Using Automatic Signature Generation as a Sensor Backend

Dan Wyschogrod  
Principal Engineer  
BAE Systems

Jeff Dezso  
Software Engineer  
BAE Systems

Overview

• 0-day Exploit Problem
• Existing Intrusion Detection Technologies
• ASG Bridges Anomaly and Signature Based Intrusion Detection
• Benign Traffic Filtering
• Models for False Alarm Dependence on Amount of Training Data
• Experimental data on training database
• Experimental data on false alarm rate
• Remarks on missed detection
• Summary
0-Day Exploit Problem

- 0-Day exploits are attacks first seen in the wild that exploit vulnerabilities for which there are as yet no known vendor defenses
- Typically, there are no existing signatures to defend against 0-Day exploits
- Particularly prized by attackers since large numbers of machines may be affected without the knowledge of victims
  - Useful for creation of botnets
  - Exploits may exist for months without security community being aware

Signature Based Intrusion Detection

**Advantages**
- Intrusions blocked at the firewall at significant distance from vulnerable assets
- Programs such as Snort are optimized for efficient pattern matching
- Search rate largely independent of numbers of rules
- Hardware exists from such vendors as CPacket and Cavium OCTEON for hardware rate pattern matching
- Signatures can be hand tuned for low false alarm rate
- Once a signature for an attack is extracted, it can be widely disseminated
- No computational load placed on hosts

**Disadvantages**
- No way to detect zero day attacks
- Exploit may exist in wild for prolonged periods
- Even when samples of attack are captured, signature extraction may take hours
- Signatures typically tuned by hand
- Not useful against encrypted or otherwise obscured attacks
- Techniques such as polymorphism seek to deliberately obscure attack and avoid signatures
Anomaly Based Intrusion Detection

Advantages
- Can be effective against zero day attacks
- Can take advantage of both host based and network based sensors
- Host based sensors can be used after decryption

Disadvantages
- Frequently suffers high FA rate
- Requires training for “normal” traffic
- Host based anomaly sensors detect attacks near vulnerable resources
- Host based anomaly sensors impose computational load and perhaps latency to allow incoming traffic to be analyzed
- Network based anomaly sensors typically placed on subnet with vulnerable servers
- Once attack is found on one host under favorable circumstances, this information is not easily disseminated to other vulnerable hosts
- Requires heavily instrumented hosts and networks and can’t leverage results to non-instrumented networks

Solution: Bridge Anomaly and Signature Based Intrusion Detection

• Critical technology is Automatic Signature Generation (ASG) from attack packet examples
• Anomaly detection remains part of system so that protection against initial attacks exists
• Anomaly detector forwards anomalous packets to ASG
  - Front end may be stochastic or protocol anomaly detector or even a honeypot
• ASG generates signatures and forwards them to signature matching hardware or software both locally and remotely
• ASG must be relatively robust to the possibility that anomaly detector may have non-negligible FA rate
Approach Used in Extracting Signatures from Attack Packet Examples

- Benign traffic is collected from a network for a period of time and all substrings within a range of lengths for a given service (TCP or UDP port) are extracted and saved.
- All possible substrings from an attack packet within the given range of lengths are extracted.
- Benign substrings are used to filter the attack substrings so that only "unique" substrings survive.
- More benign traffic used, the better the filtering.
- As long as any signature strings survive, repeated instances of the exact original attack can be found.
- In the case of polymorphic attacks, if short attack invariant substrings are retained, variants can be detected.

Efficient Implementation of Benign Traffic Filtering

- Use trie data structure.
- Tries efficiently store large numbers of strings.
- Allow for straightforward implementation of intersection, subtraction and other operations.
- One trie is produced for attack packets.
- Benign traffic can be organized in a huge trie.
  - In later work, we have used Bloom filters.
- Trie subtraction is used to produce signature trie.
Models for False Alarm Dependence on Amount of Training Data

- We derive two simple models for two kinds of services
- Basic Models
  1. Services are modeled as streams of uncorrelated, random data
  2. Assume that text is linguistic in nature in that it is composed of words that repeat themselves with varying frequency
- Many services are mixtures of the above models

False Alarm Rate Derivation

Symbols

- $S_c$ – number of bytes collected in the benign database
- $P_{mbt}(s)$ – probability that string $s$ is missing from database after a certain amount of training
- $P_a(s)$ – probability of the string occurring in an attack packet as part of the benign portion of the packet payload
- $P_b(s)$ – probability that string $s$ occurs per number of bytes in benign traffic when system is “live”
- $N_s$ – mean number of input bytes between sightings of a benign string $s$
- $P_{BA}$ – average percentage of each attack packet that contains benign traffic
- $W(N_s)$ – weighting function for occurrences of strings with equal values of $N_s$
- $|s|$ – Length of string in characters
- $C_W$ – constant frequency weighting.
False Alarm Rate Derivation (Continued)

- False alarm rate per byte of benign traffic is product of three factors:
  - $P_{mdb}(s)$ – Probability that string $s$ is missing from database after a certain amount of training
  - $P_a(s)$ – Frequency of string $s$ in benign portion of attack packet (e.g. http header, etc.) in calculation of signatures
  - $P_b(s)$ – Probability of string $s$ in live traffic (after training)

False Alarm Rate Derivation (Continued)

**Probability Factors**

\[
\begin{align*}
P_{mdb}(s) &= (1 - \frac{1}{N_s})^S, \\
P_a(s) &= \frac{1}{N_s}, \\
P_b(s) &= P_{b/A} \cdot \frac{1}{N_s}
\end{align*}
\]

- False Alarm Rate per unit time is product of probability factors (assuming statistical independence):

\[
P_{fa} = P_{b/A} \cdot \frac{1}{N_s^2} \cdot (1 - \frac{1}{N_s})^S.
\]
False Alarm Rate Derivation
(Continued)

- To calculate $P_{FA}(S_o)$

$$P_{FA}(S_o) = P_{B/A} \sum_{i} W(N_s) \left( 1 - \frac{1}{N_s} \right)^s dN_s$$

- Two possible values for $W(N_s)$ lead to the random and syntactic models
  - Random (used Dirac Delta Function)
    $$W(N_s) = \delta(N_s - 2^{|s|})$$
  - Syntactic (assume constant weighting for all frequencies)
    $$W(N_s) = C_{w}$$

FA Rate for Random Model

- Evaluation of integral leads to

$$P_{FA}(S_o) \approx \frac{P_{B/A}}{2^{|s|}}$$

- Original signature occurs very infrequently, we are unlikely to ever match it again in benign data
  - However, these signatures that are unlikely to match still take up space in the matching hardware or software

- “Random” type service
  - Encrypted traffic as with ssh where it is hard to find anomalies
  - Binary content services for transfer of jpegs, mpegs, etc.

- Longer signatures less likely to be seen again and cause false alarms
FA Rate for Syntactic Model

- Evaluation of integral leads to
  \[ P_{FA}(S_c) = P_{B}/\gamma C_W \frac{1}{1 + S_c} \]
- Initially, training leads to rapid drop in FA rate, then more training needed for higher amounts of reduction
- FA rate can be driven arbitrarily low, but not to zero
- Experimental results presented shortly

Experimental Results for Training of Benign Database for Various Services

- N-gram size is fixed at 5
- Growth of the size of the ssh database, which is an encrypted service and therefore random, is explosive
- Port 631 which is the Internet Printing Protocol, is syntactical in nature with a small vocabulary and therefore flattens out very quickly
- One port 80 plot shows all packets and increases explosively with transmitted binary data
- Other port 80 plot uses only ASCII and is much more bounded
## Experimental Results for Syntactic Model – Apache Attack

- Metasploit `apache_chunked_win32` attack
- Fit syntactic model
  - Fitting constant $P_{fa}C_{up} = 537$
  - Indicates slower convergence
- FA drops in quantized amounts when particular signatures are filtered out
- Final point takes PFA rate to zero for test data
- To achieve PFA of zero on test data, three times as much training data was needed
- PFA/MB is tested using 856 MB of test data

## Experimental Results for Syntactic Model – Samba Attack

- Metasploit `samba_trans2open`
- Fit syntactic model
  - Fitting constant $P_{fa}C_{up} = 183$
  - Indicates faster convergence
- FA drops in quantized amounts when particular signatures are filtered out
- Final point takes PFA rate to zero for test data
- To achieve PFA of zero on test data, twice as much training data was needed
- PFA/MB is tested using 3,264.7 MB of test data
Remarks on Missed Detection

- If attack is repeated with exactly the same packet, if any signatures survive filtering, the repeat attack will be detected
  - For the two examples above, signatures survived filtering for all amounts of training

- In the presence of polymorphic attacks, subsequent variant attacks will be detected as long as attack invariants survive filtering
  - By using multiple length strings, we hope to capture the shortest unique strings as signatures, thus including the invariants
  - Initial experiments using Metasploit ability to add varying filler to different instances of an attack were able to extract attack invariant elements

Summary

- ASG “back end” facilitates distribution of 0-day detection capabilities achieved by anomaly and other detection mechanisms to broad range of systems
- ASG signatures can be refined to have extraordinarily low false alarm rates
- ASG module can be used with wide range of front end sensors
- ASG extends perimeter of protection away from vulnerable hosts
- ASG makes efficient use of existing high speed string matching technology