WonderWall:
A Machine-Learned Network Filtering Engine

Kruesit Upatising    Christian Barcenas    Yesha Ouyang    Scott Collins
Dr. Jonathan M. Smith

ABSTRACT

Wonderwall is a proof-of-concept network filtering engine utilizing machine learning to identify malicious network packets. Trained using modern profile-generated datasets, Wonderwall aims to augment human reaction and rules-based IDSs to respond to attacks in real-time. It is designed to be integrated into a virtual network such as one built with OpenFlow to scalably handle malicious network attacks.

1. INTRODUCTION

Computer networks are traditionally defended against malicious activity by proactive network administrators using active traffic management techniques. For example, a network administrator may react to an active DDoS attack by manually modifying firewall rules and routing policies. A web system administrator may respond to repeated SQL injection attempts by installing a web application firewall (WAF) that matches and rejects HTTP requests containing parameters containing known strings used in SQL injection. However, tactics such as these have proven inadequate in the face of increasingly complex and adaptive cyber-attacks which drive up the time-to-response of an incident as well as required to respond to that incident.

Humans are ill-suited for the tedious task of differentiating between malicious and normal network traffic; it is difficult to manually generate a rules-based model to represent all possible attacks against a network. Further, rules-based IDS’s after deployment are often seen as too inflexible and unable to cope with the rapidly changing nature of network attacks. The security community at large is moving away from static rules- and signature-based attack classification.

We developed a network filtering engine which utilizes machine learning to identify malicious network traffic. Our model is trained on different datasets for different attack scenarios. This setup allows the classifier to be updated easily when a new attack appears. A packet to be classified is passed to an ensemble of classifiers. These classifiers output binary classifications and confidence scores based on their belief that an individual packet or window of packets represent a certain kind of network attack.

Our filtering engine integrates easily with any OpenFlow-capable software defined network. When an attack is identified by a classifier, OpenFlow can be used to instruct the network switch to block the individual offending packet or to install routing rules to block the source of malicious traffic entirely. To
simulate a deployment of Wonderwall on a network we used Mininet for network virtualization in software.

2. RELATED WORK

The concept of applying machine learning to intrusion detection and network security has been an active area of research for quite some time. Related work to Wonderwall falls into two categories. The first is the basic application of classifiers to identify malicious packets, and if possible, distinguish amongst different types of intrusive traffic. The second is in the domain of intrusion prevention, using machine learning to automatically generate network rules and respond to in-progress attacks.

The intrusion detection problem has been approached using numerous different types of machine learning techniques. These include decision trees, K-nearest neighbor, support vector machines, neural nets, naïve Bayes, genetic algorithms, and various ensemble classifiers such as random forests and boosted trees. Much of this work has concentrated on the few publicly available intrusion detection datasets, such as the 1999 KDD Cup and the 1998 and 1999 DARPA evaluation datasets. Between 2000 and 2007, out of 55 papers surveyed, 30 used the 1999 KDD Cup, and 18 used the 1998 DARPA dataset. Few studies used private or self-generated datasets, indicating that these public datasets served as a common standard.

However, more recent work has called into question the accuracy and ability of these DARPA datasets to reflect modern real-world attacks. Sommer and Paxson (2010) contend that “the most significant challenge an evaluation [of an intrusion detection system] faces is the lack of appropriate public datasets.” As a result of this and other challenges, machine learning approaches suffer from low precision; that is, they generate too many false positives to be realistically deployed in a real-world setting. To combat this, Wonderwall is trained on a more modern IDS dataset that features more realistic network traffic and more diverse intrusion scenarios.


The intrusion prevention problem has received comparatively less attention. Some early attempts included the use of genetic algorithms and decision trees to classify and filter connections within a network\textsuperscript{7}, and the use of data mining techniques to extract features that can be transformed into IDS rules\textsuperscript{8}. More recently, researchers at the University of Pennsylvania created an experimental system called GOALIE that aims to automatically detect network intrusions and generate firewall rules in real-time using decision trees.\textsuperscript{9}

Wonderwall aims to solve a similar problem space by creating an all-in-one solution. The main difference and improvement between Wonderwall and GOALIE is that Wonderwall aims to be a more modern, adaptable solution that may be used more easily in the real world. By building Wonderwall as a component of a software-defined network, network administrators can integrate it seamlessly into any OpenFlow supported system.

3. **APPROACH**

3.1 **Data**

The performance of a machine-learned classifier is highly dependent on dataset upon which it is trained. For our purposes, a dataset must be up-to-date, labeled, representative of real-world cyber attacks, and of sufficient size. To that end, finding a dataset of network tcp/ip packets that meets the criteria proved a challenging task. Due to security concerns, packets of actual real-world attacks are seldom, if ever, publicly released. We resorted to alternatives, data from simulated attacks or competitions.

The first dataset we used was processed tcpdump packet captures from simulating a U.S. Air Force LAN, peppered with multiple attacks.\textsuperscript{10} The dataset was used for a 1999 KDD intrusion detection contest, where the goal was to train a predictive model based on this data. Although the data at first appeared promising, inadequacies were later revealed after careful examination. While many types of attacks were represented in the dataset, the majority were distributed denial of service (DDOS) attacks, which were relatively simple to detect. Once these attacks were filtered out, performance faltered. Furthermore, many of the features were derived and stream-based, such as number of failed login attempts and percent of connections that have “SYN” errors. These were difficult to extract in real-time and proved unsuitable for our project.


The dataset we finally settled on was from University of New Brunswick Intrusion Detection Evaluation dataset. A systematic approach was taken to generate these packets from profiles that mimicked both real-world normal network traffic and cases of malicious behavior. Data isn’t skewed towards DDOS attacks as care was taken to ensure attack type diversity. The dataset as a whole was intended intrusion detection testing, evaluation, and comparison, which aligned with our goals. The dataset also contained the raw packet captures, which we used in our simulated network.

In total, there were 2,071,657 data points, of which about 4% were malicious. Data was initially separated and categorized by day, but aggregating them was simpler to work with. Although the pcaps contain the data payload of each packet, these were dropped for our purposes, leaving 13 features to be used for classification.

### 3.2 Snort

Snort represents the status quo that WonderWall aims to surpass. Snort is a simple rules-based IDS. It relies on a repository of signatures that apply actions to matched packets, such as “alert,” “log,” or “drop.” Its rigidity poses several problems.

First, a hacker may deduce what signatures a host is using in its Snort configuration by looking through standard public repositories. The hacker may then attack the host by making his traffic not match any of the signatures. Once he has gained root access to the host, he may change the Snort configuration to make continuation of the attack easier.

WonderWall introduces a barrier to entry that would prevent such a simple maneuver. WonderWall would indeed rely on a public repository, but of massive datasets, not concise, human-readable signatures. This provides a hefty level of obfuscation, as a hacker would need comparable resources to the owners of the host to learn from this data. Perhaps in the future even the most complex of attack patterns could be learned using a paradigm such as deep learning, which requires prohibitive computational resources.

Second, when retrofitted to be a classifier by interpreting signature matches as “attack” classifications, Snort configured with the standard EmergingThreats.net open rules set simply performed worse than WonderWall, with an F₁ score in the 30-40% range versus WonderWall’s 90-100% range. This indicates that a network operator must add more value to Snort than to WonderWall, needing to create a more extensive rules set and having to deal with more unclassified attacks on the fly.

Overall, Snort’s simple rules-based logic is no match for the multi-staged complexity of modern attacks. A network operator must add significant intelligence on top of the stock snort configuration to have an effective IDS. With WonderWall, our preliminary results show great promise for the expressiveness of a repository of datasets as opposed to rules. There is a closer match between the evolution of network

---

http://www.unb.ca/research/iscx/dataset/iscx-IDS-dataset.html
attacks and the emergence of new ML paradigms to meet them than the stagnant rules-based system poses. Due to the overwhelming hurdles to obtaining network captures of actual attacks such as confidentiality and the difficulty in labeling the data, WonderWall will rely on a community of research institutions contributing the latest synthetic attack data sets to the repository, choreographed on testbed networks, like the dataset we used from the University of New Brunswick (UNB). However, as UNB expresses with confidence, this is the direction in which the security community ought to move, as it has already begun in the case of UNB.

4. MACHINE LEARNING CLASSIFICATION

The goal of our machine learning classifier to maximize recall and precision in detecting malicious packets. The run time performance, in terms of how quickly the prediction is calculated once the model has been trained, is also of importance as millions of packets will be classified.

As aforementioned 13 features were extracted, and the resulting data was converted and compiled into a 2,071,657 by 13 matrix of integers. IP addresses and categorical data, such as TCP flags and protocol, were mapped to integers, as they were unusable in their raw string format. As there are only 13 features, each distinct, feature selection is unnecessary. This data is then used for classification training.

The techniques and models for machine learning are many. For the purpose of this project, only supervised-learning binary classification methods are applied. Unsupervised clustering algorithms were attempted, but the endeavor was unfruitful. Of the applicable machine learning models, four were used: truncated decision trees, Adaboost, Support Vector Machines (SVM) and Naive Bayes.

A decision tree is a simple classification model where internal nodes represent decisions, i.e. comparison of features and some fixed value, and the leaves the classification output. Features are split by order of information gain. The tree is truncated to height 5 to reduce overfitting. Adaboost employ a large number of weak learners, and the final output is the weighted sum of each independent classifier. In our case, we used a tree of height 1 as weak learners. SVM classifies by generating a hyperplane that best separates the data, calculated by maximizing the distance to the closest data point of each label. The hyperplane then dictates what label the model predicts for each new data point. Naive Bayes is a probabilistic model employing Bayes theorem, but simplified with the assumption of independence between features.

We used the Python library scikit-learn[^13], containing out-of-the-box configured implementations of machine learning algorithms, for our project. As our dataset does not contain a separate testing set, test performance is estimated by repeated runs of 5-fold cross validation.

[^12]: “UNB ISCX Intrusion Detection Evaluation DataSet,” 2015, Information Security Center of eXcellence


5. NETWORK VIRTUALIZATION AND OPENFLOW

The end goal of the Wonderwall project was to build a commoditizable network filtering system that could be deployed easily as a hardware appliance at the perimeter of a home or corporate network. Designing the hardware for Wonderwall was outside of our capabilities, so we decided instead to use network virtualization technology to emulate a deployment of Wonderwall during development and testing.

Mininet\textsuperscript{14} is software that enables emulation of large OpenFlow-powered networks on an individual computer such as a laptop. Mininet is widely used for rapid prototyping of software-defined networks and provides direct control over individual hosts of a software-defined network by using system resource isolation/namespacing mechanisms of a host operating system. The authors of Mininet have published Linux virtual machine images\textsuperscript{15} which can be used to avoid the complexities associated with installing Mininet on a host VM. We used this virtual machine (running Mininet 2.2.0 on Ubuntu 14.04 LTS) to run our network virtualization.

Mininet is capable of emulating arbitrary network topologies. During development we elected to use a “star” layer 2 topology. A “star” network has switching hub and multiple hosts which share a single subnet. The switching hub switches intra-subnet traffic and bridges the subnet to the WAN. The “star” network topology is a the de-facto standard network layout for home networks and small businesses, which motivated our use of it as a layer 2 topology during development.

Figure 1: A typical “star” topology, with a single switching hub and multiple attached devices. (Wikimedia)

One of the principal tenants of software-defined networking is separation of the “control plane” from the “data plane.” In an OpenFlow network, OpenFlow switches make forwarding decisions based on OpenFlow controllers which can make case-by-case decisions for IP packet forwarding when routing rules - which are also determined by OpenFlow controllers - do not apply. Communication between OpenFlow controllers and switches transits the control plane while actual network traffic transits the data

\textsuperscript{15} “Download/Get Started With Mininet” 24 Apr. 2016
http://mininet.org/download/
plane. Our emulation runs the WonderWall classifier on a single OpenFlow controller which is directly connected to the OpenFlow-enabled switch as a leaf of the network “star”. Upon system start, OpenFlow configures the network switch to behave much like a simple “dumb” switch when forwarding LAN Ethernet frames, and like a typical router when forwarding packets between LAN and WAN. However, OpenFlow is also used to forward LAN-WAN traffic over the control plane to the Wonderwall classifier; the OpenFlow classifier performs out-of-band threat classification on the forwarded traffic.

The OpenFlow controller can react to positively-classified/malicious traffic in a number of ways. A network administrator could either elect to simply receive alerts about such traffic, block individual offending packets, or block the source of offending packets entirely. If Wonderwall is configured to block the source of malicious traffic entirely, it can accomplish this by installing null-route rules in the switch’s route table with OpenFlow; it can do this asynchronously with no impact to network performance. If Wonderwall is configured to block individual packets, however, the entire network will take a performance hit because the switch will be reconfigured such that classifier will lie along the critical path of LAN-WAN network traffic; in this case the OpenFlow switch will rely on the controller for every ethernet frame’s routing decision. Thus, for most “real world” applications, we expect network administrators to choose either alerting or source blocking as Wonderwall’s automated response.

6. RESULTS

The table below summarized classification performance. Decision Trees showed adequate performance, though test performance is noticeably worse. Adaboost achieved the best result, beating decision tree by a few percentage points and other classifiers and the rule-based snort by a handy margin. SVM, usually well-suited in other areas, falls short here likely due to class imbalance; the vast majority of packets are normal. Naive bayes also performs poorly for similar reasons and the naive assumption is inappropriate as it is the combination of a few key features that likely indicates an attack. Naturally, we ended up using Adaboost for our classification.

<table>
<thead>
<tr>
<th></th>
<th>Training Performance</th>
<th>Test Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Recall</td>
<td>Precision</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>96%</td>
<td>88%</td>
</tr>
<tr>
<td>Adaboost</td>
<td>97%</td>
<td>97%</td>
</tr>
<tr>
<td>SVM</td>
<td>30%</td>
<td>3%</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>98%</td>
<td>36%</td>
</tr>
</tbody>
</table>

Table 1: Classification Performance of Machine-Learned Models
Decision Trees also has an added benefit of showing feature importance as splits are ordered by information gained. The features in descending order of importance are: Source Port, Destination Address, Source Byte Size, Source Address, Destination Byte Size, Source TCP Flags. The other features provide minute contribution to the classification, and are most likely truncated.

The graph below demonstrates the the 2 million data points is adequate for our classification. $F_1$ score levels off after about 100,000 samples. A dataset 10 times larger would yield negligible increases in performance. The curve is similar for other classification models.

![Learning Curve - Truncated Decision Tree](image)

**Figure 2: Learning Curve showing $F_1$ score plotted against dataset size**

7. ETHICAL AND PRIVACY CONSIDERATIONS.

Our system relies on tagged network traffic dumps for use during classifier training. Since the actual network traffic data from an organization deploying Wonderwall may contain personal and/or confidential information (financial, medical, etc.), we expect network administrators to exercise the same level of care for the security and the privacy with their Wonderwall training data as they would with any other traffic dump of their network. Similar concerns apply to attack models that would be shared between deployments of Wonderwall. Because Wonderwall’s base model is trained on a publicly available dataset we do not believe it poses any privacy concerns.
We do not believe our project raises any ethical concerns.

8. FUTURE WORK

We would like to extend the Wonderwall training tools to include a user-friendly interface used by administrators to tag “questionable” network traffic such as TCP streams as malicious/benign. The tags could then be used to improve the deployment-local Wonderwall classifier model.

One practical extension to WonderWall would be a decision-tree-to-Snort-signatures translator. The practicality is supported by the decision tree’s superior F<sub>1</sub> score to the Snort classifier’s F<sub>1</sub> score.

We would like to extend our system to identify more complex attack patterns; we believe that more sophisticated learning methods, such as deep learning, would be sufficient to capture a good model to identify such attacks.

A network performance impact study to determine whether Wonderwall is suitable for high-volume networks should be conducted, with the primary objective of identifying bottlenecks and improving overall network throughput.

Finally, another addition would be to enable the Wonderwall OpenFlow controller software to be distributed across different hardware controllers, providing better fault-tolerance and scalability for larger networks.

9. CONCLUSION

In this report we describe a system for classifying computer network traffic as malicious or benign based on a machine-learned model that was trained on a generalized dataset which we believe to be representative of most categories of network attacks. We describe a virtualized, software-defined network environment to simulate the deployment of such a classification system in a real-world network, and describe different ways for this system to automatically respond to threatening network traffic. Finally, we suggest a variety of future extensions to the project which enable Wonderwall to become an enterprise-grade intrusion detection/prevention system.

10. REFERENCES

[See footnotes.]