Literature Survey on modern Intrusion Detection Systems based on SDN and Honeypot

Thivyavignesh Ramasamy Gurumurthy
MS in Computer Science
University of Southern California
tramasam@usc.edu

***Abstract*—In the cyberspace, the capabilities of an adversary, to find and exploit a system vulnerability to perform malicious actions, increase in proportion to the rising scale of computational and technological advancement. In correspondence, the defense mechanisms against such actions should also be revised and updated to either gain the upper hand or maintain a balance between the attack and defense. One such defense mechanism which plays an active role in detecting anomalous presence through periodic audits or behavioral analysis is commonly referred to as the Intrusion Detection System (IDS).**

In this survey, I plan to briefly elucidate the concept of Intrusion detection system and discuss the functionalities of some modern IDS exclusively focusing on IDS implementations on SDN architectures and IDS implementations using Honeypot technology.

The paper is structured as follows: Section I introduces the concepts of IDS, SDN and Honeypot, followed by Section II describing in detail some modern techniques based on SDNs and Honeypots and finally Section III concluding the paper.

***Index Terms*—Intrusion Detection Systems, Software Defined Networking, Honeypot**

I. Introduction

A system is said to be compromised or under attack when it malfunctions either completely defeating its purpose of installation or performing tasks which are suspicious. A task is labelled suspicious when it is either anomalous or when it violates the system policy. The consequences faced during or after an attack can be as severe as critical information leak or critical system breakdown past recovery.

The most practical approach to mitigate the severeness of the consequences is to detect the attack in its initial stages or to prevent the attack from entering the system or the network connected to the system. This ideology is the base for the development of Intrusion Detection systems.

A. Intrusion Detection System

As the name suggests Intrusion Detection Systems (IDS) are systems or models designed to detect intrusions within the network of systems or a specific system. Anderson [1] classifies penetrators into three categories namely

- **Masquerador** is either an outsider[1] who is not privileged to access neither the system nor its resources or an insider[2] with certain privileges, however defeating the system's procedural controls by spoofing to be a/another privileged user and gaining access controls tied to that level of privilege.
- **Legitimate user** is the insider with certain privileges, however misusing those privileges to leak information or misusing/overusing the system resources to make it unavailable for other users. [2] refers to these intruders as Misfeasors.
- **Clandestine user** is either an insider/outsider who is able to gain the root control or a high privilege user control so that he/she can evade audit trails to remain undetected.

The IDS needs to efficiently identify these penetrators and take necessary actions before any irrecoverable or serious exploit takes place. The IDS should be able to gather relevant information, analyze those and take necessary actions to either mitigate or prevent the attack. Moreover, it should be intelligent enough to evolve so that no such future events take place. A pragmatic design of intrusion detection systems makes this possible.

1) General Design of IDS: In general, IDS should include hardware or software components designed to perform each of the detection, analysis and notification functions. Along with performing their functions, there should be a reliable means of communication between these components to facilitate efficient data transfer. The three main modules, as shown in the figure 1, comprised of these components are

Information Collection

This is the module which collects raw data from the target. The target is either the host when the module is located at the host (computer system) or the network when the module performs information collection by monitoring the network. These information gathering is referred to as host based and network based respectively by [3].

In host based module, the sources of information are system logs, security specific logs, port scanners, buffer overflow checkers and other host installed applications. On the other hand, network based gathering involves multiple host based applications working in a distributed manner alongside network monitors for the medium of communication. By splitting the task of observation at multiple locations in the network, the global perspective of the data collected is revealed thus aiding in profound analysis of the data. The raw information, with or without preprocessing to some

[1]non-privileged user who is not a part of the organization
[2]privileged user within the organization

predecided specific format, is then sent to the detection module for further analysis.

Detection

This module is the crux of the intrusion detection systems, where the raw information sent from the collector modules are analysed using various techniques for identifying the state of the sytem. The analysis could potentially identify whether the system is already under attack or about an imminent passive threat. The approach of the analysis [2] could be either rule based which defines the ideal behaviour of a legitimate user, thus can distinguish the behaviour of an intruder or statistical anomaly based which defines expected behaviours of the users, aiding in detecting anomalous traces of executions. The result of the analysis along with the detailed report of corresponding events substantiating the result is sent to the response module.

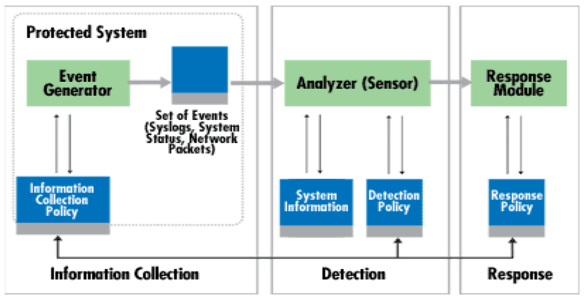


Fig. 1. IDS modules [source: http://www.techgenix.com]

Response

Response module is the notification authority that informs the administrator of the result of the detection module. The module can function in either proactive or reactive mode. The proactive module waits for the instruction from the administrator to proceed further to secure the system while the reactive mode is pre-instructed to execute certain countermeasures (for instance system lockdown) to mitigate the severeness of the compromise. Most of the response modules nowadays are hybrid in the aspects of using both modes where the administrator is notified after performing some rudimentary tasks as countermeasures giving the ultimate decision making authority to the administrator.

The policies of these modules are designed considering the organizational rules and regulations as well as the federal regulations. Bishop [3] refers to these modules as agents, director and notifier respectively. Moreover Bishop categorizes the handling of intrusions after its detection into the following 6 phases.

- Preparation : This phase prepares the IDS for any imminent attacks by defining follow up procedures and mechanisms to respond to the attacks. The system backups are taken as a cautious measure.
- Identification : This phase identifies the attack by comparing with the signatures of known attacks and classifies the severity of the attack (can also find the zero day attack by comparing with activity signatures).
- Containment : This phase tries to mitigate the damage by either monitoring passively or by reacting to the attack by undertaking some countermeasures.
- Eradication : This phase eradicates the attack and tries to prevent such attacks in the future by supplying information about the attack to the preparation phase.
- Recovery : After eradication of the attack, this phase helps in recovery of the system using the backups.
- Follow up : This is the final phase, where reporting the details to the cyber law enforcement departments to alert other vulnerable networks and to capture the attakcer behind the scenes.

In recent times, the IDS falls under the umbrella technology **SIEM** [4] (Security Information and Event Management) which combines the functionalities of Security Information Management (SIM) and Security Event Management (SEM).

2) *Classification of IDS:* [5] [6] classifies Intrusion Detection Systems based on the following factors:

- Detection Model
  - Signature based (referred as Misuse model by [3])
  - Anomaly based
  - Specification based (additionally classified by [3])
- Location or Information Source
  - Host based
  - Network based
  - Hybrid
- Time of Detection
  - Online
  - Offline
- Architecture
  - Centralized
  - Distributed
- Deployment
  - Single Host
  - Multiple Host
- Environment
  - Wired
  - Wireless
  - Heterogeneous

There can be more classifications of IDS based on some other factors. However, the core objective of the IDS is to detect and respond to unauthorized intrusions.

B. *Software Defined Networking Architecture*

In conventional networking architectures, the control plane that provides information to build the routing table and the data plane that looks up the routing table to route the packets are co-existent on the same networking device like switch. These networking devices perform additional tasks (control plane tasks) along with the main function of routing packets, thus incurring more work overheads that inhibits to perform with full efficiency for routing.

On the other hand, Software Defined Networking physically separated the control plane from the data plane existent on the networking devices by relocating it on a separate physical device, called SDN controller. The controller acts as a centralized control point that can program the functioning of the networking devices as per the controller policy, thus abstracting the business application and other higher layer application from the infrastructure underneath.

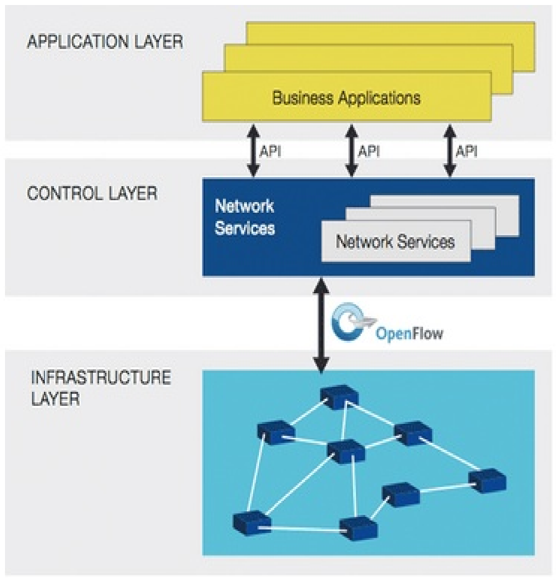


Fig. 2. SDN framework [source : www.sdxcentral.com]

As the SDN framework, shown in figure 2, the control plane interacts with business logics and other network applications using the Northbound APIs, thus deploying the network services on the controller. The controller communicates with the networking devices using the Southbound APIs, the most popular protocol being OpenFlow (OF) [7]. SDN also coined the term flow to be the sequence of packets sharing similar header fields. These flows are used by the controller to instruct the networking devices to build the routing table. This flexible architecture can revolutionize the workings of cloud and data centers, WANs and Network Virtualization.

C. Honeypots

The harsh reality that haunts the field of security for a very long time is that there are no perfectly secure real world systems. This ideology substantiates the claim that there are security vulnerabilities in the systems, thus motivating the attackers to find and exploit those. On the other side, the security administrators try to patch these vulnerabilities. One proactive approach that aids in detecting these vulnerabilities to prevent any system compromise is the technology of Honeypots.

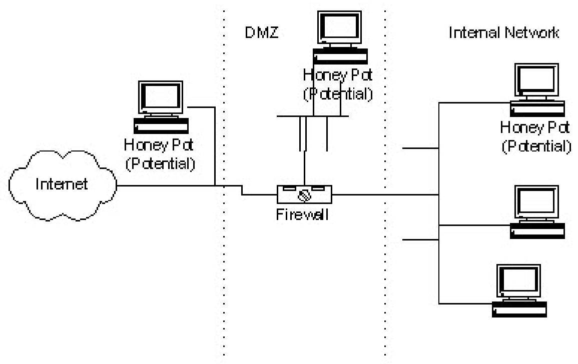


Fig. 3. Honeypot [source : www.thewindowsclub.com]

Honeypots (referred to as decoy systems [2]) are traps laid down for the attacker by the security administrators to understand the attack behaviours of the attackers, learning the vulnerabilities and technologies used by the attacker to initiate the protective measures to fix the vulnerabilities and strengthen the system defenses. Technically, honeypots are systems that run on original operating system providing valid services, pretending to be the original server that are targetted. However, the system uses no real data, rather uses fabricated data and with weak security measures. Thus, luring an external attacker to attack the system. The administrator observes the system logs and monitors all the traffic involved with the honeypot, as there are no chances that any legitimate user would try to access this system.

The location of honeypots in the network play a crucial role in trapping the attacker. Honeypots can be located outside the external firewall or within the DeMilitarized Zone (DMZ) or within the internal firewall as shown in the figure 3. The location of honeypot can lure attackers of varying expertise. For instance, when placed within the DMZ, a clever attacker who would not be deceived by the honeypot outside the firewall falls into the trap of believing that this honeypot is the real server within the internal network breaking past the external firewall. Honeypots incorporated alongside IDS can help in passive monitoring without the worry of severity of attack, thus extending the scope of attack identification by the IDS.

II. Modern IDS

There are many modern Intrusion Detection Systems, which are implemented across wide domains such as Cyber Physical systems, Supervisory control and data acquisition systems (SCADA) and much more. However, the basic functionality remains the same, that is to detect and respond to the intruder attacks. These designs also vary in aspects of the underlying architecture it is built upon and also on the technologies implemented alongside to assist the inrusion detections. In this paper, IDS implemented on the SDN architectures and IDS implemented using Honeypots in their design are discussed.

A. IDS implemented on SDN architecture

The security mechanisms implemented in the traditional networking architectures will not provide all aspects of security to the SDNs as the architecture varies in the fundamental structure of connecting data and control planes. Similarly, the traditional IDS lack the knowledge of the architecture, hence leaves many open holes in the design for the attacker to exploit. Considering this weakness, there are some modern Intrusion detection systems designed specifically for the SDN architectures implemented across wide domains. In this survey, three such IDSs are discussed, focusing on Modbus/TCP oriented attacks in ICS [8], DDoS detection using deep learning approach [9] and malware detection using fingerprint matching [10] respectively.

1) 2-Level IDS for ICS networks: This paper [8] implements a two level IDS on SDN architecture focused mainly on detecting and responding to Modbus/TCP oriented attacks in Industrial Control Systems (ICS). ICS communicates, between master server and the programmable logic controllers (PLC) located in different geographical locations, in real time by sending the data in the format specified by the application layer level protocol Modbus, that eventually is encapsulated as the payload in TCP segments.

The first level involves whitelisting of the incoming packets at the switches. The switches are programmed as packet processors using the programming language P4, which is independent of the protocols thus flexible to include more protocols in the future. The packets are whitelisted based on both the flow and the modbus information. The flow here indicates the sequence of TCP packets, identified by the identical protocol headers and the Modbus information here indicates the function code and the message length fields in the Modbus data format. The incoming packet is checked for a permissible entry in both the flow and modbus tables at the switches. When both the information matches with valid entries in both the tables, the packets are forwarded to the corresponding destination port. Else, they are rerouted to the security engine in the level 2, a dedicated host sytem which is part of the network. The controller is responsible for both prepopulating the tables at the switches before the installation of the IDS and modifying the table entries adapting to the network monitored by the IDS when active.

The setup of the IDS, as shown in the figure 4, shows the security engine, controller, master server (MTU), PLCs, and the switches. The data communication between the MTU and the PLC is aided by the forwarding of packets at the switches. The second level of IDS involves packet analysis by the security engine. The security engine runs a deep packet inspector and the network security monitor Bro. The packets with the table-miss at the switches are received by the security engine and analysed for attacks. The legitimate traffic which were not prepopulated are identified and sent to the controller which then instructs the switches to update their look-up tables. On the other side, packets identified malicious by Bro and that exceeds a counter threshold limit is sent to the controller which then instructs the corresponding switches to block this traffic.

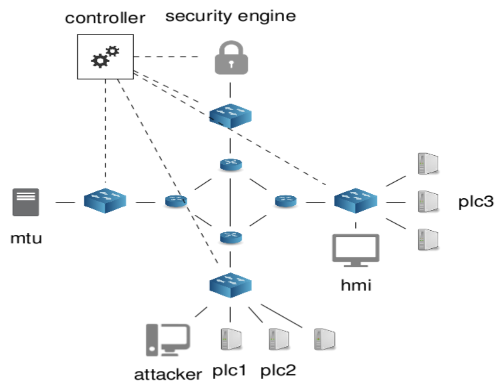


Fig. 4. 2-level IDS setup [source: [8]]

Some of the possible attack scenarios, detected by the 2 level IDS are

- **Function Code Scan attack:** Modbus is the serial communication protocol with the data format that includes a function code to specify the mode of the communication. The protocol is designed in such a way that when an incoming Modbus request contains an unsupported function code, the server in return sends an exception code. By monitoring the exception codes, the attacker can identify supported function codes which can be used to send malicious data disrupting the ICS functioning. This is detected by the IDS in the security engine and the corresponding flow exceeding the counter threshold is blocked.
- **SYN flooding attack:** When an attacker performs SYN flooding attack, the corresponding flow information will not be prepopulated at the look-up tables. This is because the prepopulation occurs either manually or by machine learning approaches using legitimate traffic test data flows. Thus, these packets are rerouted to the security engine which detects the malicious nature of packets and ends up blocking it.
- **DoS attack:** In the ICS, there are some function code messages which when used by an attacker can enact a Denial of Service attack. For instance, the function code ”Force Listen Mode” [8] can instruct the master server to enter listen mode, which disables the responding features of the server. These function codes will not be prepopulated in the modbus tables, thus they are rerouted to the security engine. The security engine allows this packet only when originated at the PLC which are authorized and trusted to not be compromised.

This design of IDS is tested on the Mininet Emulator and found to reduce false positives to great extent.

2) Deep Learning based DDoS detection: This paper [9] implements a Stacked AutoEncoder (SAE) based deep learning approach to detect Distributed Denial of Service attacks from the normal traffic, analysing the features extracted from the packet headers. SAE is an unsupervised deep learning technique where many sparse autoencoders are stacked together such that output of an autoencoder is fed to the other in a sequential basis. The individual sparse autoencoders are capable of adjusting the edge weight of the underlying neural network framework from the backpropagation of errors. This approach focuses on detecting attacks on both the data plane, like packet flooding attacks and the control plane, where the entire SDN collapses when the central control point is compromised.

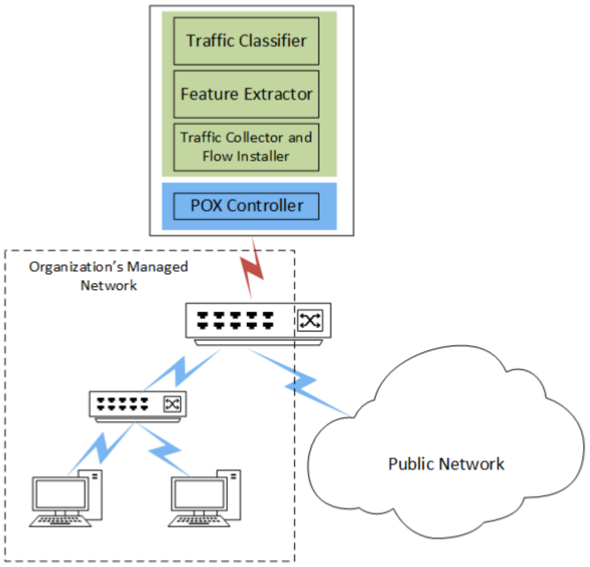

Traffic Classifier
Feature Extractor
Traffic Collector and Flow Installer
POX Controller
Organization's Managed Network
Public Network

Fig. 5. DDoS detection as a Network application [source: [9]]

The main difference of this approach from the 2-level IDS is that the detection module of the IDS is implemented directly on the controller, as shown in the figure 5, while the latter has a dedicated host system as security engine. The DDoS detection system is implemented as a network application in the POX controller. The functioning of the IDS is similar to the previous approach where the packets that don't have an entry in the flow table are rerouted to the controller, however the flow here deals with TCP, UDP and ICMP. The detection application compromises of the following modules:

Traffic Collector and Flow Installer: Once the packets are received at the controller, it distinguishes packets that are rerouted and the packets that are destined for the controller. The flow of the rerouted packets are analysed to find symmetric flows, that is communication between both end points (source to destination and viceversa). If found, the flow is added to the table at the switches. Else, appended to the list of flows which is used by the other modules.

Feature Extractor: This module extracts features from the packet headers, like number of incoming TCP flows; Bytes per incoming flow [9],for the SAE. There are however some features that were computed rather than just looking for the protocol header fields, like Entropy of TTL values for incoming TCP flows.

Traffic Classifier: The features extracted in the previous module is used as test dataset for the SAE, where the traffic is classified into one of the eight classes out of which one represents the normal traffic. The other classes are more specific in identifying the DoS attack based on TCP/UDP/ICMP/combination of one or other. The SAE was trained using training dataset generated in a wireless home network which is a mix of malicious and normal traffic.

The results were verified by calculating performance metrics such as Accuracy, Precision, Recall, F-measure and ROC using a confusion matrix of traffic classes, and showed that the false positives were reduced.

3) Malware Detection using Statistical Fingerprints: This paper [10] designs a malware detection IDS that collects information from the SDN architecture to label the traffic as either malware or normal. The event based controller Ryu interacts with the application Stats Manager, which helps in classifying traffic and building the flow table at the switches. The events that trigger the interaction between the application and Ryu are as shown in figure 6. Random Forest classifier, an ensemble of classifiers approach that subsamples the data set and passes as inputs into all classifier components and outputs the result of the majority vote, is implemented at the controller to classify the traffic from the features extracted from the flow.

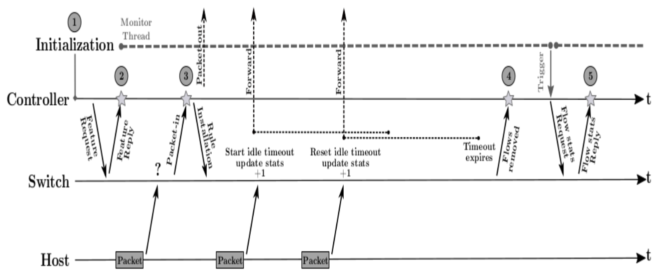


Fig. 6. Stats Manager application interaction with the controller [source: [10]]

The statistics of the flow, which are collected both synchronously and asynchronously by the stats manager application, along with the features extracted and derived from the flows are used to update the flow table. The training and test datasets are populated using the active and ended flows seen by the controller. These datasets are then used for training a classifier model which later is used for actual classification of the test dataset. The flows used to generate datasets comprise of wide range of attack data, some of which is generated by botnets such as Asprox, Cutwail and trojans such as ZeroAccess, TBot.

The IDS is emulated using Mininet and found to classify more true positive results and less false positive results. The accuracy level of the implemented IDS was shown to be in the range of 88 to 97%. Table I lists out the benefits, drawbacks

TABLE I
IDS IMPLEMENTED ON SDN

Publication	Goals	Benefits & Drawbacks	Future Suggestions
2-Level IDS for ICS	To detect Modbus function code and message length exploits and to detect TCP based flow attacks (such as Function code scan attack, SYN flooding attack and so on)	Since only the suspected traffic is analyzed, there is less load on the analysis engine. However, the payload can only be analysed when it comes unencrypted. Thus, cannot be used in the ICS where the data is encrypted unless, the security engine has the corresponding decryption algorithms & keys. Moreover, if the prepopulation of the tables misidentifies attack flows as legitimate, then the whole IDS malfunctions.	The approach could be extended for detection of other protocol flooding attacks, like that of ping flood attack where the ICMP echo request overwhelms the server. And also, DoS of the IDS itself can be considered by implementing mechanisms to drop packets of such random flow nature at the infrastructure level.
Deep learning based IDS	To detect DDoS attacks on both data and control planes in the SDN architecture alongside classifying the traffic into specific classes	The classification of the traffic helps in designing mitigation measures catering to a specific class. However, the detection engine is colocated on the controller, thus DoS against the IDS crashes the entire control plane.	The controller and the DDoS detection application can be separately located so that the chances of controller compromise reduces.
Malware Detection IDS	To detect malwares using the Random Forest classifier which classifies the traffic as malware/normal based on the training and test datasets generated using the active and ended flows observed at the Ryu controller	The traffic collected to generate datasets span from wide range of malware data, thus the efficiency of the classifier output is high. However, the IDS is implemented at the controller with the help of a network application, thus controller load increases.	The IDS could be compared with other designs implemented using other classifiers to show the strength of the IDS design. On contrary, there is a possibility to find better classifier for the implemented design. The design could also be modified by reducing the workload on the controller to increase its efficiency.

and suggestions for the three IDSs based on SDN architecture discussed so far.

B. IDS implemented using Honeypot

Most IDS implement passive monitoring to learn about the attacker's approach, which helps in understanding the vulnerabilities exploited by the attacker and also taking the apt countermeasures. However, in doing so the system falls within the attacker's domain, hence a small negligence of precautions can be catastrophic. For instance, if certain important directories are not removed user privileges, then the attacker can easily access those thus defeating the purpose of IDS. An alternate feasible solution is by using Honeypots to trap the attacker, since the honeypots are designed by the administrator only using the fabricated informations. In this survey, three IDSs [11] [12] [13] which use honeypots in their designs are dicussed as follows:

1) 3-Phase Honeypot based IDS: This paper [11] implements an IDS to detect Rogue Access Points (RAP) in wireless networks. The RAPs are classified by [11] into "Improperly Configured AP", "Unauthorized AP", "Phishing AP" and "Compromised AP". In all these classifications, the main focus is the access points which are controlled by the attacker and part of the wireless network where the users can't distinguish its malicious purpose from the rest of the legitimate access points. The IDS is designed to detect both internal and external attackers who control the RAPs.

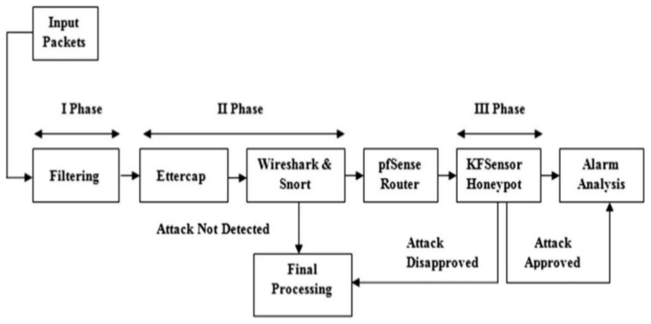


Fig. 7. 3 Phases of Honeypot based IDS [source: [11]]

The IDS functions in 3 phases as shown in the figure 7. In the first phase, MAC addresses are compared with an authorized MAC address whitelist of APs. The APs whose MAC addresses not in the list are identified as RAPs. However improperly configured and compromised APs are not detected in this phase. In the second phase, the APs which are found legitimate by the first phase are analysed further using three security tools listed below:

- Ettercap is used to detect unauthorized IP addresses, hence unauthorized RAP are filtered out here. However, if the attacker poisoned the ARP cache, the legitimate IP address can be spoofed which will bypass this detection. The APs identifed as authorized are only sent to Wireshark and Snort for further analysis.
- Wireshark is used here to analyse the packet flow rate to detect possible attacks such as Man in the Middle attack, where the flow rate of packets decreases because all the

TABLE II
IDS DESIGNED USING HONEYPOT

Publication	Goals	Benefits & Drawbacks	Future Suggestions
3-Phase Honeypot based IDS	To detect Rogue Access Points in wireless networks and to find both internal and external attacks performed via RAPs	The IDS is very efficient in aspects of reducing the traffic analyzed by the Honeypot by filtering out the workload in the previous phases. However, there are high chances that the false negatives are increased because RAP can deceive the Wireshark and Snort in the second phase, by following new attack patterns not identified by Wireshark and Snort.	The IDS could scrutinize the analysis in the second phase to reduce false negatives by introducing additional security mechanisms that can learn to detect new attack patterns (an unsupervised machine learning approach could be integrated).
Honeypot based Proactive IDS	To detect selfish and malicious nodes in VANETs using bait RREQs	The accuracy of detection is improved when compared to other threshold based detections. However, more congestion in the network is introduced by these bait RREQs. Moreover, only energy based selfish nodes are detected in this approach.	The IDS could integrate identifying selfish nodes that are concerned about other resources like CPU utilizations. There can be a dedicated host serving as honeypot, sending baits thus reducing network traffic congestion with respect to the legitimate nodes.
IDS based on Log sequence Clustering of Honeypot	To find Sequential attack patterns targeting the Modbus data using Request Similarity Clustering model	The design explored all possible clustering by varying the cluster threshold from 0 to 1 and later using the best threshold that gives high accuracy to carry out the similarity evaluation. Thus, the design is dynamically adaptible to any attack data. However, the IDS focuses only on Modbus based logs.	The mechanism for TCP flow level analysis could be integrated to find more unexpected access commands.

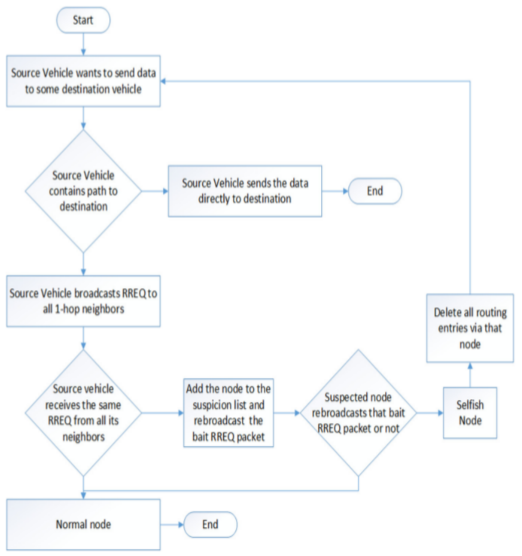
packets from both ends are relayed through the man in the middle RAP and Denial of Service attack, where the packet flow rate is abruptly high.

- Along with the Network sniffer Wireshark, the open source NIDS (Network IDS) Snort is used to generate alerts when certain rules that indicate attacks are met like that of DNS spoofing.

The RAP traffic which are identified as rogue after this phase is then rerouted to the KFSensor Honeypot, the third phase, to trap and analyze the attack patterns. The experimental results are shown that detects attacks like ARP Cache poisoning, DNS spoofing and DoS along with mathematical analysis to substantiate the claim that the false alarm rate is greatly reduced.

2) Honeypot based Proactive IDS: This paper [12] presents a honeypot based proactive IDS to detect selfish nodes in Vehicular Ad-hoc Networks (VANETs). Ad-hoc networks are mainly successful because of the active participation of all nodes in the network without any infrastructure to oversee the communications, unlike Infrastructures like Access points. However, some nodes don't participate in the communications in order to preserve their resources for efficient functioning. This, however affects the network performance in terms of delay and sometimes network breakage. This IDS is proposed to detect such nodes which don't participate in the network communication until their residual energy is greater than 50%, also referred to as "selfish" nodes. This approach is also referred as HSPD, "Honeypot Selfishness Detection Scheme".

In this approach, when a node doesn't find an entry in its routing table for the corresponding destination, it broadcasts a RREQ packet (AODV protocol) to all its neighbours seeking a path to the destination through those nodes. This broadcast is repeated by the neighbour nodes if they don't know any such path either. Finally the node with a valid path to the destination traces back to the broadcast initiator node. There are however some selfish nodes which don't participate in it. The source baits such nodes by sending a bait RREQ to all those neighbours from which it didn't receive RREQ rebroadcast after its broadcast. If the bait RREQ is not forwarded, the source node classifies such neighbours as selfish and deletes all the routing paths through that node, since this could be a malicious node which is just a sniffer.


Start
Source Vehicle wants to send data to some destination vehicle
Source Vehicle contains path to destination
Source Vehicle sends the data directly to destination
End
Source Vehicle broadcasts RREQ to all 1-hop neighbors
Delete all routing entries via that node
Source vehicle receives the same RREQ from all its neighbors
Add the node to the suspicion list and rebroadcast the bait RREQ packet
Suspected node rebroadcasts that bait RREQ packet or not
Selfish Node
Normal node
End
Fig. 8. Flowchart to detect selfish vehicles in proactive IDS [source: [12]]

This bait approach, functioning as shown in figure 8, helps in identifying malicious nodes in VANETs. The network is simulated in NS2 & Sumo and shown to detect selfish nodes with high accuracy when compared to threshold based detections, that looks out for responses to the fake messages.

3) IDS based on Log sequence Clustering of Honeypot: The widely used serial communication protocol in SCADA and ICS networks is Modbus TCP protocol. The protocol in itself doesn't contain security implementations to keep the data transfer simple. Thus there are many IDS focus on detecting attacks on this protocol like the SDN based IDS [8] discussed in the section II-A1. In this paper [13], the log sequences collected at the honeypot are clustered into attack patterns using similarity comparisons, using an unsupervised clustering approach. The log sequences correspond to the commands accessing Modbus data.

The working of this IDS can be categorised into three main steps as follows:

- First, the Modbus data logged by the Conpot Honeypot is preprocessed and sent to the clustering model. The preprocessing involves collecting together log data based on identical packet header fields like identical function codes. This is similar to the ”flow” in SDN architecture. All these data are considered attack data because the setup ensures that no legitimate users access the honeypot either by placing the honeypot in DMZs or external to the network.
- The log sequences are then clustered to find ”SAP” as stated by [13], ”Sequential attacking patterns”. SAP refers to the set of sequentially executed commands that share high similarities with the signatures generated as attack patterns by the clustering model. The clustering model is based on ”Request Similarity evaluation”, that compares two sequences using ”Discrete data type similarity”, that compares similarity of discrete features from the header fields such as Function code, Access section, and ”Numeric data type similarity”, that compares similarity of numeric attributes Word count and Byte size from the header. The representative SAP sequence, from all such clusters formed after agglomerative clustering indicates different attack patterns.
- Once the SAPs are generated, a flow graph representing the sequence of commands in the SAP is developed for easy user understanding.

This design is experimentally tested and shown to detect scanning attacks and DoS attacks against Modbus service with 0% false positive rate. Table II lists out the benefits, drawbacks and suggestions for the three IDSs, that included Honeypots in its design, discussed so far.

III. Conclusion

In this Survey, an overview of Intrusion Detection Systems, Software Defined Networking and Honeypots are discussed. Three modern IDS designed for SDN architectures and three modern IDS designed using Honeypots are compared, discussed and some future suggestions are given.

The main purpose of this paper is for the reader to understand the concepts of IDS and its significance and implementations to strengthen security in various domains such as SCADA, ICS, VANETs and in modern architectures such as Software defined Networking.

IV. Acknowledgements

I sincerely thank Professor Clifford Neuman for his valuable advice in the choice of research papers and his comments on writing a survey paper that greatly improved the survey.

References

[1] Anderson, J. Computer Security Threat Monitoring and Surveillance. Fort Washington, PA: James P. Anderson Co., April 1980.

[2] Stallings, William. Cryptography and Network Security: Principles and Practice. 5th ed., Pearson, 2017.

[3] Bishop, Matt. Computer security: art and science. Addison-Wesley Professional, 2003.

[4] Williams, Amrit (2005-05-02). ”Improve IT Security With Vulnerability Management”.

[5] Pharate, A., Bhat, H., Shilimkar, V., and Mhetre, N. (2015). Classification of Intrusion Detection System. International Journal of Computer Applications, 118(7).

[6] F. Sabahi and A. Movaghar. 2008. Intrusion Detection: A Survey. In Proceedings of the 2008 Third International Conference on Systems and Networks Communications (ICSNC '08). IEEE Computer Society, Washington, DC, USA, 23-26. DOI: https://doi.org/10.1109/ICSNC.2008.44

[7] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. et al. (2008) OpenFlow: Enabling Innovation in Campus Networks. SIGCOMM Comput. Commun. Rev. 38(2): 6974.

[8] Kabasele Ndonda, Gorby, and Ramin Sadre. ”A Two-level Intrusion Detection System for Industrial Control System Networks using P4.” ICS-CSR 2018. 2018.

[9] Niyaz, Quamar, Weiqing Sun, and Ahmad Y. Javaid. ”A deep learning based DDoS detection system in software-defined networking (SDN).” arXiv preprint arXiv:1611.07400 (2016).

[10] Bigotto, Francesco, et al. ”Statistical fingerprint-based IDS in SDN architecture.” Proceedings of the 21st International Symposium on Performance Evaluation of Computer and Telecommunication Systems. Society for Computer Simulation International, 2018.

[11] Agrawal, Neha, and Shashikala Tapaswi. ”The performance analysis of honeypot based intrusion detection system for wireless network.” International Journal of Wireless Information Networks 24.1 (2017): 14-26.

[12] Patel, Priya, and Rutvij Jhaveri. ”A Honeypot Scheme to Detect Selfish Vehicles in Vehicular Ad-hoc Network.” Computing and Network Sustainability. Springer, Singapore, 2017. 389-401.

[13] Wang, Pin-Han, et al. ”An intrusion detection method based on log sequence clustering of honeypot for Modbus TCP protocol.” 2018 IEEE International Conference on Applied System Invention (ICASI). IEEE, 2018.