Hanebeck, U.D.: Template matching using fast normalized cross correlation. In: Aerospace/Defense Sensing, Simulation, and Controls. pp. 95–102. International Society for Optics and Photonics (2001)
6. Chen, X., Zhang, M., Mao, Z.M., Bahl, P.: Automating network application dependency discovery: Experiences, limitations, and new solutions. In: USENIX Symposium on Operating Systems Design and Implementation (OSDI). vol. 8, pp. 117–130 (2008)
7. Edwards, J., Bramante, R.: Networking self-teaching guide: OSI, TCP/IP, LANs, MANs, WANs, implementation, management, and maintenance. John Wiley & Sons (2015)
8. Goodall, J.R., D'Amico, A., Kopylec, J.K.: Camus: automatically mapping cyber assets to missions and users. In: Military Communications Conference (MILCOM). pp. 1–7. IEEE (2009)
9. Höppner, F., Klawonn, F.: Compensation of translational displacement in time series clustering using cross correlation. In: Advances in Intelligent Data Analysis VIII, pp. 71–82. Springer (2009)
10. Jakobson, G.: Mission cyber security situation assessment using impact dependency graphs. In: Information Fusion (FUSION). pp. 1–8 (2011)
11. Kotenko, I., Chechulin, A.: A cyber attack modeling and impact assessment framework. In: Cyber Conflict (CyCon), 2013 5th International Conference on. pp. 1–24 (June 2013)
12. MITRE: Common vulnerabilities and exposures. https://cve.mitre.org/ (2000)
13. Natarajan, A., Ning, P., Liu, Y., Jajodia, S., Hutchinson, S.E.: NSDMiner: Automated discovery of network service dependencies. In: IEEE International Conference on Computer Communications (IEEE INFOCOM 2012). IEEE (2012)
14. Touch, J., Kojo, M., Lear, E., Mankin, A., Ono, K., Stiemerling, M., Eggert, L.: Service name and transport protocol port number registry. The Internet Assigned Numbers Authority (IANA) (2013)