Implementing a Network-based IDS(NIDS) in a Software-Defined Network(SDN) to detect and contain Ransomware

(revised May 2019)

Jay Chow, Kevin Hamilton, James Ballard, Graduate Students, Johns Hopkins University Information Security Institute

Abstract - Network-based intrusion detection systems(NIDS) is an important aspect of network security, providing a layer of defense by monitoring network traffic for malicious traffic or patterns and ultimately alert administrators when bad malicious traffic such as receiving a public encryption key for a ransomware attack from remote to local is detected. Software Defined Networking(SDN) is a powerful network architecture that is flexible, agile and programmable where the data and control plane separated, offering a centralized controller that is programmable. Ransomware is a form of malware that uses a public key to encrypt data and non-system files to prevent users from accessing them until a ransom is paid. Modern ransomware connect the infected machines back to a command and control server for to key exchange needed for encryption or decryption. Since the SDN architecture consists of a centralized SDN controller or Network Operating System(NOS) that creates an abstraction for the network applications from lower-level forward devices, the logic for ransomware detection and containment could be implemented in the SDN controller.

1 Project Goal

The goal of our final research project is to implement a NIDS in a SDN to detect and contain ransomware using GENI and CloudLab. GENI (Global Environment for Network Innovations) provides a virtual laboratory for networking and distributed systems in a scalable manner. This allows us to obtain additional compute resources using layer 2 networks, install custom software and operating systems on these computing resources, control how network switches handle traffic flows and run layer 3 and above protocols by installing protocol software and providing flow controllers for the switches. We are also able to use Cloudlab, a security hands-on resource for SDN and NFV. First, we conducted literature review to assess the current state of intrusion detection systems in a SDN architecture and network signatures of ransomware. The effort in doing so is to support our goal of early detection of ransomware attack network signatures in order to contain the attack with the flexibility of SDN. Our project will focus specifically on openflow as our SDN standard. Detection and containment of modern ransomware is a difficult problem as malware developers are starting to use more sophisticated techniques such as encrypting the traffic between the infected host and the command and control server for key exchange.

1.1 Introduction To SDN

Software Defined Networking(SDN) is a powerful network architecture that is flexible, agile and fully-programmable. Openflow is the framework that offers this standardization and programmability for the software switches that are used in SDN architectures. In the SDN architecture, the network and control plane are separated, offering a centralized controller that can be easily programmable. This is powerful because in more complicated topologies, this can allow for duplication of traffic through a switch to a monitor, a machine decentralized from the network and dedicated to monitoring for attacks. We call this a mirroring port, which has value when we install a monitoring node where all traffic that goes through the switch is duplicated to it through the mirroring port.

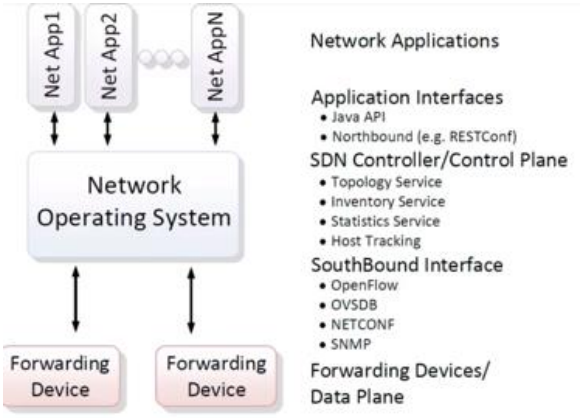

Fig. 1. SDN architecture

As shown in Fig.1, Forward devices receive packets, take actions on packets and update counters. Actions include dropping the packet, modifying packet headers and sending packets out on a single port or multiple ports. Forwarding devices can cache entries so that they do not need to query the SDN controller every time for packets that have been seen before. As such, future packets of the same type can take a fast path. Examples of forwarding devices include a hardware switch(greater performance) that has a programmable interface like Openflow(protocol for handling packet instructions) or a software switch(greater flexibility) like open V switch(OVS).

The SDN controller or NOS will have core services to provide a programmable interface to the network applications The instructions on how to handle packets originate from the SDN controller. NOS has a global view of ALL forwarding devices below it and can communicate packet forwarding instructions to them. This creates a simplified and abstract view as a single, large switch for network applications to implement policies. Note that the NOS is logically centralized, NOT physically centralized.

Network applications are network focused. Now when a packet flows through the forwarding device, the device will parse the packet headers and either already knows what to do with the packet or queries the SDN controller. Via the NOS or SDN controller as a translator, the network applications will determine what action to take on a packet and will push this information down to the forwarding device.

1.2 Introduction To OVS

OVS is an open source OpenFlow capable virtual switch that is used with hypervisors to interconnect VMs within a host and VMs between different hosts across networks. Could also be used on some dedicated switching hardware, could be a critical piece in a SDN solution.

Supports traditional switching features:

1) VLAN tagging, 802.1q trunking
2) Standard spanning tree protocol(802.1D)
3) LACP
4) Port mirroring(SPAN/ RSPAN)
5) Flow export(e.g sflow, netflow and ipfix)
6) Tunneling(e.g GRE, VXLAN, IPSEC)
7) QoS control

It is necessary to understand OVS as we will be utilizing OVS for the implementation of our project in the near future. As such, it is vital to be familiar with the architecture of OVS that consists of three core components:

1) VSWITCHD is the core component and runs in user space.

2) OVSDB-SERVER is a database server that stores the configurations, configuration changes are persistent and will survive a system reboot.
3) Kernel module is when a packet arrives at a virtual switch, and there is a cache match in the kernel module, then cached actions are taken. If no cache match for packet arriving in OVS, the packet is punted into VSWITCHD in user space. Cached packets have fast path.

1.3 Introduction To Ransomware

Ransomware is a form of malware that uses encryption preventing users from accessing data until a ransom is paid. Modern variants of ransomware connect the infected machines back to a command and control server for key exchange needed for encryption and decryption. Similarly to SDNs, the idea of ransomware, has been around since the early days of the internet. However, the ability for the ransomers has become increasingly easier with the widespread usage of digital transactions. Specifically, the creation of cryptocurrencies allows for ransoms to be paid but also makes them nearly untraceable. In 2013, the first version of cryptolocker was created and with the subsequent variants of the ransomware by the end of 2015 it was estimated that $27 million had been paid in ransom to the authors. Ransomware since has only grown to be a larger and larger problem, in 2014-2015 Kaspersky

found that crypto ransomware rose 448 percent. Like many other forms of malware authors have turned to selling them on the dark web. This has also been the case with ransomware increasing the quality of the malicious software to be nearly the same as benign off the shelf software. Ransomware still dominates much of the news in terms of major cyber security incidents, on Tuesday, March 19th, 2019 Norsk Hydro, one of the largest aluminium manufacturers had its nearly all of it's IT infrastructure affected by a ransomware attack. Over the last ten years the attackers seem to be ahead of the curve with how effective their attacks have been. It goes without saying that more research should be done in the area of ransomware defense.

2 Literature Review

In "An SDN-Supported Collaborative Approach for DDoS Flooding Detection and Containment", the paper explores a collaborative approach of attack detection and containment that is unique to SDN. This paper focuses on a system of monitors distributed over a network with a monitor node attached to the Open vSwitch(OVS). A monitor is used to detect anomalies in network traffic.

In terms of SDN supported security and SDN flow-based rules, SENSS provides an interface for security attack detection initiated by a victim through queries, where Internet Service Providers trace collaborate to trace back a network-based attack such as DDoS. The OrchSec architecture uses decoupled monitors and SDN controllers to mitigate different types of attacks. The NICE framework employs an IDS agent to monitor mirrored traffic and to propose potential attack countermeasures. The detection is based on attack graphs of known system vulnerabilities by an attack analyzer at the SDN controller. This system is close to the traditional distributed IDS where a centralized manager or director aggregates inputs from multiple sensors or agents. OpenFlow provides a mechanism to support an IDS sensor.

Global Environment for Network Innovations(GENI) has been an emerging virtual infrastructure. GENI and SDN gives experimenters control of network packet forwarding within a slice, permits GENI aggregate operators a flexible mechanism to create virtual network slices for experiments, and provides system administrators the opportunity to allocate network traffic into specific slices with appropriate authorization control. As an SDN instantiation, OpenFlow is used to forward network traffic in GENI experiments.

The paper's investigation includes three generic tasks: first a monitor raises an alert when an attack occurs; second, informed by the alert, a network attack correlator further evaluates the situation and if needed, identifies the attackers location and the attack path. If applicable, OVS take mitigation actions to block attack traffic or reroute attack packets. In large networks with many monitors and correlators, requirements to optimize the detection and mitigation actions become significant.

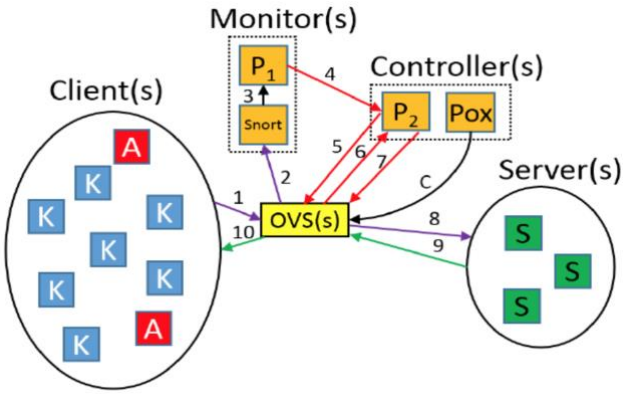

Fig. 2. Architecture for anomaly detection with OVS

The distributed monitors constantly observe the network traffic for anomalies. The monitor can employ different anomaly detection algorithms to flag a range of potential attacks. The correlators at the OVSs respond to alerts from monitors on demand. Once an alert is received, a correlator immediately takes action based on the type of alert. As such, multiple correlators need to communicate with one another to access the relevant OVSs to reveal the attacks and generate insights of the attack. The SDN controllers take actions to modify the network flows in attack mitigation. Confirmation of attack triggers attack containment actions as planned. These containment actions could be executed by SDN controllers, or the Network Operating System, and may range from dropping the attack packets, deploying honeypots to trick the attackers for more evidence, dynamically re-configuring the network or reshaping the traffic. Note that The attack may originate from a network segment different from where the monitor is.

In "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments", this second paper explores an efficient and scalable mechanism for

performing anomaly detection and mitigation in SDN architectures. Flow statistics may reveal anomalies triggered by large scale malicious events. Then, network owners and operators can raise mitigation policies against these threats. There is a need for data center operators and cloud service providers to set up programmable network environments for scalability and flexibility. What we learned from this paper is how to expand OpenFlow(OF) functionalities such as rich diversity in traffic management, load balancing, routing and firewall with scalable mechanism for performing anomaly detection and mitigation in an OF SDN architecture.

This paper's main contributions include

- Architectural design of a modular mechanism that permits anomaly detection and mitigation on SDN environments
- Scalability improvements, in comparison with the already proposed native OF approaches, employing sFlow protocol for flow-based data gathering.
- Effective reduction of the required communication between switches and OF controllers, eliminating the potential control plane overloading, under anomaly conditions to the data plane
- Performance tests using real traffic traces in order to validate the scalability and the effectiveness of the proposed sFlow-based approach compared to the native OF mechanisms in terms of anomaly detection accuracy and overhead on system resources (CPU usage, flow table size).

Lastly, we learned key design principles of the proposed architecture on anomaly detection and mitigation in SDN environments:

- Modular design with data gathering, anomaly detection and anomaly mitigation function decoupling
- Compatibility with any OF-enabled Layer 2 or Layer 3 device
- Elimination of OF related constraints regarding statistics collection
- Exploitation of data and control plane decoupling for fast anomaly detection and mitigation on real-time environments
- Scalable traffic manipulation using sFlow packet sampling techniques

Additionally, we learned about this IDS on SDN environments:

Additionally, we learned about this IDS on SDN environments:

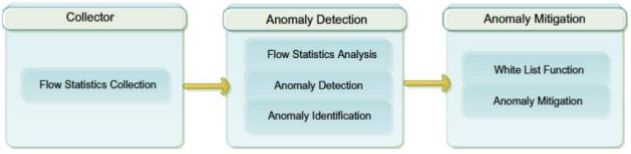

Fig. 3. Architecture for proposed IDS for SDN environments

The paper leveraged the packet sampling capability of sFlow collection technique by decoupling entirely the flow collection process from the forwarding logic, because packet samples provide all the necessary flow-related statistics. This practice permits the use of more efficient and aggregated forwarding logic, since there is no longer a need for specific flow-entries. This, there is a significant reduction in the number of required flow entries, thus overcoming any possible flow table size limitations, which are typical for ordinary hardware OF switch implementations. This process gathers sufficient information for a reliable anomaly detection process, reducing the CPU-usage.

Table 3
Classification of anomalies based on entropy change.

Anomaly	Description	Entropy change
Distributed Denial of Service (DDoS) attack	An attack on a specific service, making the resource unavailable to its users	Significant decrease in dstIP and dstPort
Worm propagation	A self-replicating program that tries to infect other machines by exploiting a specific vulnerability	Significant decrease in srcIP and dstPort
Portscan	Sending probe packets to a wide range of ports in a specific host to check which services are available	Significant decrease in srcIP, dstIP Significant decrease in srcPort if attacker is not using random source ports

From above, we learned that we could implement an entropy-based method, independent of network topology and traffic characteristics that can be applied to monitor every type of network for anomaly detection and classification purposes. Entropy measures the randomness of a specific data set. High entropy values signify a more dispersed probability distribution, while low entropy values denote concentration of a distribution. The paper focused on 3 anomalies, namely DDoS, worm propagation and portscan.

For instance, an anomaly of an infected host trying to infect other hosts in the Internet (worm propagation) results in decrease of the entropy of the source IP addresses. The infected host machine produces a large number of flows, causing the same source IP address to dominate in the flow distribution of source IP addresses. On the other hand, during a port scanning activity, the entropy of the destination port increases due to the scan of random destination ports. Based on such fluctuations, the controller can identify the presence of an anomaly using predefined

thresholds on the changes in the corresponding entropy values.

For ransomware literature review, we read a paper that is highly relevant. In "Intelligent and Dynami Ransomware Spread Detection and Mitigation in Integrated Clinical Environment. Ransomware is still a major cybersecurity problem as existing anti-virus, IDS or firewalls are not suitable to detect new ransomware. It is also useful to note that not all traffic patterns generated by ransomware are distinguishable from normal traffic patterns(i.e application compressing files are similar to malware encrypting files) and malware developers use encrypted traffic to avoid payload inspection. Therefore, achieving an acceptable balance between detection and false alarm is hard. To solve this problem, the paper utilized various supervised and unsupervised machine machine learning to detect abnormal patterns in network communications. Once ransomware has been detected and classified, the mechanism isolates infected devices through the SDN paradigm and replaces software controllers using NFV techniques.

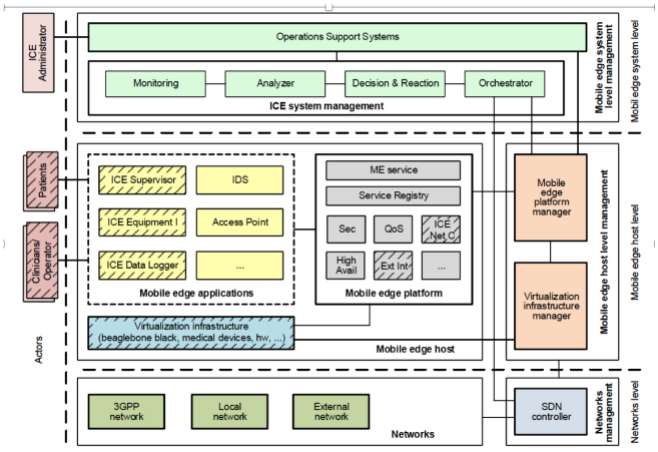

Fig. 4. ICE++ Architecture for Ransomware Detection and Mitigation in Mobile Edge Computing paradigm

The Mobile Edge System Level is the upper level of our architecture, focusing on defining and managing the behaviour of components(i.e interfaces, applications). Next, the middle level is the Mobile Edge Host Level, which focuses on running the mitigation countermeasures. The Networks Level is the lowest level that contains the physical infrastructure required to provide connectivity between the different mobile edge applications. The Networks management contains the SDN Controller, which is able to monitor and manage in real time and on-demand the communications of mobile edge applications.

3 Current Ransomware Detection

3.1 Host-Based Detection Practices

If we want to use an SDN-based intrusion detection system to be able to detect signatures of ransomware, it is important to understand both the vectors that are available to ransomware attackers, but also what the current state-of-the-art methods that exist for detecting and preventing them. Most of the current research efforts are focused on detecting ransomware at the host level relying on things found at the hardware level. These attempts can provide some amount of direction to transitioning to a network-based approach to detection. An approach that has been catching on in recent years has been using Canary files to detect the large amounts of encryption that occur during a ransomware attack. The computer intrusion detection system running on the host would do something like raising alert if it detects the Canary file being changed. A major drawback of this type of strategy is that normally ransomware requires the host system to become compromised which might result in any intrusion detection being disabled. Other work arounds Canary files have been employed by ransomware that will only encrypt recently used files or files of applications that have recently been run , because Canary files will likely have been created when the IDs was created it is unlikely that this file would be encrypted and deleted and therefore circumvented.

A more complex system that has been employed relies on monitoring things like disk reads and writes and uses statistical modeling combined with machine learning to generate results. A system designed by Kharraz and Kirda was able achieve perfect detection and only a 1% rate of false positives. This was accomplished using a multitude of detection methods. The first method it offers is comparing the entropy of read files to that of files being written to disk. the idea here is that ransomware needs to read and encrypt a file, this means that file writes would have significantly more entropy than the files being read in, due to the changes involved with encryption. That method is the most relevant to our proposed solution but there are some other addition piece that are worth mentioning. The detection engine also monitors for the deletion and overwriting of files. After applying a recursive feature

elimination algorithm it generating weighting scores for all the features that they had elucidated. Using those weights they experimented with the scoring threshold to create an IDS that had a TP rate of 1 and a false positive of .005, clearly an effective solution. It was even able to detect newer versions on ransomware it had not been trained on.

Even though this system performed very well, a major drawback of this system and others like it are that there are some performance restrictions that come with it, so this would not be a feasible solution to secure higher end processing devices. Another issue with these solutions is that while they can detect these attacks while they are occuring, they might not be able to stop the encryption from getting to some of the more important files on the system. This means that a better solution needs to stop the attack in its entirety. Even more advanced ideas forgo detection entirely in a certain sense. These instead seek to log any of the keys that are used by applications on the system, which would give the user access to the key used by the ransomware. Auction by hooking into libraries that are responsible for generating cryptographic keys and this would result in defeating the purpose of the ransomware because it would then be trivial for the user to decrypt their files. While all of these host-based methods seem rather promising most are vulnerable to a privilege escalation and deletion attack. This needs to be resolved if this strategy is going to be successful against more advanced attackers.

3.2 Network-Based Detection Practices

The other area of research for detecting ransomware attacks do focus on network traffic, this is the area of research that we wish to expand upon. The portion of a ransomware attack that's conducted over the network is any of the communication that is involved with the command and control center. An interesting approach that has been tried is blacklisting of known command and control IP addresses, but this is starting to fall out of favor because address randomization and DNS encryption are strong counters to this method. A more novel concept has been doing traffic analysis trying to detect the encrypted communications used by ransomware applications by analyzing the message entropy of HTTP traffic. It relies on the fact that encryption schemes act in a way like random functions such that their outputs should be uniformly distributed. In a non-ransomware case HTTP payload traffic will most likely represent the character distribution of whatever language the that the traffic is being conducted. There is some very promising research that joins both software-defined networking and the detection of ransomware attacks. This also takes advantage of the HTTP traffic that is used by ransomware software, but instead of relying on entropy, they use the size and sequence of payloads; paired with an instance-based learning style detector to attempt to make a determination. This was conducted in more a misuse detection style; using samples from a few ransomware families to generate IBL clusters based on characteristics in their netflows. Because these signatures were only somewhat unique the systems performance was diminished somewhat. The best performance they were able to achieve it was able to get was .98 detection and .05 false positive rate. It would be interesting to see what the results would have been if they had opted for something like a Markov chain style machine learning algorithm which would likely better represent the order events that identify the sequence of packets as malicious. The only concern with extending this area of research is that malware designers are starting to use HTTPS to encrypt their messages, which would render this method of detection obsolete almost completely. On the whole there are some very promising research that is looking into detecting ransomware, but the cat-and-mouse game of security will carry on.

4 Proposed Solution

On face value the centralized aspect of SDN should make it much easier to develop a NIDS that can function in a software-defined networking environment. The main advantage of an IDS in this atmosphere is that an IDS can be directly programmed into the SDN network controller or network operating system, this could directly solve the insertion and evasion problems that plagued early NIDS systems. In other words, a SDN could result in a more intelligent and centralized NIDS. Our proposed system would function as a kind of hybrid design of some of the current cutting edge practices in ransomware detection. The goal would be to set up an instance-based learning detection engine indie software defined networking monitor. This could allow the intrusion detection system to create a kind of rolling threshold for the network. This is because the monitor can actively record all of the flows going through the network, and therefore could actively keep track of metrics that we want to monitor. In this case the most important metric would be network flow entropy. This is because as stated earlier many current versions of popular ransomware communicate with their command

and control via encrypted HTTP payloads. A marked increase in entropy in a singular flow as compared to the rest of the network would be potentially indicative of the beginnings of a ransomware attack. This is because generically HTTP payloads contain ASCII text that should resemble english. The instance-based learning aspect is useful because it allows us to continually reset training data with incoming netflows. An additional benefit of monitoring all connections simultaneously is that the outliers should be fairly easy to identify. This would also allow for general increases in network entropy to not set of any false alarms. The great benefit that comes with deploying this solution in the SDN environment is that a response mechanism is fairly simple to implement. The ideal method for detecting the response would be to redirect traffic towards a honeypot on the network; because of the malleability of the network provided by the SDN controller the ransomware command and control wouldn't notice this shift and it would be possible to conduct research on the attack. This would provide a window into what ransomware designers are doing. If no honeypot is available on the network the network controller could simply block any traffic coming from or going to the command and control. For many types of ransomware this would stop the attack dead in its tracks. In the case of WannaCry it needs to get its encryption keys from the server before and malicious behavior starts. If this is detected and blocked the ransomware can not proceed and can be removed from the infected machine after an alert is raised.

5 Future Work

Our immediate future work will be to implement this NIDS in a SDN architecture using GENI, Cloudlab to provide a virtual test bed for us. Soley for academic and research We could then introduce open-source or self-written ransomware to test our NIDS in terms of detection and containment. One large problem with testing and training ransomware type detection techniques is getting a data set that represents the current type of attacks found in the wild. Future work could include the aggregation of data samples of ransomware attacks to be used in research to further detection. Mainly future focus needs to be placed on the testing on the effectiveness of proposed solutions in realistic type test environments.

6 References

[1] T. Chin, X. Mountrouidou, X. Li and K. Xiong, "An SDN-supported collaborative approach for DDoS flooding detection and containment," *MILCOM 2015 - 2015 IEEE Military Communications Conference*, Tampa, FL, 2015, pp. 659-664. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7357519&isnumber=7357245

[2] Fernández Maimó L, Huertas Celdrán A, Perales Gómez ÁL, García Clemente FJ, Weimer J, Lee I, "Intelligent and Dynamic Ransomware Spread Detection and Mitigation in Integrated Clinical Environments," *Sensors (Basel, Switzerland)*. 2019;19(5).

[3] "GENI", Available : https://www.geni.net/

[4] Cybersecurity and Infrastructure Security Agency, "Ransomware" Available: https://www.us-cert.gov/Ransomware

[5] Wikipedia, "Mobile Edge Computing" Available: https://en.wikipedia.org/wiki/Mobile_edge_computing

[6] Cisco, "What is ransomware?" Available: https://www.cisco.com/c/en/us/solutions/security/ransomware-defense/what-is-ransomware.html

[7] "Cloudlab, a Security Hands-on Labs in SDN/NFV"Available: http://irislab.me/lab.html

[8] M. Rosch, "Snort - Lightweight Intrusion Detection for Networks" Available: https://www.usenix.org/legacy/events/lisa99/full_papers/roesch/roesch.pdf

[9] B. L. Yun Feng, Chaoge Liu, "A new approach to detecting ransomware with deception", 38th IEEE Symposium on Security and Privacy (2017) https://www.ieee-security.org/TC/SP2017/poster-abstracts/IEEE-SP17_Posters_paper_26.pdf

[10] A. Continella, A. Guagnelli, G. Zingaro, G.D. Pasquale, A. Barenghi, S. Zanero, F. Maggi, "ShieldFS: a self-healing, ransomware-aware filesystem", Proceedings of the 32nd Annual Conference on Computer Security Applications - ACSAC 16, ACM Press (2016)

[11] A. Kharraz, E. Kirda, "Redemption: real-time protection against ransomware at end-hosts", Research in Attacks, Intrusions, and Defenses, Springer International Publishing (2017), pp. 98-119

[12] E. Kolodenker, W. Koch, G.Stringhini, M. Egele, "PayBreak: defense against cryptographic ransomware", Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security - ASIA CCS - 17, ACM Press (2017)

[13] "DNS blacklisting is dead. What's next?," Available: https://www.telecompaper.com/industry-resources/dns-blacklisting-is-dead-whats-next--1237470

[14] D. Mülders and P. Meessen, "Network-based Ransomware Detection," *HITB SECURITY CONFERENCE*, 2017.

[15] K. Cabaj, M. Gregorczyk, and W. Mazurczyk, "Software-defined networking-based crypto ransomware detection using HTTP traffic characteristics," *Computers & Electrical Engineering*, vol. 66, pp. 353–368, 2018.

[16] G. Cusack, O. Michel, and E. Keller, "Machine Learning-Based Detection of Ransomware Using SDN," *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization - SDN-NFV Sec18*, 2018.

[17] Feamster, N., Rexford, J., & Zegura, E. (2014). The road to SDN: an intellectual history of programmable networks. *ACM SIGCOMM Computer Communication Review*, *44*(2), 87-98.

[18] Richardson, R., & North, M. (2017). Ransomware: Evolution, mitigation and prevention. International Management Review, 13(1), 10-21,101. Available: https://search.proquest.com/docview/1881414570?accountid=11752

[19] Townsend, K. (2016, June 24). History and statistics of Ransomware. Available: http://www.securityweek.com/history-and-statistics-ransomware

[20] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," Computer Networks, vol. 62, pp. 122–136, 2014.

[21] https://www.sciencedirect.com/science/article/pii/S1389128613004003