Using a Deep Understanding of Network Activities for Network Vulnerability Assessment

Mona Lange
Universität zu Lübeck
Germany
lange@ifis.uni-luebeck.de

Felix Kuhr
Hamburg University of Technology
Germany
felix.kuhr@tuhh.de

Ralf Möller
Universität zu Lübeck
Germany
moeller@ifis.uni-luebeck.de

ABSTRACT

In data-communication networks, network reliability is of great concern to both network operators and customers. Therefore, network operators want to determine what services could be affected by software vulnerabilities being exploited that are present within their data-communication network. To determine what services could be affected by a software vulnerability being exploited, 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. From a data mining perspective, we are interested to test the potential of applying data mining techniques to real-life applications.

CCS Concepts

•**Security and privacy** → **Information flow control;** *Vulnerability management;* •**Networks** → Network reliability;

Keywords

Network Dependency Analysis, Network Vulnerability Assessment, Information Flow

1. INTRODUCTION

Over the last few years, approximately 2500 software vulnerabilities were discovered every year [25]. The United States intelligence community has identified malicious actors exploiting cyberspace as a top national security threat [6]. Furthermore, IBM's 2015 cyber security intelligence index reveals that approximately half of all cyber attacks originate from within a company's own network [7]. Hence, network devices that are not connected to the internet also have to be considered as potential entry points for cyber attacks. Due to the large number of software vulnerabilities and the security threat they impose, understanding their impact on a monitored network has become an important objective. So network administrators are faced with the challenge of assessing the security impact of vulnerabilities on the network in order to choose appropriate mitigation actions.

For deriving how susceptible a network is to attackers exploiting software vulnerabilities, it is essential to understand what ongoing network activities could potentially be affected by a cyber attack. 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 mission 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. This is why an automatic network service dependency methodology called Mission Oriented Network Analysis (MONA) [18] was introduced to derive these missions as network activity patterns. MONA was compared to three state of the art network service dependency discovery methodologies: NSDMiner [21], Sherlock [4] and Orion [5]. NSDMiner addresses the same problem of network service dependency for network stability and automatic manageability. Sherlock 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, which was developed to use spike detection analysis in the delay distribution of flow pairs to infer dependencies. MONA was compared via F-measures to all these state of the art methodologies and was shown to have a better performance. Current network vulnerability approaches [19] focus on identifying critical nodes in a network without focusing on the impact of software vulnerabilities. Even though, software vulnerabilities can be remotely exploitable and sometimes even exploits are readily available online, network vulnerability assessment currently does not take currently present known vulnerabilities into account.

Developing a deeper understanding of network activities allows network vulnerability assessment to analyze what network services would be potentially be affected by a software vulnerability that was detected in a monitored network. Knowing what network activities would be affected by a software vulnerability being exploited, supports network operators in developing a deeper understanding on how their network is affected by software vulnerabilities.

Example for a network activity.

An example for a network activity pattern in an IT network is given in Figure 1. Every node in Figure 1 represents a network device. Client network devices are represented in blue and server network devices are represented as rose nodes. A majority of communication protocols consist of request and response pairs. Let us

PRAISE '16 Brussels, Belgium
© 2016 ACM. ISBN 123-4567-24-567/08/06. . . $15.00
DOI: 10.1145/2970030.2970043

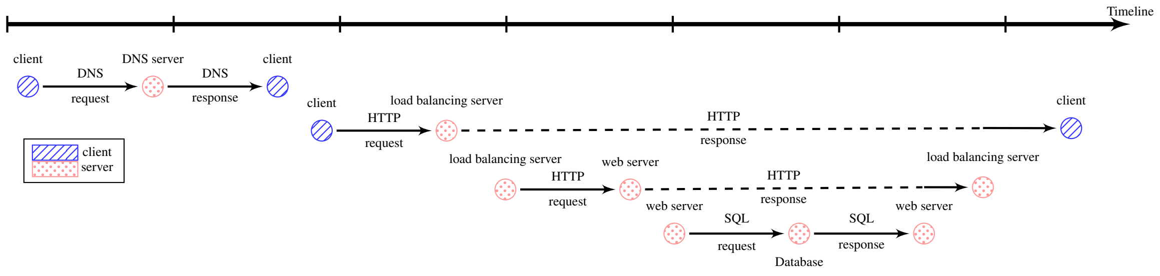

Figure 1: Example for network activities.

assume that a client wants to access a specific web server. Achieving this might require a DNS lookup. Assuming that IP-addresses are returned, a load balancing server receives a HTTP request. The load balancing server passes on the HTTP request a web server. The requested information is not available locally, but it is stored in an external Database. So the web server sends an SQL request to the database. The database sends the information back to the web server. The web server in its turn sends an HTTP response to the load balancing server, which forwards the HTTP response to the client that initiated the sequence of tasks. Per network interface available on a network device, the network device can be connected to another subnetwork. Hence, such a sequence of tasks can require multiple network device and subnetworks to be operational. The purpose of network service dependency discovery is to capture these interactions and how they link to network devices and applications.

2. RELATED WORK

To motivate our approach, we provide a background to other network vulnerability assessment methodologies and illustrate their limitations. 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.

Mission Impact Modeling.

Multiple distinct mission-centric approaches to cyber security have been proposed [8, 10, 13, 15, 24, 20]. An example for a mission-centric approach is the framework for cyber attack modeling and impact assessment [15]. They rely on a mission model for generating attack graphs. We understand a mission as network activities with a common purpose, as illustrated in Figure 1. So our understanding of what constitutes a mission corresponds to Barreto's [8]. To know how software vulnerabilities affect a mission, we need to understand network services dependencies. We say one network service depends on the other if the former requires the latter to operate properly. Automated discovery of network services dependencies from network traffic is discussed in the next paragraph.

Network Service Dependency Discovery.

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 [3] was designed. The approach aims at identifying complex dependencies between network services and components that may potentially be affected and prevent unexpected consequences. NSDMiner [21] addresses the same problem of network service dependency for network stability and automatic manageability. Sherlock [4] 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 [5], which was developed to use spike detection analysis in the delay distribution of flow pairs to infer dependencies. All previously mentioned approaches are based on analyzing network traffic and do not require additional software to be deployed on network devices. Orion, NSDMiner, Sherlock are going to be used later for comparison with MONA.

3. NETWORK VULNERABILITY ASSESSMENT

Network vulnerability analysis helps network operators verify, whether a software vulnerability discovered by vulnerability scanners within a monitored network might endanger ongoing network activities. Network vulnerability assessment consists of two parts: detecting present software vulnerabilities in a monitored network and analyzing a network's sensitivity to particular software vulnerabilities. According to the ISO 27005 standard, a vulnerability is "A weakness of an asset or group of assets that can be exploited by one or more threats" [12]. Whereas an asset is defined by ISO13355 ISO/IEC TR13355-1 [11] as being "anything that can have value to the organization, its business operations and their continuity, including information resources that support the organization's mission". The corporation MITRE, a non-profit organization, since 1999 defines Common Vulnerabilities and Exposures (CVE) identifiers for software vulnerabilities.

The non-profit organization MITRE since 1999 defines common Vulnerabilities and Exposures (CVE) identifiers for software vulnerabilities [17]. Vulnerability databases such as the Open Source Vulnerability Database [16] or the National Vulnerability Database [1] (NVD) can be used to search for software vulnerabilities.

3.1 Vulnerability Model

Our vulnerability model is inspired by MulVAL's [23]. MulVAL's vulnerability model relies on the ICAT data model, which was a database describing vulnerability effects. The National Institute of Standards and Technology has integrated ICAT into the NVD and relaunched ICAT vulnerability Web site and relaunched it as the NVD.

Common Vulnerability Scoring System (CVSS) values are contained in NVD are used to rank a software vulnerabilities effect on a monitored data-communication network. Let us assume a set of all known vulnerabilities N. Then, based on the NVD, which contains a CVSS base score vector in the format "(AV/AC/Au/C/I/A)", we are able to model a vulnerability $\nu_{id} \in N$ as $\nu_{id} = (AV, AC, Au, C, I, A)$

for

- AV: Access Vector (Local, Adjacent Network or Network),
- AC: Access Complexity (Low, Medium or High),
- Au: Authentication Required (Multiple, Single or None),
- C: Confidentiality (None, Partial, or Complete),
- I: Integrity (None, Partial, or Complete), and
- A: Availability (None, Partial, or Complete).

The Access Vector AV, Access Complexity AC, and Authentication metrics Au capture how the vulnerability is accessed and whether or not extra conditions are required to exploit it. The more remote to the intended target an attacker can launch an exploit, the higher the AV score. Often an AV score of "Network" is referred to as remotely exploitable. Access Complexity AC captures how easy it is to launch an exploit of the described vulnerability. How often an attacker has to authenticate to a targeted network device is captured by the Authentication score Au. Aside from descriptive values, base scores are also associated with numerical values. These three base scores AV, AC and Au can be combined into an common exportability score $\xi_{access}(\nu_{id})$ capturing the ease of exploiting a vulnerability

$$\xi_{access} : \nu_{id} \rightarrow 20 \times AC \times AV \times Au. \tag{1}$$

A high exportability score $\xi(\nu_{id})$ implies a vulnerability is highly exploitable for a remote attacker. For a network operator, this information combined with the impact a vulnerability has on ongoing network activities, describes how sever a vulnerability is to the safety of a monitored network.

Our vulnerability model also encompasses CVSS base scores, which model the impact of a vulnerability. Confidentiality C, Integrity I and Availability A are referred to as Impact scores. Loss of Confidentiality implies that informational disclosure, loss of integrity implies modification of information and loss of availability implies reduced performance up to a total shut down of an affected network device. The CVSS score describing the impact of a vulnerability results in three functions $\xi_{Confidentiality}$, $\xi_{Integrity}$ and $\xi_{Availability}$:

$$\begin{aligned} \xi_{Confidentiality} &: \nu_{id} \rightarrow \{None, Partial, Complete\}, \\ \xi_{Integrity} &: \nu_{id} \rightarrow \{None, Partial, Complete\}, \text{and} \\ \xi_{Availability} &: \nu_{id} \rightarrow \{None, Partial, Complete\}. \end{aligned} \tag{2}$$

Equation 2 describes the local effect of a vulnerability on confidentiality, integrity or availability by linking the vulnerability to a corresponding impact score value $\{None, Partial, Complete\}$. Impact score values "None" implies that no confidentiality, integrity and availability compromise is possible. "Partial" implies that the system's compromise is limited to the software layer and the impact score 'Complete" is chosen, when an exploited vulnerability results in a full compromise down to operation system level.

3.2 Network Model

We rely on the network model introduced in [18].

Network Device.

Let MAC and IP be non empty sets of MAC and IP addresses, respectively ($MAC \cap IP = \emptyset$), then

$$D^{CY} \subseteq \mathcal{P}(MAC) \setminus \{\emptyset\} \times \mathcal{P}(IP) \tag{3}$$

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.

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

$$SERV : D^{CY} \times \Psi \times \mathbb{N} \rightarrow S. \tag{4}$$

To derive all network services hosted by a device d_j, we define the 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

$$d_j = HOSTS^{-1}(s_i^j). \tag{5}$$

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

$$DEV : \mathcal{P}(IP) \times \mathbb{N}. \tag{6}$$

This allows us to derive all involved network services for a given IP-address and port pair by $HOSTS(DEV(sIP, sPort)) \rightarrow \mathcal{P}(S)$. 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.

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

$$P = (sIP, sPort, dIP, dPort, \psi, t), \tag{7}$$

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

Statically and dynamically assigned ports.

Requests are often sent through a dynamically assigned port. Dynamically assigned ports are chosen from specifically assigned port ranges [26]. 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. 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

$$s_*^j \in S, \tag{8}$$

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.

Network Flow.

Based on Equation 7, 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_i^j and s_k^l, 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

$$\begin{aligned} SDEP = \{(s_i^j, s_k^l) \mid s_i^j \text{ sends a packet to } s_k^l \\ \text{in the period under consideration.}\} \end{aligned} \tag{9}$$

Additionally, we distinguish requests and responses exchanged between network services based on Equation 8. 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

$$\begin{aligned} SDEP^{rq} = \{(s_*^j, s_k^l) \mid s_*^j \text{ sends a request to } s_k^l \\ \text{in the period under consideration,}\} \end{aligned} \tag{10}$$

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

$$\begin{aligned} SDEP^{rsp} = \{(s_k^l, s_*^j) \mid s_k^l, \text{ sends a response to } s_*^j \\ \text{in the period under consideration.}\} \end{aligned} \tag{11}$$

4. LOCAL VULNERABILITY IMPACT

Vulnerability scanning a data-communication network is the process of assessing whether software vulnerabilities can be linked to monitored network devices. Software vulnerabilities can be linked to operating systems, software or firmware [9, 22]. Vulnerability scanning provides us with a mapping function $SVULN$. This mapping function $SVULN$ allows us to link a vulnerability $\nu_{id} \in N$ to network services $s_i \in S$ in a monitored data-communication network. Such that we are able to associate a network service $s_i \in S$

$$SVULN : s_i \rightarrow \nu_{id} \tag{12}$$

with a vulnerability ν_{id}.

Let us assume a vulnerability scanner links network service s_i to a vulnerability $\nu_{id} \in N$. Given that vulnerability ν_{id} has an confidentiality $\xi_{Confidentiality}(\nu_{id})$, integrity $\xi_{Integrity}(\nu_{id})$ or availability $\xi_{Availability}(\nu_{id})$ impact score value "Complete", we assume that the compromise down to operation system level leads to all network services being compromise with respect to confidentiality, integrity or availability. This is computed by the following function:

$$DVULN : HOSTS(HOSTS^{-1}(s_i)) \rightarrow \nu_{id}. \tag{13}$$

Then, all network services hosted on the same network device as s_i are also affected with respect to confidentiality, integrity or availability by vulnerability ν_{id}.

4.1 Network Vulnerability Impact

We consider three views of network impact for a software vulnerability $\nu_{id} \in N$: confidentiality, integrity and availability impact on a network. CVSS describes the local effect on confidentiality, integrity and availability of vulnerabilities, which we capture within our vulnerability model. Based on this vulnerability model, the task of network vulnerability assessment is to analyze the global effect on the overall network.

4.1.1 Confidentiality Impact

A software vulnerability with a confidentiality impact signifies the threat of information disclosure. If a network service was able to request information from other network devices, this information could be leaked as well. Hence, these network devices are affected by the threat of information disclosure as well. The set of network services ASC, based on Equation 10, which are directly affected by detected software vulnerabilities with a confidentiality impact is defined as

$$ASC = CC\left((\forall s_i \in S : SVULN(s_i), map(asSet, SDEP^{rq})\right)), \tag{14}$$

where CC denotes the connected components (CC) of the hypergraph given as parameter ($asSet$ maps a tuple into a set of components).

4.1.2 Integrity Impact

A software vulnerability with an integrity impact signifies the threat of data modification. If data on a network device can be modified, responses from this network device cannot be trusted. Consequently, these network devices are also affected by the threat of data modification. Based on Equation 11, the set of network services ASI affected by software vulnerabilities with an integrity impact is defined as

$$ASI = CC\left((\forall s_i \in S : SVULN(s_i), map(asSet, SDEP^{rsp})\right)). \tag{15}$$

4.1.3 Availability Impact

Software vulnerabilities with an availability impact could lead to performance degradation, therefore all network services relying on responses from this network device are affected. The set of network services ASA affected by software vulnerabilities with an availability impact is defined as

$$ASA = CC\left((\forall s_i \in S : SVULN(s_i), map(asSet, SDEP^{rsp})\right)). \quad (16)$$

5. NETWORK ACTIVITY DISCOVERY

Companies, organizations and enterprises have a workflow, which translates into network activities within their data communication network. Workflows often cause reoccuring network activity patterns. We understand these network activity patterns as workflows and network service dependency analysis has the purpose of detecting network activity events. and we rely on an automatic network service dependency methodology called Mission Oriented Network Analysis (MONA) [18]. In the following, we thus rely on the same network model introduced by MONA.

5.1 Indirect Dependencies

In the context of this work, we introduce network dependency analysis as a basis for network activity mining. Following MONA [18], normalized cross correlation provides us with a heuristic for learning indirect dependencies $ISDEP$, which captures all LR (local-remote) and RR (remote-remote) dependencies by the following equation:

$$ISDEP = SDEP \bowtie SDEP. \quad (17)$$

An indirect dependency event $\iota_i = \{\delta(s_i^j, s_k^l), \delta(s_m^j, s_o^n)\}$ is based on direct dependency events δ. The set of all indirect dependencies $ISDEP$ translates into a set of indirect dependency events $\Omega = \{\iota_0, \dots, \iota_n\}$. MONA creates probabilities $\varrho(\tau_{delay}) \in P_\varrho$ ranging between $\varrho(\tau_{delay}) = [0, \dots, 1]$ and provides a set of observed network activity events $F \subseteq \Omega$.

$$\begin{aligned} p(\iota_{hg}(\delta_h(s_i^j, s_k^l), \delta_g(s_m^j, s_o^n)) \mid \delta_h(s_i^j, s_k^l) \wedge \delta_g(s_m^j, s_o^n)) \\ = \varrho_{r,s}(\tau_{delay}) \end{aligned} \quad (18)$$

Obviously, there is a level of uncertainty associated with detected indirect dependencies. By understanding indirect dependencies as indirect dependency events, we are able model the probability of uncertain event by relying on Kolmogorov axioms of probability theory [14].

5.2 Network Activity Mining

Normalized cross correlation provides an heuristic approach for detecting indirect dependency events and the result is described by a probability space (Ω, F, P_ϱ) described in Section 5. Based on the probability space, we model network activities as Hidden Markov Models (HMMs). Using HMMs, it is possible to identify the probability whether a specific network activity is present or not. An HMM $\lambda = (a_{ij}, e_{\iota_{kl}}, \pi)$ representing a network activity is defined as follows:

- n states $\Omega = \{\iota_0, \dots, \iota_{n-1}\}$,
- an alphabet $\Delta = \{\delta_0, \dots, \delta_{m-1}\}$ of m symbols,
- a transition probability matrix $a_{ij} = \iota_i \times \iota_j$,
- emission probabilities $e_{\iota_{kl}}(\delta_l)$ representing the probability of a state ι_{kl} emitting symbol δ_l and
- initial state distribution vector $\pi = \pi_o$.

We refer to a sequence of observed symbols as $O = \delta_0, \delta_1, \delta_2, \dots$ and a sequence of states as $Q = \iota_0, \iota_1, \iota_2, \dots$. Based on tumbling windows $w_{t_i} \in W$

$$W = w_{t_0}, \dots, w_{t_i}, \dots, w_{t_{n-1}} \quad (19)$$

with a shift Δ_t, we derive indirect dependency events based on observed direct dependencies. Direct dependencies imply that network packets are exchanged between two network services. Normalized cross correlation is a heuristic approach, hence observed network activity events can be untrue and existing indirect dependencies might not be detected. However, repeatedly reoccurring network activity events are very likely to be actual network activities events.

The parameters of a_{ij} and $e_{\iota_{kl}}(\delta_l)$ the HMM $\lambda = (a_{ij}, e_{\iota_{kl}}, \pi)$ can be learned over multiple tumbling windows $w_{t_i} \in W$ and $t_i \in [t_0, \dots, t_p]$ by:

$$a_{ij} = \frac{A_{ij}}{\sum_{q=\{0,\dots,p\}} A_{iq}}, \quad (20)$$

where A_{ij} is the number of observed state transitions from state ι_i to ι_j over p tumbling windows and it is normalized over all of ι_i's outgoing state transitions. An emission probability $e_{\iota_{kl}}(\delta_l)$ is derived as

$$e_{\iota_{kl}}(\delta_l) = \frac{E_{\iota_{kl}}(\delta_l)}{\sum'_{\forall \iota_{xl}, \delta_l \subset \iota_{xl}} E_{\iota_{xl}}(\delta_l)}, \quad (21)$$

where $E_{\iota_{kl}}(\delta_l)$ is the number of times that state ι_{kl} is observed, when symbol δ_l is emitted. This is normalized over the number of occurrences of all states $\iota_{xl} \subset \delta_l, \iota_{xl} \in F$, which also emit symbol δ_l. By observed network traffic within an monitored network traffic, this HMM allows us to automatically derive network activities based on network service dependency analysis. This introduced methodology is applied to a real-life network and the results of this experimental evaluation are shown in the next section.

5.3 Network Dependency based Vulnerability Assessment

To identify the set of network activities, which are affected by a software vulnerability, we link affected network services ASC, ASI and ASA to indirect dependencies as derived in Equation 17. Hence, the set of indirect dependencies IS^C, IS^I and IS^A affected a loss of confidentiality, integrity and availability is defined as

$$\begin{aligned} IS^C &= SCC\left((\forall s_i \in ASC, map(asSet, ISDEP))\right), \\ IS^I &= SCC\left((\forall s_i \in ASI, map(asSet, ISDEP))\right), \text{and} \\ IS^A &= SCC\left((\forall s_i \in ASA, map(asSet, ISDEP))\right). \end{aligned} \quad (22)$$

where SCC denotes the strongly connected components (SCC) of the hypergraph given as parameter ($asSet$ maps a tuple into a set of components).

Example for a Strongly Connected Component.

Suppose that a network service for the network activities illustrated in Figure 1 is linked to a software vulnerability $cveId_i$ by the mapping function $SVULN$. Then set of affected indirect dependencies is described in Figure 2.

6. EVALUATION

Using network activities for evaluating the impact of vulnerabilities results in a strong dependency on accurately mined network

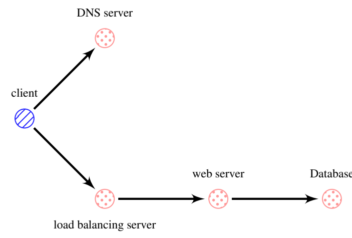

Figure 2: Example for a network dependencies, which would be affected by any network service having a software vulnerability.

activities. There are two parts of to our experimental evaluation, first we show the results of deploying our network based vulnerability assessment methodology within the operational network in Subsection 6.1. To provide a comparison with two state of the art network dependency discovery methodologies based on a known ground truth of all existing and non existing indirect dependencies, experiments based on synthetic networks are presented in Subsection 6.2. We generate synthetic networks based on response times observed in the operational, real-life network and conduct a comparative evaluation with Orion [5] and Sherlock [4].

6.1 Experimental Evaluation based on an operational Network

The disaster recovery site of an energy distribution network, provided an Italian water and energy distribution company, was available for non-invasive experimentation. Based on this network, we are able to collect and analyze real-life network traffic and also scan the network for present software vulnerabilities. Figure 3 shows all network service dependencies detected by MONA. These network service dependencies were considered complete and correctly identified by network operators.

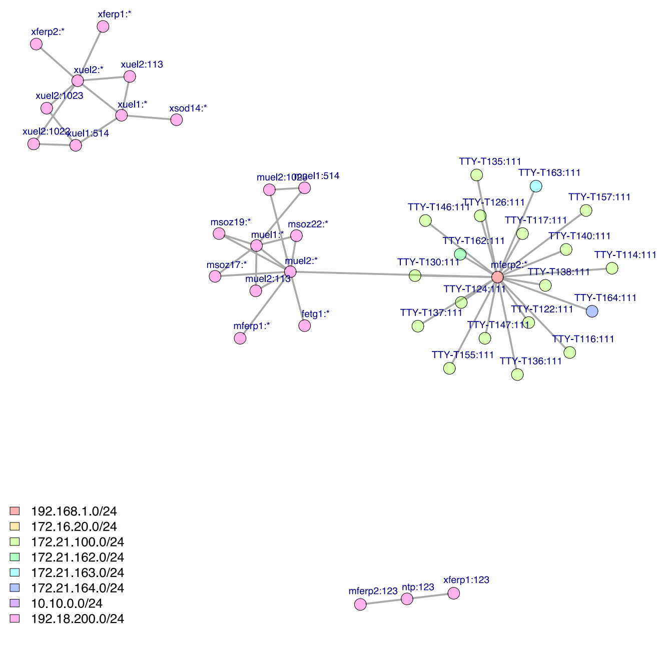

Figure 3: Network service dependency analysis in an energy distribution network.

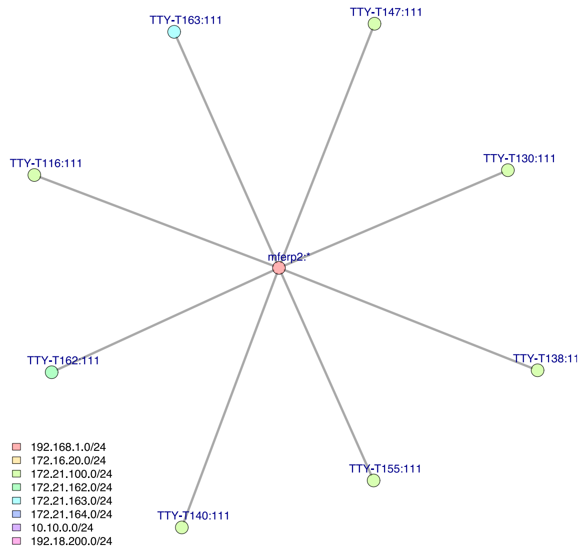

Figure 4: Network dependency based vulnerability assessment for vulnerabilities CVE-2007-5423 and CVE-2010-2075, who were detected on mferp2.

Figure 4 shows the result of network dependency based vulnerability assessment for software vulnerabilities CVE-2007-5423 and CVE-2010-2075 that were detected via network scanning on network device mferp2. Both software vulnerabilities can be exploited by automated code and therefore according to Equation 1 are easily exploitable for remote attackers. CVE-2007-5423 is a vulnerability that allows remote attackers to execute arbitrary code in TikiWiki 1.9.8 and CVE-2010-2075 is an unauthorized-access vulnerability due to a backdoor in UnrealIRCd 3.2.8.1. TTY-T[116-163] are remote terminal units of substations, which are dependent on requests from the front end server mferp2. Hence, network based vulnerability assessment concludes that TTY-T[116-163] are affected by CVE-2007-5423 and CVE-2010-2075, which were detected on mferp2.

6.2 Experimental Evaluation based on synthetic Networks

The disaster recovery site of an energy distribution network, provided by an Italian water and energy distribution company, was available for network traffic analysis. Based on this network, we are able to collect and analyze real-life network traffic. Real-life network traffic consists of network services frequently to rarely interacting and we are able to determine typical response times. Some network services have a common purpose and show similar communication patterns. As this network is a real-life network, absolute knowledge of all network dependencies is not available. During first experiments on data sets from the disaster recovery site, we often found new network dependencies that had been previously forgotten by the network operators. As every network relies on third party software, which might have their own network dependencies unknown to network operators, collecting a known ground truth on this network is not feasible. Hence, we extracted the communication patterns of known network dependencies and used this information to develop a random network generator in ns-3 [2]. This random network generator can create synthetic data sets with a known ground truth for testing our network dependency analysis. As we are interested in equally maximizing precision and recall, we rely on the F-measure in the following as a test methodology. Figure 5 shows the F-measures based evaluation for MONA,

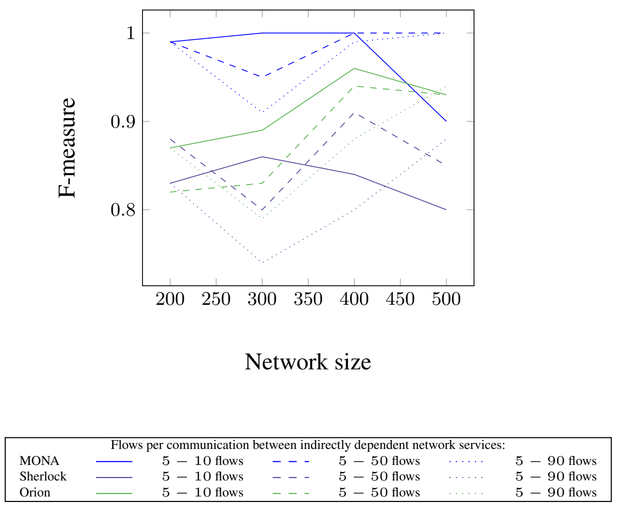

Figure 5: F-measures for MONA, Sherlock and Orion in increasingly large networks with 10 direct dependencies 20 indirect dependencies. The number of flows per communication between indirectly dependent network services is varied between 5-10, 5-50 and 5-90.

Sherlock and Orion in increasingly large networks with 10 direct dependencies 20 indirect dependencies. The number of flows per communication between indirectly dependent network services is varied between 5-10, 5-50 and 5-90. As all four compared methodologies rely on analyzing network traffic patterns, a correlation between resulting F-measure curves becomes apparent. Generally, Orion surpasses Sherlock except for smaller networks with less than 275 network devices and 5-50 flows per communication between indirectly dependent network services. MONA almost always surpasses Sherlock and Orion, except for networks with 475 to 500 network devices and 5-10 flows per communication between indirectly dependent network services. In this case Orion surpasses MONA with a margin of less than 0.15.

Figure 6 shows the F-measures based evaluation for MONA, Sherlock and Orion in increasingly large networks with 70 direct dependencies 70 indirect dependencies. The number of flows per communication between indirectly dependent network services is varied between 5-10, 5-50 and 5-90. MONA's F-measure results clearly surpass Sherlock's and Orion's in this experimental set up with more direct and indirect dependencies. Generally, Orion surpasses Sherlock's F-measure results except for networks with less than 275 network devices and 5-90 flows per communication between indirectly dependent network services. For networks with 200 network devices Sherlock and MONA diverge by less then 0.01 for 5-90 flows per communication between indirectly dependent network services.

7. CONCLUSION

We have introduced a novel network based vulnerability analysis approach. Network service dependency analysis allows the automatic detection of ongoing network activities. Based on automatically detected network service dependencies, we are able to link exploitable software vulnerabilities to ongoing network activities. The proposed framework is fully automated and is able to integrate

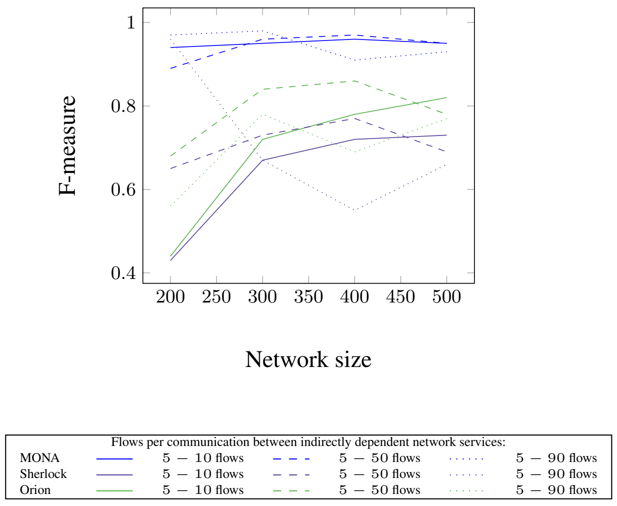

Figure 6: F-measures for MONA, Sherlock and Orion in increasingly large networks with 70 direct dependencies 70 indirect dependencies. The number of flows per communication between indirectly dependent network services is varied between 5-10, 5-50 and 5-90.

vulnerability specification from the bug-reporting community and helps network operators develop a deeper understanding on how networks are affected by software vulnerabilities.

Acknowledgments

This work was partly supported by the Seventh Framework Programme (FP7) of the European Commission as part of the PANOPTESEC integrated research project (GA 610416).

8. REFERENCES

[1] National vulnerability database. https://nvd.nist.gov/, 2016.
[2] Afanasyev, A., Moiseenko, I., and Zhang, L. ndnSIM: NDN simulator for NS-3. Technical report, NDN, 2012.
[3] 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)* (2006), ACM, pp. 97–102.
[4] Bahl, P., Chandra, R., Greenberg, A., Kandula, S., Maltz, D. A., and Zhang, M. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *ACM SIGCOMM Computer Communication Review* (2007), vol. 37, ACM, pp. 13–24.
[5] Chen, X., Zhang, M., Mao, Z. M., and Bahl, P. Automating network application dependency discovery: Experiences, limitations, and new solutions. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (2008), vol. 8, pp. 117–130.
[6] Clapper, J. R. Statement for the record, worldwide threat assessment of the us intelligence community. http://www.dni.gov/index.php/newsroom/testimonies/209-congressional-testimonies-2015/1174-statement-for-the-record-worldwide-threat-assessment-of-the-u-s-ic-before-the-sasc, 2014.

[7] CORPORATION, I. 2015 cyber security intelligence index, july 2015.

[8] DE BARROS BARRETO, A., COSTA, P. C. G., AND YANO, E. T. A semantic approach to evaluate the impact of cyber actions on the physical domain.

[9] GOHIL, B. G., PATHAK, R. K., AND PATEL, A. A. Federated network security administration framework.

[10] GOODALL, J. R., D'AMICO, A., AND KOPYLEC, J. K. Camus: automatically mapping cyber assets to missions and users. In *Military Communications Conference (MILCOM)* (2009), IEEE, pp. 1–7.

[11] ISO, I., AND STD, I. *ISO/IEC 13335-1: Management of information and communications technology securityâĂŤPart 1: Concepts and models for information and communications technology security management*. 2004.

[12] ISO, I., AND STD, I. Iso 27005: 2011. *Information technology–Security techniques–Information security risk management. ISO* (2011).

[13] JAKOBSON, G. Mission cyber security situation assessment using impact dependency graphs. In *Information Fusion (FUSION)* (2011), pp. 1–8.

[14] KOLMOGOROV, A. N. Foundations of the theory of probability.

[15] KOTENKO, I., AND CHECHULIN, A. A cyber attack modeling and impact assessment framework. In *Cyber Conflict (CyCon), 2013 5th International Conference on* (June 2013), pp. 1–24.

[16] MARTIN, B., SULLO, C., AND KOUNS, J. Osvdb: open source vulnerability database.

[17] MITRE. Common vulnerabilities and exposures. https://cve.mitre.org/, 2000.

[18] MONA LANGE, R. M. Time Series Data Mining for Network Service Dependency Analysis. In *The 9th International Conference on Computational Intelligence in Security for Information Systems* (2016), Springer International Publishing.

[19] MURRAY, A. T. An overview of network vulnerability modeling approaches. *GeoJournal 78*, 2 (2013), 209–221.

[20] MUSMAN, S., TANNER, M., TEMIN, A., ELSAESSER, E., AND LOREN, L. Computing the impact of cyber attacks on complex missions. In *Systems Conference (SysCon), 2011 IEEE International* (April 2011), pp. 46–51.

[21] NATARAJAN, A., NING, P., LIU, Y., JAJODIA, S., AND HUTCHINSON, S. E. NSDMiner: Automated discovery of network service dependencies. In *IEEE International Conference on Computer Communications (IEEE INFOCOM 2012)* (2012), IEEE.

[22] OF THE HONG KONG SPECIAL ADMINISTRATIVE REGION, T. G. An overview of vulnerability scanners. http://www.infosec.gov.hk/english/technical/files/vulnerability.pdf, 2008.

[23] OU, X., GOVINDAVAJHALA, S., AND APPEL, A. W. Mulval: A logic-based network security analyzer. In *USENIX security* (2005).

[24] SAWILLA, R. E., AND OU, X. Identifying critical attack assets in dependency attack graphs. In *Proceedings of the 13th European Symposium on Research in Computer Security* (2008), pp. 18–34.

[25] TECHTARGET. Growing threats make security vulnerability management essential. http://searchsecurity.techtarget.com/video/Growing-threats-make-security-vulnerability-management-essential, 2015.

[26] TOUCH, J., KOJO, M., LEAR, E., MANKIN, A., ONO, K., STIEMERLING, M., AND EGGERT, L. Service name and transport protocol port number registry. *The Internet Assigned Numbers Authority (IANA)* (2013).