

Resynchronizing NIDS Systems

Eric Hacker 2000-09-22

Introduction

This article is about fixing things. Greg Hoglund and Jon Gary have demonstrated many ways of breaking things, particularly network intrusion detection systems (NIDS) systems, in their thoughtful article [Multiple Levels of Desynchronization and other concerns with testing an IDS system](#). The article will demonstrate that a NIDS can be designed to address most of the issues they raise. Much of the first half is dry, one should feel free to skip ahead if one does not have the patience for fundamentals. This article is the exploration of ideas that will require further research to be fully validated and may not apply to all environments.

I believe NIDS re-synchronization is achievable through the use of anomalous traffic detection in addition to attack signature detection. Rather than focus NIDS on misuse detection, a NIDS can look for traffic that is not normal. Furthermore http traffic is normalized to a certain extent by the limited number of clients and proxies. Additional normalization can be done with client side code for client submissions. Anything that does not meet the criteria of normal traffic can be identified and examined. This article will focus on web servers, although many of the ideas apply to other services as well.

Attack Signature Detection vs. Anomalous Traffic Detection

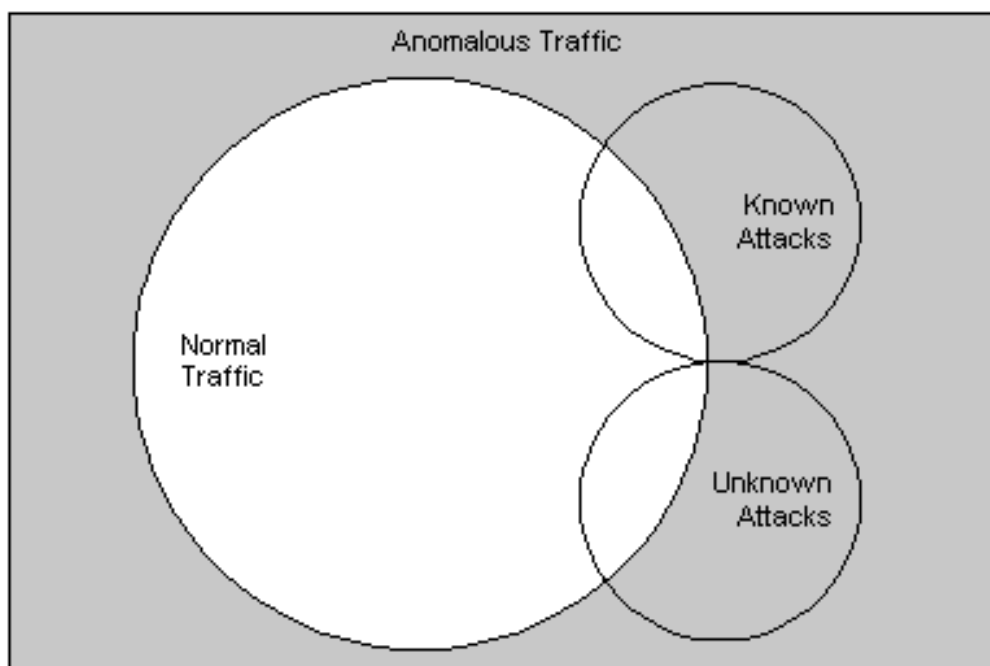
Typical NIDS target known hostile traffic. These NIDS operate by examining network traffic for attack signatures. A packet may contain the details of a buffer overflow or the phrase `/etc/passwd` which would indicate hostile activity. Signature based NIDS must be able to see the attack to be effective, otherwise there is a false negative. Thus one method of bypassing a signature NIDS is to obscure the attack with various network manipulations. This is the foundation for the Multiple Layers of De-Synchronization article.

Anomalous NIDS look for traffic that is not normal [1]. Anomalous NIDS rules are used currently for some NIDS. For instance, Stephen Northcutt has recommended in the SANS Intrusion Detection course that one should alert on all fragmented packets [2]. Fragments are rare enough in the realm of packets that they when they are present, they are often hostile. This is alerting on anomalous traffic. There is nothing in the packet itself that indicated hostility. One just does not often see this type of packet, and therefore it is worth investigation. In an

environment where fragments were not rare, they would not be anomalous.

The set of anomalous traffic is larger than the set of data identified by a signature NIDS. These sets do not necessarily overlap to a great degree and thus must be used to complement each other. Anomalous NIDS should not replace signature NIDS.

Anomalous traffic does contain some unknown attacks, which improves the set of attacks identified. Thus identifying anomalous traffic also may recognize attacks for which no signature exists. Trying to transform a known attack into something new can be caught by an anomalous NIDS, if the transformation is abnormal. Hoglund and Gary allude that such techniques may be used to identify some of the attacks in their paper. Here is a Venn diagram showing these sets.

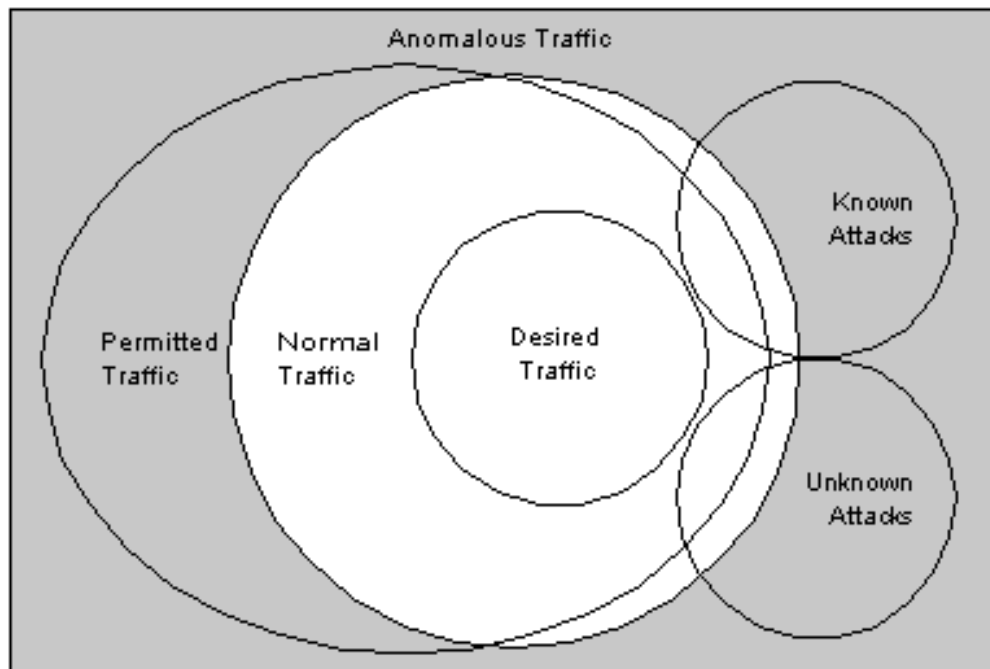


Venn Diagram #1

From this diagram it is obvious that there is a higher risk of false positives with anomalous NIDS. However false positives are not as bad as false negatives. A false positive creates an unnecessary administrative burden. If there are too many, they may be ignored due to the 'cry wolf' effect. Ignored alerts are no better than none at all. However a false negative is no alert at all. Thus minimizing false negatives is the first priority. If there were too many false positives with a specific check, then perhaps one needs to reconsider whether that traffic is anomalous.

The set of normal traffic is not the same as the set of permitted traffic, though the sets must overlap significantly. By permitted traffic, I mean the traffic allowed by either protocol

implementations at any network layer or applied security policies. For instance, a VPN server may typically receive TCP traffic to port 500 and ESP traffic on its external interface. Odds are that is all one wants to permit this traffic as well. A web server may rarely see a GET request to a particular CGI script without a referrer field. It is possible that such a request could be made and it is within the standards, so one is likely to permit it. However such a request is anomalous and thus should be worthy of investigation. Again a Venn diagram will help illustrate this relationship.



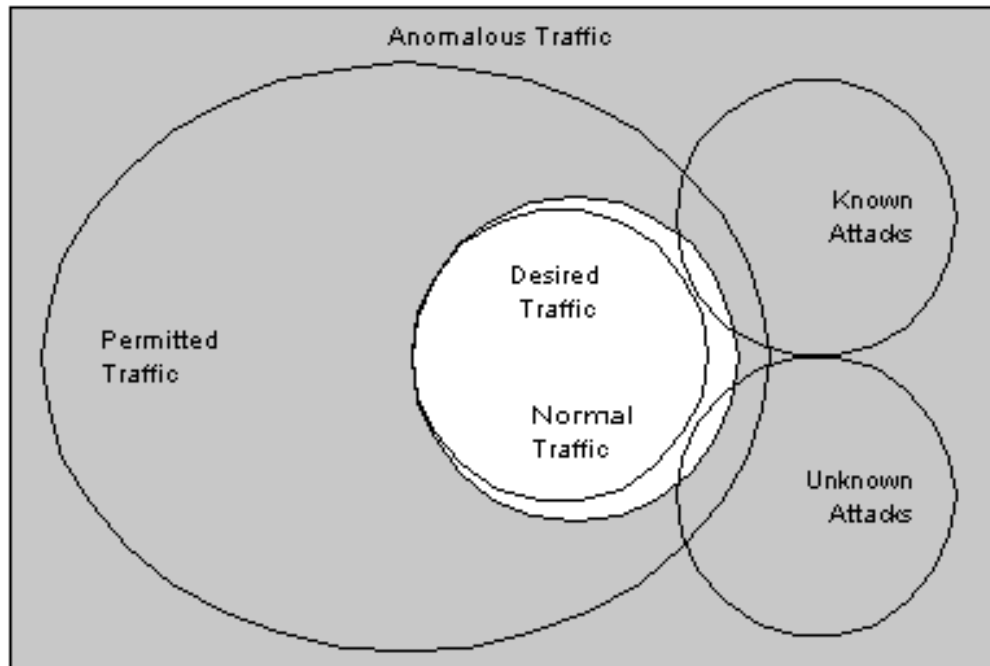
Venn Diagram #2

The set of permitted traffic may not entirely overlap the set of normal traffic. Ideally, the sets of known and unknown attacks will not overlap with the permitted, but that is not achievable. Thus the goal is to either deny or at least identify all attacks. By manipulating the traffic which one controls, one can advance on that goal.

Client Side Normalization

Security practitioners all know that client side security does not work. Ryan Russell, in *Hacking your Network*, describes this as one of the laws of information security. However that does not mean that client side security has no role in information security. If one has some control over the client, then one can use client side processing to manipulate the set of normal traffic. By making the set of normal traffic smaller, it overlaps less with the known and unknown attacks. More activity becomes anomalous and thus can be caught by an anomalous NIDS.

Client side processing can also bring the normal traffic within the set of permitted traffic. Again, this does not create an absolute restriction. It does change the normal traffic one expects. Client side security can strive to make the normal traffic the same as the desired traffic. Of course, that is not likely achievable, but it should be the goal as displayed in this Venn diagram.



Venn Diagram #3

Comparing diagrams #2 and #3, one sees that the normal traffic has been drawn within the set of permitted traffic and is almost the same as the desired traffic. This is the effect that can be achieved with client side security. With this in place, anomalous NIDS techniques combined with signature NIDS can do much to identify attacks.

Client Normalization in Practice

In web security, traffic is generated by web clients or servers. That itself is a normalizing factor. Further deliberate coding can increase the normalization. For instance, if a field is supposed to be up to 20 characters long, then enforce that length at the client. Even better, pad it out to 20 characters as well. Then anything not 20 characters did not come from a normal client. If one is concerned about the appearance of certain characters in URLs, then use client code to encode those characters or to enforce restrictions.

Client normalization is definitely not a panacea. In many situations it is simply just not possible.

To be effective, it requires collaboration between web developers and security personnel. I'll pause while you pick yourself up off the floor and stop laughing. There will still be the opportunity to attack within the bounds of the normalized traffic. Many current NIDS do not have the granular analysis capability or application protocol knowledge to make this fully realizable now. However, client side security can be effective in an overall security program when combined with anomalous NIDS to some degree today.

Normal Net Traffic

If one has ever sat down with a sniffer and looked at Internet traffic at layers three and four, one probably knows how boring it is. While there are a great many features provided in the RFCs for IP and TCP, UDP, ICMP etc., much of it is rarely used. The bulk of the traffic occupies a narrow range of the complete feature set. My original outline for this article included a more detailed look at some normal web traffic as well as particulars as to why certain packets like fragments are rare. It became clear as I began writing that covering both the minutiae of Internet traffic and the value of anomalous NIDS would not fit in a single article. I hope to present further technical details on how some things would work.

Applying Anomalous NIDS to the Ideas

Two Systems can never be exact copies of one another, and bury it deep enough and no one will find the treasure

Much of these techniques rely on IP and TCP features that are rarely seen on the Internet, particularly for web servers. Fragments and small TCP segmentation stand out like a nerd at the prom. No normal web client would segment TCP any smaller than the MSS. The MSS is dependent on the MTU. The MTU almost never dips below 500. The NIDS does not need to dig up the treasure, mere evidence of the burial is significant.

If two different IP stacks commonly reacted differently to the same packet, then there would be a serious breakdown of communication on the Internet. Standards is what makes the whole thing work and the places where OSs operate differently is at the fringes of the standard and thus anomalous.

Low TTLs can easily be trapped because one knows how far the NIDS is from one's server. One probably wants to know when traceroute is being used, so alerting on low TTLs is valuable. This

requires some fine tuning in practice, such as ignoring multicast routing protocols, but low TTLs are identifiable.

The best way for NIDS to handle HTTPS is to front end the SSL processing on a separate SSL acceleration device. Locate the NIDS between that and the web server. In this way the NIDS will see the web traffic in the clear. Another possibility is to use client side scripting to normalize the traffic. All HTTPS requests may be made the similar in length. This idea needs much more research.

HTTPS traffic patterns can also be used. For instance, most e-commerce sites only have a few SSL pages. Typically a user may need only enter these once per transaction. Thus if a single IP generates more than a few SSL requests per hour, it is worth investigating.

Cross reference the output with the input

Hoglund and Gary have identified a definite weakness in the logging capabilities of IIS. With this valuable data, one can ensure that one's own web content does not use URLs that can be inconsistently interpreted and logged. Once such URLs are removed, then those characters are anomalous. If one's web developer insists in using such URLs it is time to cut back on her caffeine supply.

Look for 'stripped' data - additional data that doesn't affect the validity of a request

Stripped data is anomalous. Web clients do not generate superfluous data in requests. "." in an URL is already detected by simple pattern matching IDS systems. Other attempts at padding can be identified as well. By using the same reduction rules as the web server uses, a NIDS can identify any request that is not the same length before and after reduction.

Use invalid filename characters

NIDS can be set to see and alert on invalid filename characters. CGI can be written to not use them in the data portion of URLs.

Exploiting NTFS case sensitivity

NIDS do not need to be case sensitive. If a known attack is not the proper case for the web server, a NIDS should still identify that an attack was attempted. One can also use case as a

normalizer. Change cgi-bin to Cgi-Bin, adjust internal links accordingly, and suddenly all the script kiddies are as clandestine as a fire muster.

Use Alternate Character Encoding or Equivalent Characters

One can also identify when encoding particular characters is being attempted. One can even go so far as to encode characters differently with client side processing, thus making typical encoding anomalous.

Add/Remove trailing meta-characters

Trailing characters do not affect a signature NIDS as the pattern of attack is not altered. Hognlund and Gary have identified that trailing characters may have potential DOS ramifications but it appears only in high quantity. Numerous query attempts on the order of 100-200 in a few seconds is likely anomalous and should trigger an anomalous NIDS.

Fortunately, this attack requires a TCP handshake and thus spoofing source IPs is not possible. However, in practice this may break with proxies, caches, and NAT devices. They can cause many hits from one IP for many people for large sites and thus may generate such traffic. Deeper application knowledge would then be required to catch this.

Inserting invalid characters into fields

Use client side processing to standardize field lengths and check for discrepancies. Design so that invalid characters are anomalous and trigger on them.

Sensitive dependence on certain fields

IDS can catch multiple attempts to abuse sensitive fields. Client normalization can also help to ensure that sensitive filed abuse is far outside the realm of normal traffic.

Conclusion

Hognlund and Gary have pointed out many problems that can beset a NIDS. This paper illustrated some ways that the problems they outline can be worked around by using anomalous NIDS principles. Anomalous NIDS techniques should work together with signature based

techniques to provide a stronger NIDS position.

De-synchronization is a very real problem for plug and pray NIDS and for security unconscious web developers. With proper planning and coordination using anomalous traffic detection, the NIDS can be brought back in to sync. It does not make intrusion detection easier, but it does make it more likely.

Eric Hacker is a security consultant with Lucent Technologies NetworkCare Professional Services.

Relevant Links

[Subscribe to the FOCUS-IDS Mailing List](#)

SecurityFocus.com

[Multiple Levels of De-synchronization](#)

Greg Hoggund and Jon Gary

[Privacy Statement](#)

Copyright 2006, SecurityFocus