International Journal of Scientific & Engineering Research, Volume 4, Issue 8, August 2013
ISSN 2229-5518

Policy Scripts to Detect Network Intrusions

Sanmeet Kaur, Maninder Singh

Abstract— Security is a big issue for all networks in today's enterprise environment. Hackers and intruders have made many successful attempts to bring down high-profile company networks and web services. Many methods have been developed to secure the network infrastructure and communication over the Internet, among them the use of firewalls, encryption, and virtual private networks. Intrusion detection is a relatively new addition to such techniques. Intrusion detection methods started appearing in the last few years. In this paper Intrusion Detection System called Bro is discussed. The major emphasis is on the design and development of the policy scripts to detect various network intrusions. These scripts are written using the scripting language of Bro, which supports various special data types to support network level activities. It also has signature-matching features to make threat signatures to match against various attacks and detect them later.

.

Index Terms— Intrusion, Intrusion Detection, Bro, Policy Scripts, HTTP, NIDS

1 INTRODUCTION

Today's network is very complex and the whole world is focusing on ease of use and functionality. This is making us more insecure. For hackers, these well-traveled paths make networks more vulnerable than ever before and with relatively little expertise hackers have significantly impacted the networks of leading brands or government agencies. Cyber crime is also no longer the prerogative of lone hackers or random attackers. Today disgruntled employees, unethical corporations, even terrorist organizations all look to the Internet as a portal to gather sensitive data and instigate economic and political disruption. With networks more vulnerable and hackers equipped to cause havoc, it's no surprise that network attacks are on the rise. So there is a huge need of detecting the threats and intrusion. For this purpose number of solutions is there, IDS is one of them. BRO is the most effective NIDS which can be used to detect these threats. However, no IDS can detect all the intrusions. So we need a combination of various techniques.

There are a number of well-known techniques for detecting network intrusions. Some of these are:

- Signatures or pattern matching
- Content analysis and parsing
- Statistical analysis
- Anomaly detection
- Bayesian methods

Each of these techniques has their relative pros and cons. Some are easy to use and quick to implement, but lead to a high number of false-positives. Others are hard to understand and are complex, but at the same time may be very effective in detecting desired flows.

The objective is to analyze these techniques and develop patterns (not in the regular expression sense but in the software patterns one) that direct us in writing effective intrusion detection modules for a variety of network traffic classes. The types of traffic which are of interest are:

- Web traffic, usually sent over HTTP [4] protocol.
- E-mail traffic, using one of the well known e-mail protocols like SMTP, POP3 and IMAP [5].
- Webmail traffic. This is placed in a separate category because this combines properties of both web and e-mail traffic.

In this paper, these techniques are implemented by developing Policy Scripts using Bro IDS. Both Bro analyzers and scripts are used to achieve the goal.

2. IMPLEMENTATION DETAILS AND EXPERIMENTAL RESULTS

In this part the policy scripts of Bro to detect various network intrusions are discussed. Various kind of traffic is captured by using Bro and analyzed offline. Some of the scripts are experimented on the live traffic also.

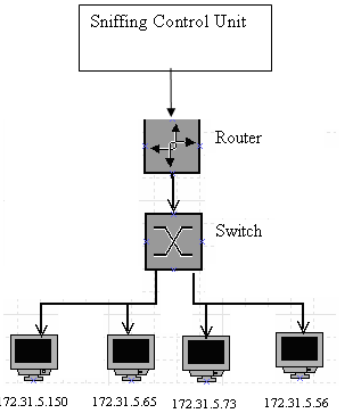

Figure 1: Implementation setup Diagram

The following steps are performed while implementing Policy Scripts:

1. First the traffic is captured by using wireshark [12] (earlier Ethreal) or by using libpcap feature inbuilt in Bro. This will be captured in a binary file with extension *.tcpdump* or *.out*,

- *Sanmeet Kaur is Assistant Professor in School of Mathematics and Computer Applications, Thapar University Patiala. E-mail: sanmeet.bhatia@thapar.edu*
- *Dr. Maninder Singh is Associate Professor and Head in Computer Science and Engineering Departent Thapar University Patiala. Email: msingh@thapar.edu*

for example *trace.out.*

2. Second, the policy script is written using Bro Scripting language. This file is having an extension *.bro* like *s_http.bro* and is placed in */usr/local/bro/policy* or */user/local/bro/site* directory.

3. Third, this script is run against the captured trace file to detect the required intrusion by using the following syntax:

bro –r tracefile scriptfile

4. If the traffic is live instead of captured one then also bro can analyze it by using *–i* used for interface instead of *–r* for read mode like:

bro –i scriptfile

The following are various tasks and milestones completed and their results. One of the traffic files were taken from the Bro Workshop 2007 [2]. This was modified using wireshark and later used in some of the experiments. The name of this traffic file is mail.trace.

3.1 Reporting all the HTTP URLs in the traffic

In this milestone all the URLs which are visited by a particular machine are reported by the Bro[1]. A script has been written to report all HTTP URLs in the traffic. The trace.out generated by the system is shown below.

Trace File: trace.out

This is the captured traffic file against which s_ http-header.bro policy script will run.

The following is the command to see the results:

bro -r trace.out s_http-header.bro

Result:

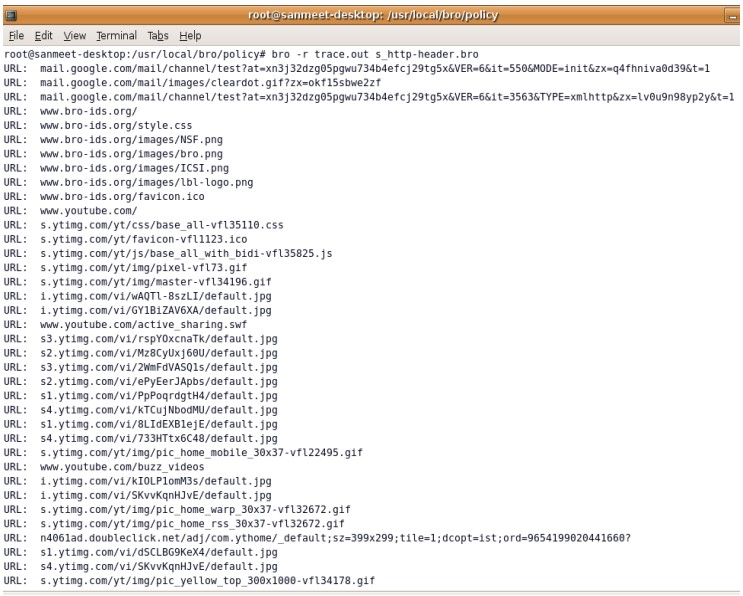

Figure 2: Reporting HTTP URLs visited by a host.

3.2 Reporting all the connections which are accessing www.youtube.com using HTTP

In this milestone all the connections which are accessing www.youtube.com using HTTP are reported by Bro. A script has been written and trace.out generated by the system is shown below.

Trace file: trace.out

The above is the captured traffic file against which s_ http-header1.bro policy script will run. The following is the command to see the results:

bro -r trace.out s_http-header1.bro

Result:

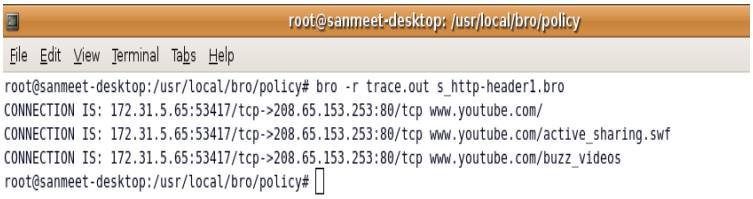

Figure 3: Reporting all the connections which are accessing www.youtube.com using HTTP

3.3 Reporting all the connections that includes emails directed to a particular email server

In this milestone all the connections that have a particular text like "@beta.banana.edu" in them are reported by Bro. The intention is to find out all emails that might be addressed to this account. SMTP [13] protocol support is required in this task. The trace.out generated by the system is shown below.

Trace File: mail.trace [Bro workshop 2007]

The above is the captured mail traffic file against which s_ smtp.bro policy script will be experimented . The following is the command to see the results:

bro -r mail.trace s_smtp.br

Result:

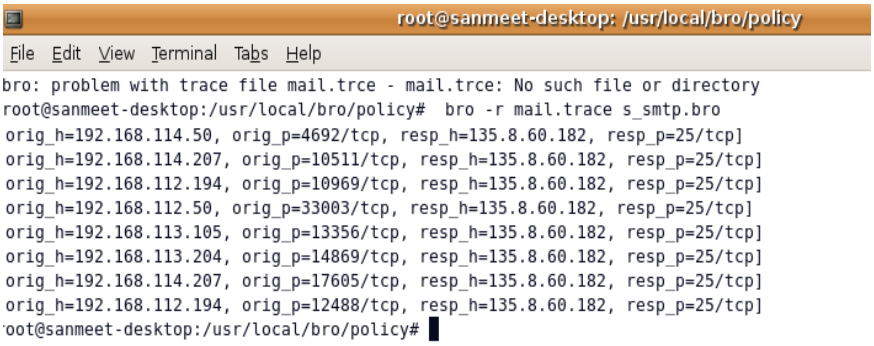

Figure 4: Reporting all the connections that have text "@beta.banana.edu"

3.4 Reporting all the connections that have a particular text like "@beta.banana.edu" or "@finch.eyrie.af.mil" in them

In this milestone all the connections that have a particular text like "@beta.banana.edu" or "@finch.eyrie.af.mil" in them are reported by Bro. The intention is to find out all emails that might be addressed to any of these accounts. SMTP protocol support is required in this task. The trace.out generated by the system is shown below.

Result:

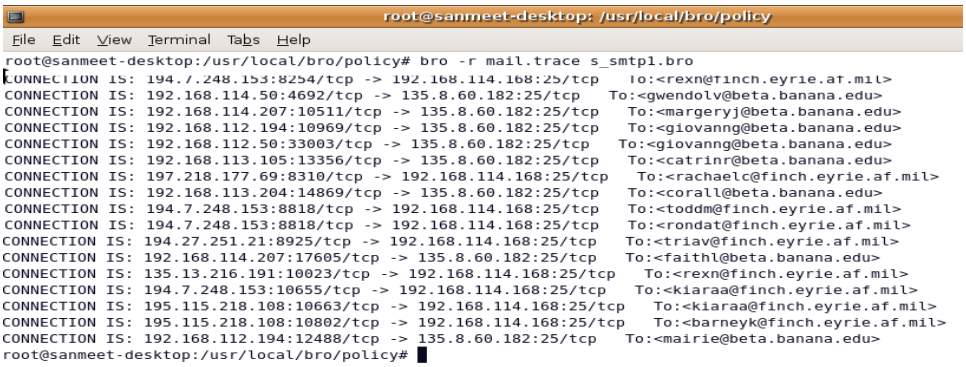

Figure 5: Reporting connections that have "@beta.banana.edu" or "@finch.eyrie.af.mil" in them

3.5 Detect if somebody is trying to access a particular website like "pic.geocities.com" using HTTP, log all further connection attempts by that host

In this milestone we have captured HTTP and other network traffic from multiple clients. Once it is detected that somebody is trying to access a particular website like "pic.geocities.com" using HTTP, all further connection attempts by that host are logged. The trace.out generated by the system is shown below.

Trace File : mail.trace

The above is the captured traffic file against which s_ excercise5.bro policy script will be experimented. The following is the command to see the results:

bro -r mail.trace s_excercise5.bro

Trace File: mail.trace [Bro workshop 2007]

The above is the captured mail traffic file against which s_ smtp.bro policy script will be experimented . The following is the command to see the results:

bro -r mail.trace s_smtp1.bro

Result:

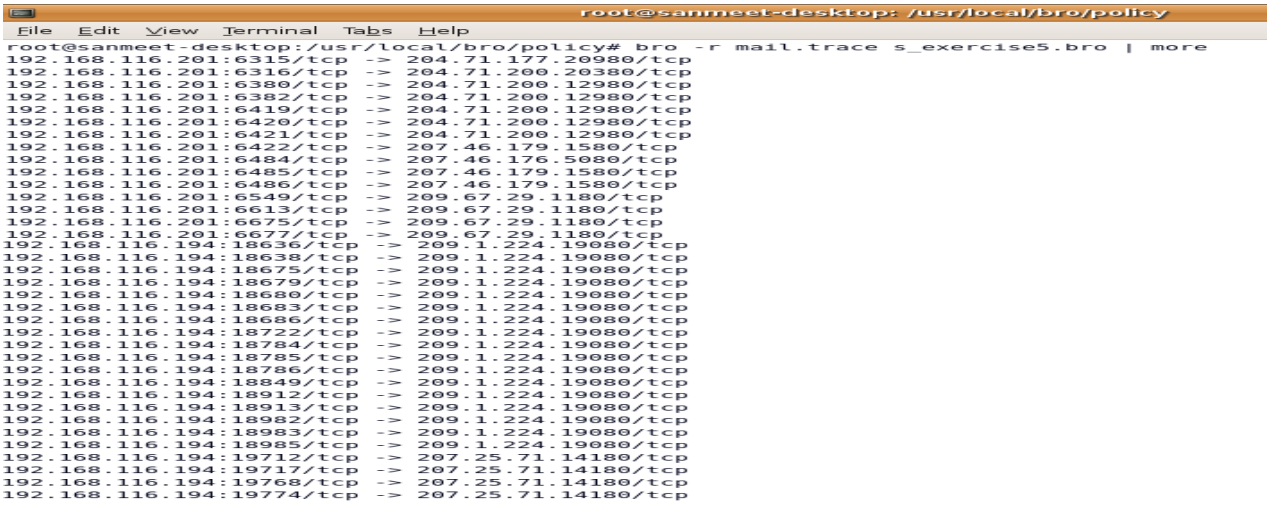

Figure 6: Logging all connections that attempts to access pic.geocities.com

3.6: Logging connections those attempt to access pic.geocities.com in a file instead of stdout

In this task we have captured HTTP and other network traffic from multiple clients. Once it is detected that somebody is trying to access pic.geocities.com using HTTP, all further connection attempts by that host are logged and written to a file, instead of stdout. The trace.out generated by the system is shown below.

Trace File : mail.trace

Result:

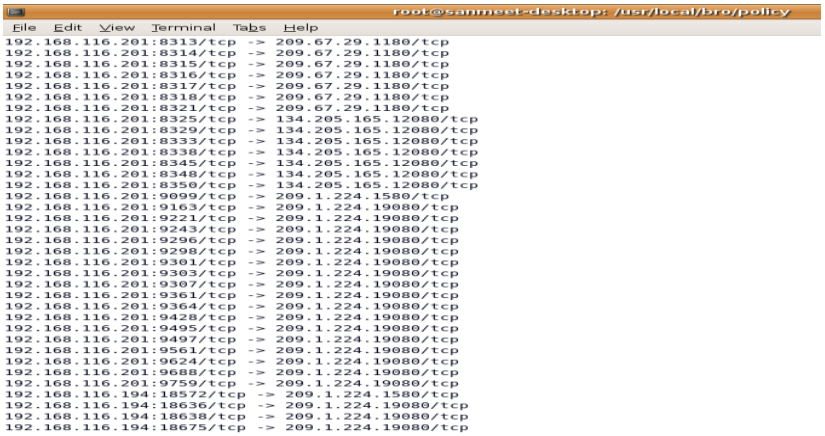

Figure 7: Logging connections those attempt to access pic.geocities.com in a file instead of stdout

3.7 Detecting all GTalk traffic in a captured file using a script

In this task Gtalk traffic is detected out of the captured traffic. There is no event for this. Here signatures are used. We have detected this with a captured traffic in some trace File.

Trace File: gtalk1.trace

The above is the captured traffic file against which s_gtalkcap.bro policy script will be experimented. The full packets can't be captured by this approach. Only contents of headers will be captured. Those contents are stored in a file called s1.trace.The following are the commands to see the results:

The above is the captured traffic file against which s_ excercise6.bro policy script will be experimented. The following are the commands to see the results:

1) *bro -r mail.trace s_exercise6.bro*

2) less http.log

bro -r gtalk1.trace s_gtalkcap.bro

Result:

Figure 8: Output showing Gtalk traffic

To see the captured content file less command can be used like below:

less s1.trace

Result:

```
POST /mail/channel/bind?at=xn3j34s9gsm5bi63e2y4c9fxfi99f5&VER=6&it=73003&SID=77C408F0F735F59C&RID=46858&zx=3looxtnl6xt5&t=1 H
TTP/1.1
Host: mail.google.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.3) Gecko/20061201 Firefox/2.0.0.3 (Ubuntu-feisty)
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Referer: http://mail.google.com/mail/?ui=2&view=js&name=js&ids=vsdmtnrembkr
Content-Length: 57
```

Figure 9: Tracefile with content information of packet headers

3.8 Detect all packets of live GTalk traffic using a script

In the previous exercise not all information about packets but only headers information is stored. Detect all packets of live GTalk traffic using a script.

In this task no trace file is there as we have experiment it on live traffic. So *bro –i* will be used. Following is the complete command to capture the packets in a trace file:

bro -i eth0 -w a.trace s_gtalklive.bro

To see the output captured in the tracefile a.trace we can not simply use *less* command because it's a binary file. However, we can use *tcpdump* command to see the captured packets. Command to See the Captured gtalk Traffic**:**

tcpdump -r a.trace

Result:

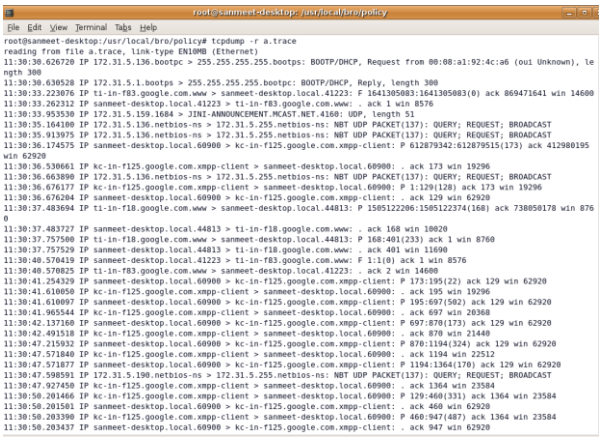

Figure 10: Captured packets with Gtalk traffic

3.9 A Bro script, where the user can maintain a list of URLs, if any of the URLs are hit, log the connection to a file

Trace File: trace.out

The above file is captured traffic file against which the script s_connlog.bro will be experimented. The following command will be used to run the script:

bro -r trace.out s_connlog.bro

The output is logged in a file instead of stdout. The following command is used to see that log file.

less http.log

Result:

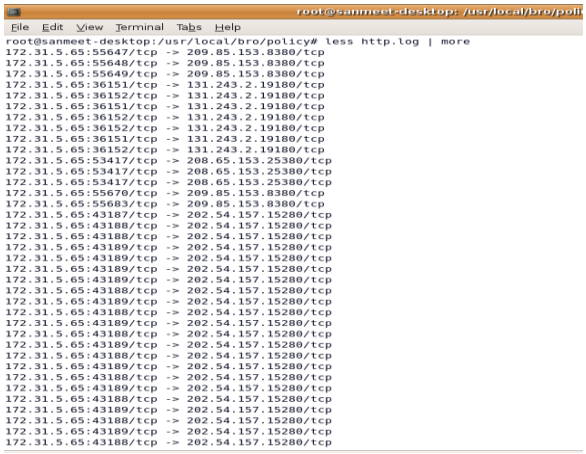

Figure 11: Connections attempting to access restricted URLs

4. CONCLUSION AND FUTURE WORK

Computer networks have brought the world together by bridging the information gap among people. Network technology has undergone a revolution with better and faster ways of sending information between computers. Unfortunately security systems and policies to govern these networks have not progressed as rapidly. Today's network is very complex and the whole world is focusing on ease of use and functionality. This is making us more insecure. For hackers, these well-traveled paths make networks more vulnerable than ever before and with relatively little expertise hackers have significantly impacted the networks of leading brands or government agencies. Cyber crime is also no longer the prerogative of lone hackers or random attackers. Today disgruntled employees, unethical corporations, even terrorist organizations all look to the Internet as a portal to gather sensitive data and instigate economic and political disruption. With networks more vulnerable and hackers equipped to cause havoc, it's no surprise that network attacks are on the rise. So there is a huge need of detecting the threats and intrusion. For this purpose number of solutions is there, IDS is one of them. Bro is the most popular and effective IDS which can be used to detect these threats.

In this paper we have explored and designed the policy scripts of Bro to detect various kinds of traffic like web traffic, mail traffic, web mail traffic etc. The scripts are experimented against captured traffic as well as live traffic. However no IDS can detect all the intrusions. So we need a combination of various techniques.

5. REFERENCES

[1] Bro: An Open Source Network Intrusion Detection System, Robin Sommer, Computer Science Department, TU Munchen, Germany, 2003

[2] Bro Workshop 2007, http://www.bro-ids.org/bro-workshop-2007

[3] Erricsson, "Managing Network Security", White Paper, October 2006

[4] Network Protocols Handbook, Second Edition, Javvin Technologies,Inc.

[5] P. Boer & M. Pels, "Host-based Intrusion Detection Systems", February 2005

[6] S. McCanne, C. Leres and V. Jacobson, libpcap, available via anonymous ftp to ftp.ee.lbl.gov, 1994

[7] Theuns Verwoerd , "Active Network Security", Honours Report 5 November, 1999

[8] Vern Paxson, Flexible, High-Speed Intrusion Detection Using Bro, Berkley, CA, USA, 2004, http://www-nrg.ee.lbl.gov/bro.html

[9] Vern Paxson , Bro: A System for Detecting Network Intruders in Real-Time, Lawrence Berkeley National Laboratory

[10] Wayne T Work, "Intrusion Detection Systems", Security Gauntle Consulting,Naugatuck

[11] William Stalling, Cryptography and Network Security Principles and Practices, Pearson Education, 2002

[12] Wireshark, Man Page