Snort NIDS Evolution & Expansion

Nicholas Noto
University of California, Santa Cruz
nnoto@ucsc.edu

***Abstract*—Since its initial iteration in 1998, the open-source intrusion detection system Snort has seen considerable expansion. Beyond the addition of new rule-sets, many have attempted to make significant changes and additions to the software based on academic research, experimentation, and varied necessitated requirements. From hybrid designs that use both anomaly and signature-based detection, to improved attack classification procedures, and even including instances of Snort used in tandem with a genetic algorithm, the attempted techniques tend to differ greatly and are quite remarkable. Certain additions have even been made on occasion for specific vulnerabilities that have come to light in the computer security community. These attempts to strengthen Snort are important not only because Snort itself is widely used in many integral systems worldwide, but also because they are valiant efforts towards progress in the security field as a whole.**

I. Introduction

Network intrusion detection systems (NIDS) monitor the traffic at certain strategic points on a network in an attempt to detect malicious activity. NIDS typically use either signature-based or anomaly-based detection to identify such threats, with both techniques using distributed rule-set configurations. A signature-based intrusion detection system will analyze the traffic on the network, searching for specific patterns that match with known signatures in a library[1]. An anomaly-based intrusion detection system will search for suspicious activity on the system with the use of heuristics to identify unauthorized system access and properly differentiate malicious intrusion from normal operation[2]. The developers of the network intrusion detection system usually provide default rule-sets and updates and allow the users to edit the rule-sets themselves as well. One of the most popular NIDS is called Snort, which utilizes a signature-based detection scheme by default. It was first released in 1998 by Martin Roesch as an open-source project and has since garnered a wide and diverse user-base and development community. In most cases, Snort works sufficiently using the rule-sets and default configurations provided by Roesch's company Sourcefire, but many individuals and teams have sought to expand upon and improve the detection and defense mechanisms over the years with varying degrees of success. The attempts at NIDS evolution, whether they succeeded or not, represent valuable resources from which to draw information and are worthy of further exploration.

II. Vanilla Snort

Snort comes loaded with three main modes of operation by default. The packet sniffer and packet logger modes are relatively basic, with the former reading network packets before displaying them to console and the latter simply logging those packets to disk. The third mode of operation is known as network intrusion detection mode, which is where the bulk of the detection capabilities come into play[3]. Using the provided rule configurations as a means for identification, pattern matching, and signature analysis, the network intrusion detection mode is able to detect a variety of common attacks such as server message block probing, CGI attacks, stealth port scans, stack smashing, OS fingerprinting attempts and more[4]. Despite the frequent rule updates and relative simplicity of design, Snort is not a perfect intrusion detection system and its default capabilities suffer from drawbacks and limitations. One such example involves the high false positive rates exhibited when attempting to identify threats. Researchers have found that NIDS in general suffer from high false positive rates and unfortunately Snort is no different. This is problematic because it can lead to what researchers refer to as "squealing," where an individual specifically crafts packets to match attack signatures. The spoofed packets can then be used to generate large numbers of false positives in order to conceal real attacks and avoid detection[5]. Another drawback of the default version of Snort is the increasingly large size of the distributed rule-sets. Having progressively larger rule-sets can lead to a decrease in performance and loss of packets in many situations[6]. In addition, the sheer amount of information being collected and communicated to Snort administrators during normal operation represents a problem as well, since it can be overwhelming to properly analyze and may result in vital information being overlooked[16]. Fortunately, members of the Snort community are always at work on improvements and fixes that can be made by both updating the existing Snort model and expanding upon it.

III. Snort Improvement

Endeavors towards improving and diversifying Snort have taken many shapes and forms. Examples include extending Snort to facilitate anomaly detection, algorithms used strategically alongside Snort, implementations of Snort in hardware, upgrading its visualization capabilities, refactoring Snort's classification techniques, and even purely adding to or enhancing the established rules.

A. Anomaly-Based Hybridization

Choosing whether to make a NIDS signature-based or anomaly-based can be tricky, since both possibilities have strong qualities as well as issues. In an attempt to improve upon Snort's signature-based design, some researchers have

theorized, created, and tested the addition of an anomaly-based pre-processor in the form of a plugin[7][8]. This type of addition could potentially offer Snort the advantages of both NIDS designs, with careful consideration taken to avoid the disadvantages of each. The hybrid version of Snort would benefit from an anomaly-based system's capabilities to detect unknown attacks that are not yet in the known signature library, as well as the reliability benefits a signature-based NIDS provides.

A group of researchers from the University of Almeria that took this approach did so with the help of a MySQL database. The database was designed to be frequently and continuously updated, and housed the system configuration information, statistical data regarding network traffic, and any anomalies that the system ended up detecting. Their anomaly detection pre-processor was outfitted with a training mode for recording normal operation information to the database, and a detection mode that used the data stored when training as a means of detecting anomalies. Their testing and subsequent results revealed that their efforts paid off for the most part, with a decrease in overall false positives compared to a normal Snort implementation, along with consistently improving performance with extended training mode usage[7].

Another team from Amrita University adopted a similar strategy, with their version using the built-in packet sniffer to log real time data. The data was then used as training information for a support vector machine[1] and its associated machine-learning algorithm for proper classification of normal operation and anomalies within the data set. Their experimental results showed that their implementation of a hybrid NIDS was able to detect anomalies prior to the default detection engine, resulting in a higher detection rate when up against threats excluded from Snort's supplied threat signature library[8].

Both team's approaches made significant use of system training protocols that effectively allowed Snort to classify network traffic. In both cases the teams had more desirable results with longer training periods. This was likely because longer training equated to more network data, leading to a more accurate model of normal operation and more easily distinguishable attack-related outliers.

B. Genetic Algorithm Assistance

A genetic algorithm is a heuristic that attempts to imitate the process of natural selection and mutation and is used for generating solutions for optimization and search-based problems[12]. In the context of NIDS, the use of a genetic algorithm within Snort shares similarities with attempts at hybridization because both offer a mechanism by which Snort's detection abilities evolve while the software is in use. However, the use of a genetic algorithm provides a resource and tool for growth for the established signature-based detection engine rather than pre-processing the data using anomaly-based techniques as a preliminary means for identification.

One particular team of students performing research at Madras Institute of Technology managed to employ network traffic clustering[2] techniques with the help of a genetic algorithm for cluster optimization and analysis that resulted in the successful derivation of updated rule-sets for Snort. The team utilized a parallel evolutionary computing[3] approach that used a hill climbing technique[4] along with the genetic algorithm to evaluate the fitness of the clusters iteratively. The fitness levels of the clusters were based on intra-cluster distance, inter-cluster distance, and the density of the clusters. The iterative process continued until the clusters no longer saw substantial increases in their fitness levels and were considered fully optimized. Once the clusters were properly formed, they were then examined for classification and rule construction. The largest resultant cluster was assumed to represent normal network traffic, and the rest of the clusters were classified as attacks. Their experimental results demonstrated effective rule generation for Snort as well as a reduction in the overall false positive rate. The new rules were more efficient in general than those included in a vanilla version of Snort and provided protection against attacks that would have otherwise gone unseen[13].

A pair of researchers from the NRI Institute of Science & Technology utilized a genetic algorithm as well, though their attempt focused primarily upon decreasing the necessary CPU and RAM usage of the default detection engine. They were able to accomplish their goal by using the genetic algorithm to analyze network data and construct functionality classifications for every rule, thus making the system faster at detecting intrusions. The functionality classifications also enabled safe reduction of Snort's rule-sets based on similarities and redundancy in the existing rules. Their results clearly demonstrated the effectiveness of their classification model, which exhibited much lower CPU and RAM usage than when using a vanilla copy of Snort, despite barely improving the actual detection rate[15].

In the end, the two teams were able to demonstrate different benefits of genetic algorithms in conjunction with Snort. Since the two approaches focus on entirely separate facets of Snort improvement, it might even be possible and advantageous to combine the approaches. The rule generation capabilities achieved by the Madras team and the redundant rule reduction techniques of the NRI team might be well-suited for each other, allowing for a more efficient NIDS in terms of both detection and resource usage.

C. Visualization Systems

Attempts at modification and customization of the Snort attack signature database can be both complicated and difficult, and writing additional detection mechanisms can be problematic as well. As a result, some researchers have chosen to forgo

[1] *Support vector machines are supervised learning models that analyze data and recognize patterns*[9].

[2] *Clustering refers to the act of grouping sets of objects in such a way that objects in the same cluster are more similar to each other than to those in other clusters*[14].

[3] *Evolutionary computing is related to the study of artificial intelligence and typically utilizes iterative progress to achieve continuous and combinatorial optimization*[10].

[4] *Hill climbing uses an iterative algorithm that continually improves upon an arbitrary solution to a problem by incrementally altering an individual component of the solution*[11].

either route and create a log visualization system to provide assistance instead. The purpose of the visualization addition would ideally allow users of the network intrusion detection system to engage in real-time analysis of the attacks taking place, the rules aiding in attack detection, and the subsequent alerts produced. If implemented properly, such a visualization system could offer improved attack visibility, convenience, and decreased instances of false positives[16][17].

Hideki Koike and Kazuhiro Ohno of The University of Electro-Communications saw worth in NIDS log visualization and chose to focus their efforts on a useful visualization utility for Snort appropriately dubbed "SnortView." SnortView allows for the simultaneous visualization of the time, type, source, destination, and details of each and every network access logged by Snort. In addition, Koike and Ohno included analysis based on log comparisons from many different environments with the intent of providing a rough guideline for how to discover and interpret possible false alarms. Based on their testing, they found that alarms which appeared consecutively, numerously, conflicted with provided services, and/or referenced other networks tended to be false alarms. As a warning, they also mentioned that their classification method has limitations and only offers a tool for NIDS administrators that can help reduce their cognitive load. SnortView doesn't visualize packet header information either, leaving the system vulnerable to denial of service and backdoor attacks that might be coded into the header. Despite its problems, the utility offered undeniable improvements towards data delivery design and progress oriented towards streamlining the NIDS administrator experience[16].

A team lead by Xiaojin Hong of the Beijing Institute of Technology sought to create a similar visualization tool for identifying relationships between rules and triggered alerts, but did so with the use of a treemap.[5] The use of a treemap for displaying hierarchical data was helpful for Hong's team because it allowed them to visualize a large expanse of information effectively within a confined space. Their tool VisSRA featured customizable treemap layouts so the user could include specific information that satisfied their needs without additional clutter. Though their tool allowed for more information visualization than SnortView, Hong's team conceded that VisSRA didn't offer much help regarding false positives and conveyed their intent on improving upon their design in future work[17].

SnortView and VisSRA represent another example of related utilities whose limitations could possibly be remedied by a joint project between the two teams. VisSRA has a better visualization model that's both more dynamic and more comprehensive than SnortView's, but the team behind SnortView made more progress from an analytical perspective, offering insight for administrators on visualization patterns and their potential identification opportunities.

[5]*Treemaps are information visualization structures for displaying tree-structured hierarchical data using nested rectangles to differentiate between each level*[18].

D. Hardware Integration

Even with visualization advantages, Snort and many other NIDS are too slow to sufficiently recognize and identify all possible network traffic in large systems that require high processing rates. The lack of comprehensive analysis is typically because the incoming data rate is often too fast for the intrusion detection systems to keep up with[19]. In an effort to optimize the efficiency of vanilla Snort installs, some researchers have designed procedures for implementing Snort's basic features and functionality in the form of specialized hardware.

One such team composed of members from the Islamic Azad University and Shahed University designed a Verilog-based processor that provided two of Snort's basic instructions in hardware. Their goal with the project was to provide a means for hardware acceleration when network traffic warranted it. In order to properly recreate the instructions in hardware, related finite state machines were used to model the instructions along with a central coordinating block for synchronizing execution. The design specifically targeted two of Snort's most common instructions, $byte\_test$ and $byte\_jump$, in order to derive the maximum benefit from their hardware model. The hardware integration of $byte\_test$ and $byte\_jump$ alone allowed the team to significantly improve the performance and reliability of Snort when up against high data rates, maintaining the flexibility of new rule additions in software form as well[21].

Rather than creating hardware that handled Snort instructions, a researcher from the Georgia Institute of Technology developed a reconfigurable string matching processor for Snort using field-programmable gate arrays(FPGA)[6]. Their goal was to enable users to fit an entire Snort rule-set on a single FPGA chip that would take care of all necessary signature pattern matching and allow the Snort system to focus on other tasks, creating a more robust system capable of handling higher incoming data rates. The process of pattern matching is typically one of the most computationally intensive components of a Signature-based NIDS such as Snort, so circumventing a software implementation would be very helpful towards increasing performance. The strategic use of an FPGA chip helped greatly towards the researcher's goals because FPGA chips have a computation cycle count that stays constant when increasing the number of possible patterns to be processed, unlike the detrimental scaling of Snort's default pattern matching implementation. Their eventual outcome could handle high data rates sufficiently, and offered support for complex pattern requirements and approximate pattern matching. The addition of approximate pattern matching allowed for detection of unknown attack signature variations using non-deterministic finite automata. Despite providing adequate speed and detection mechanisms for new attacks, it seemed as though the researcher's implementation suffered from difficult update procedures, making it a hassle to add updated rule-sets to the FPGA chips on a regular basis[22].

Both attempts at hardware integration represented useful additions to Snort facilities, though integration of instructions managed to preserve the flexibility of a normal Snort im-

[6]*Field-programmable gate arrays are integrated circuits intended to be configured by the customer after purchase*[20].

plementation while also offering the data rate capabilities it intended. With the help of hardware advancements and clever design, future researchers may find it worthwhile to integrate various other Snort instructions into hardware as well.

IV. Conclusion

The Snort network intrusion detection system is still a work in progress. The nature of an open-source project is that it will continue growing and expanding in different directions for varied purposes that the general computer security community deem necessary. However, it is worth researching and learning from the many advances and worthwhile improvements that have been made over the last sixteen years that have contributed towards pushing the project forward. Some of the expansions detailed above could have had greater successes and managed to avoid certain limitations had the researchers made increased use of their peer's work as potential stepping stones. For the sake of efficiency and progress, the vast majority of researchers attempting further expansion in the future would be wise to rely on the successes and failures of the past as a foundation to build upon.

References

[1] Tejvir Kaur and Sanmeet Kaur. *Comparative Analysis of Anomaly Based and Signature Based Intrusion Detection Systems Using PHAD and Snort.*

[2] *Anomaly-based intrusion detection system.* Wikipedia. http://en.wikipedia.org/wiki/Anomaly-based_intrusion_detection_system

[3] Sourcefire Inc., Martin Roesch, and Chris Green. *SNORT Users Manual* Available at: http://manual.snort.org/

[4] Martin Roesch. 1999. *Snort - Lightweight Intrusion Detection for Networks.* In Proceedings of the 13th USENIX conference on System administration (LISA '99). USENIX Association, Berkeley, CA, USA, 229-238.

[5] Samuel Patton, William Yurcik, and David Doss. 2001. *An Achilles heel in signature-based IDS: Squealing false positives in SNORT.* Proceedings of RAID 2001.

[6] Soumya Sen. 2006. *Performance Characterization & Improvement of Snort as an IDS.* Bell Labs Report.

[7] J. Gmez et al. 2009. *Design of a Snort-Based Hybrid Intrusion Detection System.* Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living. Springer Berlin Heidelberg, 515-522.

[8] Annie George, Prabaharan Poornachandran, and M. Ramachandra Kaimal. 2012. *adsvm: Pre-processor Plug-in Using Support Vector Machine Algorithm For Snort.* Proceedings of the First International Conference on Security of Internet of Things. ACM.

[9] *Support Vector Machine.* Wikipedia. http://en.wikipedia.org/wiki/Support_vector_machine

[10] *Evolutionary Computation.* Wikipedia. http://en.wikipedia.org/wiki/Evolutionary_computation

[11] *Hill Climbing.* Wikipedia. http://en.wikipedia.org/wiki/Hill_climbing

[12] K.F. Man, K.S. Tang, and S. Kwong. 1996. *Genetic algorithms: concepts and applications [in engineering design]*, Industrial Electronics, IEEE Transactions on vol.43, no.5, pp.519,534.

[13] Raghavan Muthuregunathan et al. 2009. *Efficient Snort Rule Generation Using Evolutionary Computing for Network Intrusion Detection.* Computational Intelligence, Communication Systems and Networks. CICSYN'09. First International Conference on (pp. 336-341). IEEE.

[14] *Cluster Analysis.* Wikipedia. http://en.wikipedia.org/wiki/Cluster_analysis

[15] Mit H. Dave and Samidha Dwivedi Sharma. 2014. *Improved Algorithm for Intrusion Detection Using Genetic Algorithm and SNORT.* International Journal of Emerging Technology and Advanced Engineering. Volume 4. Issue 8.

[16] Hideki Koike and Kazuhiro Ohno. 2004. *SnortView: visualization system of snort logs.* Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security. ACM.

[17] Xiaojin Hong et al. 2012. *VisSRA: Visualizing Snort Rules and Alerts.* Computational Intelligence and Communication Networks (CICN), 2012 Fourth International Conference on. IEEE.

[18] *Treemapping.* Wikipedia. http://en.wikipedia.org/wiki/Treemapping

[19] Monther Aldwairi. 2006. *Hardware Efficient Pattern Matching Algorithms and Architectures for Fast Intrusion Detection.*

[20] *Field-Programmable Gate Array.* Wikipedia. http://en.wikipedia.org/wiki/Field-programmable_gate_array

[21] Ehsan Azimi, M.B. Ghaznavi-Ghoushchi, and Amir Masoud Rahmani. 2009. *Implementation of simple SNORT processor for efficient Intrusion Detection systems.* Intelligent Computing and Intelligent Systems. ICIS 2009. IEEE International Conference on. Vol. 3. IEEE.

[22] Christopher R. Clark. 2004. *Design of efficient fpga circuits for matching complex patterns in network intrusion detection systems.* Georgia Institute of Technology.