

1 Introduction

Context for the lecture: We spend time looking at Bro because it is one of the few network intrusion detection systems (NIDS) explored in the literature. Since it's a systems paper, it includes a lot of engineering elements. However, since NIDS are a broad area, we can then go from this starting point to extensions in a number of directions.

Intrusion detection is goal driven; the goal is to discover that resources are not being used properly:

1. Someone has access that should not have access
2. Someone is misusing a resource (this is mostly a policy decision)
3. Someone is denying legitimate usage of a resource.

A key point is that intrusion detection consists of two parts: detection and response/remediation. Flooding an operator with events does nothing to improve security; to be effective an IDS must alert an operator only when there is an actionable event.

Intrusion detection as a field arose in a responsive fashion, and somewhat hodge-podge since people came to it from a number of different perspectives:

1. Cryptography: Starting point is to think about building systems with security properties that can be proved.
2. Networking: Starting point is thinking about how network activity manifests when observed by a monitor.
3. Operators: Starting point is “fire-fighting,” the need to cobble together the means of informing the operator of pertinent events.

Finally, NIDSs are similar to firewalls, but differ in significant ways. Firewalls reside on the *forwarding path*, and thus directly control the flow of traffic. The rules in firewalls are usually static and stateless or involve minimal state. NIDS passively monitor (unless they are NIPS—network intrusion *prevention* systems) and perform significant *contextual* analysis to assess whether activity merits generating an alert.

2 Key Issues in IDS

1. Tradeoff between false positives and false negatives
2. Base rate fallacy: even if an IDS is accurate 99.99% of the time (only one false positive in 10^4 , only one false negative in 10^4), it can still be ineffective if the sheer volume of events being sent to it is high. Consider

10^8 events/day, of which 10^2 are actual attacks. The IDS will generate 10^2 correct alerts—but also 10^4 false positives, overwhelming the operator with bogus alerts.

3. Actionable detection: Deciding how to prioritize or interpret information in such a manner that an operator knows upon which ones to act

3 Advantages of Network Intrusion Detection Systems

1. Cheap. This is the biggie!
2. Trustworthy: no assumption of end system trust, and pertinent even in a pseudo-adversarial context (user violates policy)
3. Network Visibility: additional statistics about performance or operation can be gathered

4 Limitations of Network Intrusion Detection Systems

1. Reduced semantic information (some key info is known only to sender/recipient)
2. “Vantage point”—what the monitor sees is not the same as what the end systems see
3. Reduced visibility: caused by encryption, incomplete threat models, or split routing
4. Performance
5. Evasion
6. Privacy, Legality, and User Morale

5 Styles of Network Intrusion Detection Systems

5.1 Signature Matching or Misuse Detection

This style focuses on recognizing known attacks. “Snort” is an example of this style.

The advantages of this style are that it is conceptually simple, and efficient to implement; signatures can be easily shared, and readily implemented in hardware for a major performance gain. Problems with this style are that it is easily

avoidable with polymorphic code, may generate false positives with legitimate traffic, and only works with known attacks.

5.2 Anomaly Detection

The basic idea behind anomaly detection is that attacks look peculiar and out of the ordinary. This style of detection synthesizes a model of normal activity, and alerts on deviations.

The main advantage of this style is the ability to defend against new attacks that have not been seen before. The main disadvantage is that the detector can fail to detect even known attacks, and can generate false positives on benign activity. The key question is whether there is a tight “envelope” describing normal activity for which attacks are outside the envelope, but little legitimate traffic is. Other disadvantages: the possibility of training the system from polluted data (already has attacks in it); not insight into coverage (understanding what its power will be in a particular deployment).

5.3 Specification Based

This style is somewhat similar to anomaly detection. Instead of constructing a model of what is usual, a model of what is *allowed* is constructed, and policies to enforce it are upheld.

Advantages of this style are that it codifies behavior, thus, provides a specified coverage, uses contextual information, can protect against novel attacks, and can protect against specific known attacks. The disadvantages of this are that it involves a lot of overhead in construction and maintainance, and cannot be shared across networks.

5.4 Behavioral Detection

This style of detection focuses on analyzing behaviors that reflect activity taken by an attacker subsequent to a compromise.

Advantages of this that it can detect a broad range of compromises, regardless of whether they used a new approach or not. The main disadvantages are that it’s reactive (the compromise has already occurred) and incomplete (the attacker might not do any of the monitored behaviors).

6 BRO Overview

Bro is a passive NIDS not based on signatures. It most closely mirrors the behavioral or specification based models, and is designed to work well on commodity hardware.

The design goals for Bro were:

1. High speed/volume
2. No packet filter drops
3. Real-time notification
4. Mechanism orthogonal to policy
5. Extensibility
6. Avoidance of simple mistakes
7. Protection against attacks on the monitor

7 Bro Architecture

Bro operates by passively tapping a link at the network layer. From here, libpcap splits a kernel filter and user API above to weed out the number of packets to worry about. Although kernel filters used to provide a performance boost of a factor of ten, recent applications violate conventional use of ports, making it difficult for a static filter to capture their traffic. To compensate for this, Bro uses Dynamic Protocol Detection to determine the protocol of a stream. Another facet of Bro's filtering is that by simply looking at the TCP SYN, FIN, and RST flags, you can determine the time, duration, IP hosts, ports, and volume of traffic.

At the next stage, Bro's Event Engine distills the filtered stream into high level, policy neutral events.