SAND: An Architecture for Signature-based Automatic Network Protocol Detection

The University of Tulsa
Institute for Information Security

G. Louthan and J. Hale
University of Tulsa, Tulsa, Oklahoma, USA

Abstract

Conventional network protocol identification based on well-known port numbers is no longer sufficient to identify and classify traffic in modern networks. Applications can now dynamically change port numbers, users select alternate ports, and attackers attempt to trick identification systems by changing ports. We propose an automated, signature-based method for rigorously identifying network protocols based upon their definitions rather what in some cases amounts to little more than a configuration option.

Introduction

Today, virtually all conventional network monitoring relies upon classifying packets based on TCP or UDP port numbers. Unfortunately, in networks that are not tightly controlled, lack of adherence to well-known or standard port conventions and can result in incorrect or incomplete identification of network traffic. For example, the Internet telephony service Skype is capable of using multiple (in most cases, randomly-selected) ports and other methods to evade firewalls [1].

To meet this need, we propose a new method and system for traffic classification, capable of reliably identifying application traffic regardless of port number. It is *content-aware*, making its identifications based upon the actual content of packets that adhere to known, fixed protocol formats.

Architecture

Fig. 1 shows the system's basic architecture. Logically, it includes three parts: an identification module, which performs the actual protocol detection; one or more analysis modules, which receive protocol information from callback functions they provide to the library; and a set of protocol identifiers, used to specify signatures.

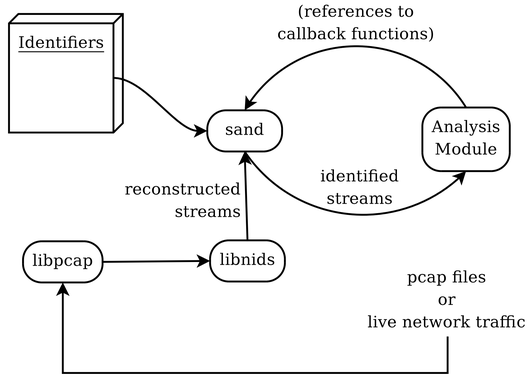

Figure 1: The basic architecture of the SAND system.

Discrete packets are reassembled into a stream using libnids, a library that provides a user-space implementation of the Linux TCP/IP stack [2] and provides the interface to libpcap, which is used to capture network traffic.

Identification Strategy

The identification strategy depends upon a set of identifiers loaded from files at run-time. A single identifier specifies the information necessary to identify a single protocol and is made up of one or more signatures.

A single signature in an identifier includes start and end terms, which sandwich potentially useful information that can be passed back for analysis. This reflects a form that network protocols frequently take: *"stringset1*stringset2"* [3]. For example, in the SSH protocol, each party must send a protocol identification string of the form "SSH-protoversion-softwareversion SP comments CR LF"; therefore, the SSH identifier finds the strings "SSH-" and "CR LF", taking the string between them to be the protocol version [4]. This identifier is provided in SAND format in Fig. 2.

```
protocol = "SSH"
threshold = 2
[server0]
start = "SSH-"
sig = "s_version"
finish = "\n"
[client0]
start = "SSH-"
sig = "c_version"
finish = "\n"
```

Figure 2: The SAND identifier for the Secure Shell protocol.

Fig. 3 illustrates the general strategy for identifying a stream. As signatures are matched, a "certainty" value is updated for that stream and protocol. When the certainty reaches the specified threshold, searching ceases and the identification is made.

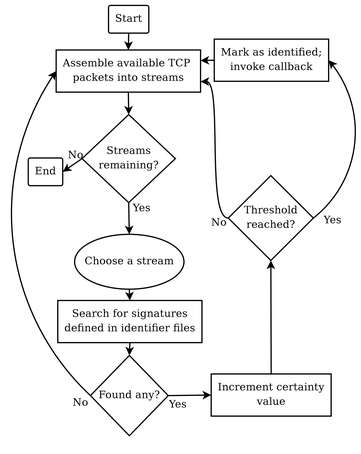

Figure 3: The strategy for identifying a stream in SAND. The certainty value is specified in the identifier file for the protocol.

Analysis

SAND identifies protocols according to signatures specified in identifier files. The search system allows the location of arbitrary signatures at arbitrary locations in TCP streams and the parsing of simple information such as version strings based upon specifications made in the identifiers.

While many protocol identification systems, including those is most common use, operate on individual packets (whether on packet headers alone or on their payloads), SAND is fully stream-based. By utilizing an actual implementation of a TCP/IP stack, the system can robustly reassemble packets into streams; this should prove more reliable than any system that depends upon single independent packets.

For example, an attacker seeking to avoid detection by a packet-based system could use TCP segmentation to construct packets that do not contain signatures but still deliver equivalent information by altering the size of each packet to ensure that the packet boundary falls in the middle of a signature element, and possibly reordering the packets. The stream reassembly component of SAND ensures that such efforts will fail.

Future Work

A survey of real-world traffic, including attacks against similar systems, would prove valuable in confirming or refuting the need for the highly robust stream reassembly we implement.

The speed and performance of the system also need to be analyzed. The exact performance consequences of properties of the identifier (such as number and length of signatures) should be weighed alongside the properties of the traffic. Detailed results in that area will allow realistic judgments to be made about the feasability of deploying a fully stream-based network device in a real network.

Finally, a word about the future of network protocol identification in general is warranted. We propose and advocate for a new approach to identifying network protocols. It is our hope that reviving discussion of this necessary study will stimulate further work in the field, because the problem is only worsening, as protocols such as Skype and BitTorrent, which violate the basic assumptions that enable port-based identification to work, proliferate even further.

References

[1] S.A. Baset and H. Schulzrinne. An Analysis of the Skype Peer-to-Peer Internel Telephony Protocol. *Arxiv preprint cs.NI/0412017*, 2004.

[2] R. Wojtczuk. Libnids, 2008.

[3] S. Sen, O. Spatscheck, and D. Wang. Accurate, scalable in-network identification of p2p traffic using application signatures. *Proceedings of the 13th international conference on World Wide Web*, pages 512–521, 2004.

[4] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Transport Layer Protocol. Technical report, RFC 4253, January 2006.