BRO - an Intrusion Detection System

ROGER LARSEN
Project - IMT4741 Intrusion Detection and Prevention
Gjøvik University College

Friday 30th September, 2011

Abstract

Network Intrusion Detection Systems (NIDS) have existed for several decades. The last 10 years they have also been a common security equipment in many companies/organisations. With the ongoing challenging cyberwar we need these kind of "smart" software to win the battle – a battle we can not loose! Bro is a very flexible, powerful and scaling Intrusion Detection System that have a different approach than "traditional". In addition to this it is open source. I have in this article described IDS in general and Bro NIDS more in details.

Categories and Subject Descriptors

C.2.0 [Computer Systems Organization]:
Computer-Communication Networks – General, Security and protection (e.g., firewalls).

General Terms

Network Intrusion Detection System

Keywords

Bro

Contents

1 INTRODUCTION

1.1 Historical Background

In the early computer days there was these large mainframe computers with console and printers directly connected to them. They often printed out any audits/log and/or error messages on the attached printer when incidents/events occur. An experienced computer administrator would then analyse these printouts and could often tell the reason for these audits/logs. But often these analysis could take days and/or weeks.

James P Anderson published a study in 1980 which described how to improve computer security auditing and surveillance. He had several ideas in developing a more automatic event/log analysing system [7] [3]. This was the start of the intrusion detection systems.

1.2 Intrusion Detection Systems in real-life

The classical security mechanisms in companies for many years now are as follows; (i) antivirus (client and servers), (ii) firewall and (iii) SPAM-filter. This have been the typical investment regarding physically infrastructure. The next level in computer security in companies is the use of a intrusion detection system (IDS). This IDS can be a single installation (one box) or it can have several so-called sensor slaves (collecting data). This IDS are typical well known of every signature/sign of evil traffic/payload. When an attack and/or unwanted traffic pattern occurs - the IDS can block this traffic in the firewall and/or router. Figure 1 illustrates the principle in how a IDS can help securing our computer network.

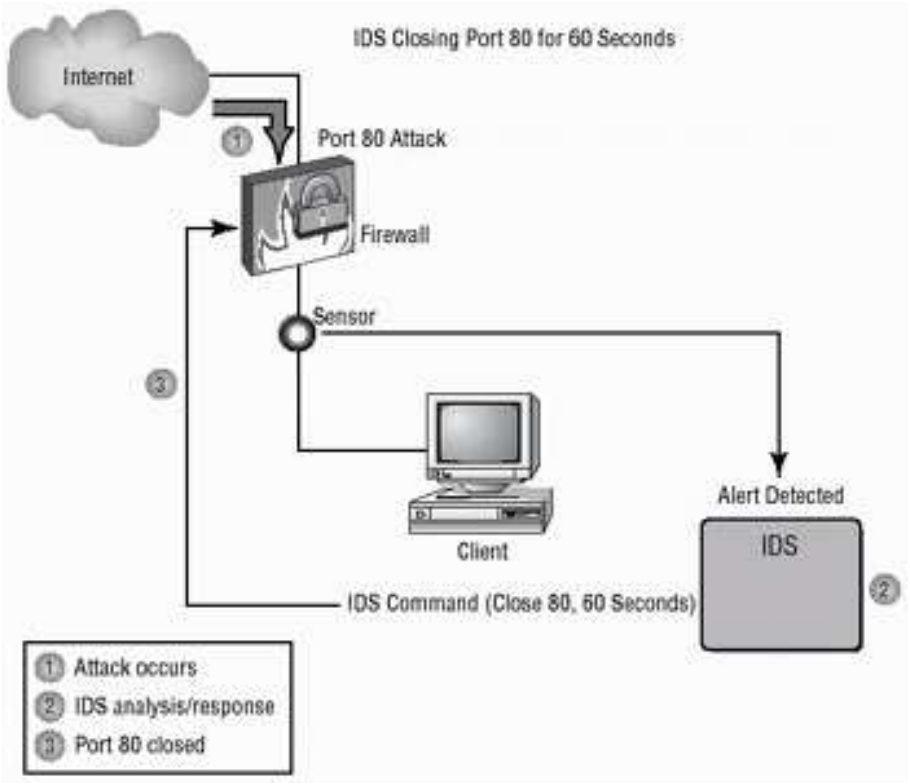

Figure 1: IDS in practice.

The IDS can with the use of one or more sensors be used by companies in every sizes. This is an important parameter nowadays when the use of Internet grows every day. Intrusion detection systems are normally not an out-of-the box product. This is mainly because every network have more or less its own distinct traffic pattern. The installation costs can in this matter be a show-stopper for companies that will increase their security level/control. By the use of proper risk management and planning the initial costs will be reasonable. I personally think this kind of system is unavoidable in the ongoing battle against computer crime.

1.3 Intrusion Detection Systems Today

IDS have become a complex piece of software. IDS includes many advanced techniques nowadays. A list of some of these techniques are listed below:

- Advanced Statistical functionality
- Advance Data Mining/Pattern Recognition functionality
- Machine Learning (Artificial Intelligence) methods
- Multiple sensors/Tier/Cluster functionality

This makes IDS capable of learning its environment (network) as it works - which makes it prepared for blocking unknown attacks in our computer networks [5].

1.4 Classification of IDS

Host-based
- Collects data from sources in the operating system, e.g. system and events logs.
- Monitors users and/or systems activity (execution of programs, data exchange et.al).
- Monitors network traffic/data that enters and leaves the actual host.

Network-based
- Collects network traffic/data as source data.
- Monitors and analyse network traffic/data in real time.
- Can filter out/remove unwanted network traffic in real time (in-line mode by using two network interfaces [1]).

Application-based
- Collects data from specific applications, e.g. application logs.
- Monitors application activity (data exchange et.al).

Target-based
- Creates its own checksum data (by adding code) in data traffic.
- Verifies integrity of data traffic.
- Monitors and analyse system calls within monitored applications.

1.5 Detection models

Signature/Misuse Detection

IDS that are preloaded with known suspicious patterns/signatures are called *Misuse Detection Models*. Some call these IDS signature models. They are known to be very efficient and with low false rate.

Anomaly

The other kind of IDS model collect networks statistics and define the most normal network behaviour as a baseline. When this baseline are challenged in large degree (large deviation), the alarms go off. This kind of IDS are called *Anomaly Detection Model*. This kind of IDS can produce large number of false positive, but is in the same time very efficient against new kind of network attacks.

1.6 This article

Bro have not updated the general documentation on their web page [19]. Bro spesific material are partly based on several workshops presentations in 2009 [17]. I asked the Bro community for newer (and perhaps more detailed) documentation without success. The answer was; *The Bro team is overhauling their documentation.*

I start this article with some background and general information regarding IDS. Further I describe Bro NIDS in more details, the developers philosophy/design and especially the Bro policy script language. I finally describe some useful and powerful components/Add-Ons for Bro.

1.7 My Motivation

My motivation with this paper is to learn more about intrusion detection systems. I hope writing a master thesis in intrusion detection systems. These kind of security systems/software match my working experience in large degree.

[1] Note! An Network-based IDS in in-line mode in sensible to bandwidth issues.

2 BRO - an Intrusion Detection System

2.1 Background

Bro is a network-based IDS (also called NIDS) that initial was developed by Vern Paxon in 1999 [12] in International Computer Science Institute in Berkeley. He still leads this project together with a team of researchers and developers from International Computer Science Institute in Berkeley and the National Center for Supercomputing Applications in Urbana-Champaign, both USA [19].

Bro is short for the term *"Big Brother"* from the classical book; "1984 Nineteen Eighty-Four" by Georg Orwell. The link here is that network sniffing/monitoring/analysing tools are often in danger of violate privacy. [2] [11].

2.2 Bro Design Philosophy

Bro was intentionally a stand alone system for detecting network intruders in real time. The research team lead by Vern Paxson was aware of existing commercial products, of course, but the details and knowledge in building these kind of network monitors was not public known in 1999.

Bro's developers brags about bridging the gap between academia and commercial/operational NIDS for 15 years [19]. Here are a list of The Bro Project's main focus/philosophy [13]:

- Real-time network analysing framework
- Separate packet collector mechanism from policy/analysis mechanism (avoid packet filter drops et.al)
- Neither anomaly or misuse/signature architecture
- Capable of analysing high-performance networks in large scale
- A script language that helps operators avoid mistakes (because of its simplistic structure)
- Keep application-layer state in its analysing networks
- Analyse high-level semantics in application layer with existing analysers
- Comprehensively log facilities (which makes forensics community pleased)
- Open interface to exchange data to other applications in real-time
- Open Source with BSD Licensing model which makes this software available for free usage in general [2][10]
- Powerful script engines for extensive customisation

2.3 Bro Architecture Explained

The Bro architecture is illustrated in figure 2. Source: [12], [13].

Network

Bro needs a physical network connection to get a copy of the network traffic it will analyse. This is normally done by the use of port mirroring functionality in switches/routers or a TAP device [21] [15].

libcap

`Libcap` is the application `tcpdump`'s packet filtering library [16]. This isolate Bro from the physical network medium in the operating system.

[2] Berkeley Software Distribution License (Regents of the University of California, University of California, Berkeley, 1998)

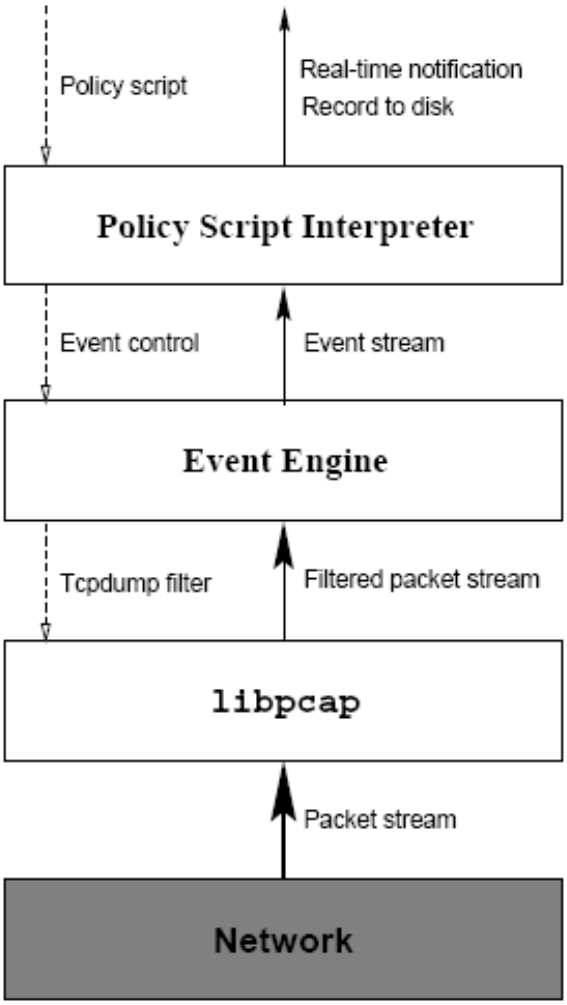

Figure 2: Structure of the Bro system.

Event Engine

The filtered network data packages from `libcap` are then fed into the next level; the `Event engine`. This event engine try to reassemble all the network traffic it gets to known events/patterns as high as possible in the OSI ISO Model [4]. Typical the event engine will find connection attempts (transport-level), FTP requests/replies, HTTP requests/replies (application-level) and login failed/success (application level). In October 2009 there was about 320 types of these known events that the Bro's event engine could identify [17].

The event engine performs several health checks and try to reassemble the packages:

1. Integrity checks (are the packet headers intact? are the IP packet headers correct regarding checksums? etc.)
2. If integrity checks $\neq$ OK; write an error event + discard packet
3. If IP packets; Reassemble IP fragments into datagrams
4. If integrity checks = OK; look up the connection state with associated; (i) source IP address, (ii) destination IP address and (iii) TCP or UDP port numbers
5. Dispatch the packet to a connection handler (TCP or UDP) for the further corresponding connection

TCP connection handler For every TCP package the connection handler performs the following actions;

1. Verify the TCP header
2. Verify the TCP packets checksums for header
3. Verify the TCP packets checksums for payload
4. If verification above = OK; Are there any `SYN/FIN/RST` control bits/flags?
5. If flags above are present; set the actual connection state to the active control bit/flag.
6. Process other data acknowledgement in header (if any)
7. Process payload data (if any)

UDP connection handler The UDP connection handler are similar to TCP but much simpler because it is connection less (e.g. no connection state). But – UDP sessions use different ports when starting a UDP packet stream then replying this UDP stream. This states are called pseudo connection states.

Tracking-/Packet Record file The event engine creates a `tcpdump` network trace file. The connection handlers acknowledge to this trace file writing after their process if the actual package was; (i) highly interesting (the whole package are then saved), (ii) medium interesting (header of package saved) or not interesting (package history dropped).

Supported Protocols/Applications Bro supported the following protocols/applications due to October 2009; ARP, IP, ICMP, TCP, UDP, BitTorrent, DCE-RPC, DHCP, DNS, FTP, Finger, Gnutella, HTTP, IRC, Ident, NCP, NFS, NTP, NetBIOS, POP3, Portmapper, RPC, Rsh, Rlogin, SMB, SMTP, SSH, SSL, SunRPC, Telnet.

Policy Script Interpreter

The policy script interpretor processes events from the event engine. For every event handled to the policy script interpretor – it performs the following steps:

1. Look up the corresponding event handler's (semi-)compiled code/script
2. Bind the value(s) of the event(s) to the argument of the handler
3. Interpret the actual event code/script

The policy script interpreter is in general a event handler. The result of this process can execute further scripts/commands including; (i) generate new events, (ii) log events, (iii) invoke other event handlers.

Bro ships with a large number of ready made policy scripts for various analysis. When adding new functionality to Bro it is writing a new protocol analyser to the event engine and/or writing a new event handler in the policy script interpreter.

2.4 Script Languages

Bro have an extensive script language that are very flexible and powerful. This script language makes Bro very customisable. It is in a "C" like syntax. The Bro script language are very network focused, which is a very useful in this context of course. Please note that the following description of the Bro script language may not be complete. This is much because the most detailed documentation was the original article by Vern Paxson in 1999 [12] .

2.4.1 Data Types & Constants

Bro policy script language supports several types familiar to traditional script languages. Please find an overview in table 1:

Bro Policy Script Language – Data Types			
Data Type	Description	Examples	"C"-version
`bool`	boolean	T / F	
`int`	integer		similar
`count`	non negative integers		unsigned
`double`	large integers		similar
`string`	text/series of bytes	http	similar
`time`	absolute time	30 min	
`interval`	difference in time		
`port`	TCP or UDP port number	'80/tcp'	
`addr`	IP address	192.168.100.20	
`hostname`	hostname	'myserver.hig.no'	
`record`	a collection of elements of arbitrary types		
`file`			
`list`	holds zero or more instanses of a value		
`pattern`	unix-style regular expressions		
`table`	table		
`set`	set of elements	{0,1,2,3..,..35,36}	

Table 1: Bro Policy Script Language – Data Types.

2.4.2 Operators

Table 2 shows the operators in the Bro policy script language.

Bro Policy Script Language – Operators			
Operator	Description	Examples	"C"-version
`+, -, *, /, %, !, &&, ||, ?:, relationals like <=`	general operators		similar
`++`	increment		
`--`	decrement		
`in`	infix operator "in"	`[src_addr, dst_addr, serv] in RPC_okay`	
`!in`	infix operator "not in"		

Table 2: Bro Policy Script Language – Operators.

2.4.3 Predefined Functions

Table 3 shows the predefined functions in Bro Policy Script Language.

Bro Policy Script Language – Predefined Functions	
Function	Description
`fmt`	*sprintf*-style formatting for use in printing and manipulation
`edit`	returns a copy of a string that has been edited using the given editing characters
`mask_addr`	takes an `addr` and returns another `addr` corresponding to its top n bits
`open`	open files
`close`	close file
`network_time`	returns the timestamp of the most resent received package
`getenv`	reads local environment variables
`skip_further_processing`	stops/breaks any further analysis on connection/packet etc.
`set_record_packets`	instruct the event engine to record (or not) any future network packets
`set_contents_file`	
`system`	execute a string as a unix shell command
`parse_ftp_port`	takes an FTP 'PORT' command and returns a `record` with the corresponding `addr` and `port`

Table 3: Bro Policy Script Language – Predefined Functions.

2.4.4 Variables

Bro script language supports two kind of variables; `local` or `global`.

- `local` – variables in a function or a event handler
- `global` – variables globally to the entire Bro script

For both of these variable classes you can declare them as `const` – the variable are then constant, and can not be changed.

The variable declaration syntax:

```
{class} {identifier} [':' {type}] ['=' {init}]
```

Example on defining variables:

```
const ftp_serv = { ftp.lbl.gov, www.lbl.gov };
```

2.4.5 Statements

Bro script language supports a rather modest collection of statements. You will e.g not find loop statements here – this is mainly because the fear of time delaying processing (the need for speed in capturing and analysing network traffic). Table 4 shows the statements in Bro policy script language.

Bro Policy Script Language – Statements	
Statement	Description
`if`	expression evaluation
`return`	expression evaluation
`print`	print a list of expressions to a file (*stdout* by default)
`log`	log a list of expressions
`add`	add an element to a set
`delete`	delete an element from a set or a table
`event`	generate a new event

Table 4: Bro Policy Script Language – Statements.

Statements can be can grouped into blocks by using {}'s. See text box below as an example to this. Sources: [14] [12]

```
global ssh_hosts: set[addr];
event connection_established(c: connection) {
    local responder = c$id$resp_h; # Responder's address
    local service = c$id$resp_p; # Responder's port
    if ( service != 22/tcp )
    return; # Not SSH.
    if ( responder in ssh_hosts )
    return; # We already know this one.
    add ssh_hosts[responder]; # Found a new host.
    print "New SSH host found", responder;
}
```

2.5 Notice Action Filters and Notice Policy

Bro ships with a large number of policy scripts. The goals with these scripts are typical to notice us and/or log the traffic for suspicious/unwanted behaviour. In Bro terminology it is called to rise a NOTICE or call the NOTICE function. All Notices are grouped by their characteristic property/protocol/environment/attack type et.al. and mapped into predefined actions. This is done by so-called *Actions*. Table 5 shows Bro's current defined actions:

Bro Notice Actions	
Action	Description
NOTICE_IGNORE	Ignore Notice completely
NOTICE_FILE	File Notice only to notice.log; do not write an entry into alarm.log
NOTICE_ALARM_ALWAYS	Report in alarm.log
NOTICE_EMAIL	Send out a mail and report in alarm.log
NOTICE_PAGE	Page security officer and report in alarm.log
NOTICE_DROP	Block connectivity for offending IP and report in alarm.log

Table 5: Bro Notice Alarms.

We filter out these notices in two levels; first by using so-called *Notice Action Filters* and then by *Notice Policy*. This is for greater flexibility and correlation of handling the notices.

2.6 Signature Language

Bro have in addition to its powerful script language an signature language. This signature language is not the preferred detection tool, but it is much more human understandable then the script language and is almost similar to several other NIDS systems [18].

The Signature Format

Every signature has this format: *signature* $< id > <$ *attributes* $>$.

$< id >$ is a unique for every individual signature.

$<$ *attributes* $>$ can be two types: *conditions* and *actions*.

The conditions defines the signature matches – and the actions condition speak for itself; what action / what do we want to execute when the condition is true.

Conditions can be four main types: *header, content, dependency* and *context*.

The text box below shows an example on signature language.

```
signature my-first-sig {
    ip-proto == tcp
    dst-port == 80
    payload /.*root/
    event "Found root!"
}
```

2.7 Research

Bro are in large scale used as a NIDS research platform. The technical capabilities and that Bro initially started as an research project may be the main reason for an extended research activity regarding Bro. In addition most developers and researchers are located at International Computer Science Institute [6].

Today Friday 30th September, 2011 there have been published 12 articles with Bro as a major part and there are current 4 ongoing research projects [20].

- HILTI: A High-level Intermediary Language for Traffic Analysis
- BinPAC++: A Next-Generation Parser Generator for Network Protocols
- Exploiting Multi-Core Processors for High-Performance Traffic Monitoring
- Establishing a Cross-Institutional Platform for Cooperative Security Monitoring and Forensics

3 Bro Add-On/Supplemental Applications

Bro have some important add-on/supplemental applications available. Some of them are included in the distribution of Bro.

3.1 Broccoli

Broccoli – The **Bro C**lient **Co**mmunications **Li**brary. Broccoli makes non Bro instances communicate and/or control them. Broccoli is an communication API [3]. Broccoli application will typically offer the following services/:

- Configure/control tasks in Bro
- Interface Bro to other systems (eg. monitoring and alerting systems)
- Send network traffic from host-based sensors to Bro

Broccoli is bundled with Bro. Broccoli is also licensed under BSD License [10] [8].

3.2 BroControl

BroControl is a interactive shell for communication with and control the Bro installation. The BroControl are installed in two modes; (i) Stand alone or (ii) Cluster. In Cluster mode BroControl have many useful statistics in its log file.

3.3 Time Machine

Time Machine - High-Performance Packet Recording. Time Machine is an application that can save/record the entire network data/stream raw. This makes it possible to go back in time (aka Time Machine) regarding the network data. When analysing network data we normally filter out a lot of data. With Time Machine we can in retrospect replay the network traffic when we typically have narrowed our filters/policy scripts/analysis et.al. Computer forensics loves this piece of software. The Time Machine application is a developed by a project from Technische Universität Berlin, the Technische Universität München, and the ICSI (University of California Berkeley). It is licensed under the BSD license.[9] [1].

[3] API = Application Programming Interface

3.4 Bro Cluster

Bro Cluster - Operating a high-performance cluster of Bro systems. Bro Cluster makes the Bro installation (i) scalable, (ii) flexible) and (iii) load balancing. Bro Cluster system consists of four different kind of nodes (roles);

- Frontend(s) – more then one for load balancing
- Worker node(s) – for processing the analysis on their slice of network traffic which is controlled by the frontend(s).
- Proxy(ies) – which relay the communication between the worker nodes
- Manager – a managing user interface for control and logging

References

[1] The time machine. URL, July 2011.

[2] Jon Bashor. Software that detects hackers helps catch big league intruder. URL, March 2000.

[3] Guy Bruneau. The history and evolution of intrusion detection. URL, 2001.

[4] J. D. Day and H. Zimmermann. The OSI reference model. *Proceedings of the IEEE*, 71(12):1334–1340, 1983.

[5] Mohammad A. Faysel and Syed S. Haque. Towards cyber defense: Research in intrusion detection and intrusion prevention system. In *IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.7, July 2010*, volume 10. University of Medicine and Dentistry of New Jerse, 2010.

[6] International Computer Science Institute. About icsi. URL, September 2011.

[7] James P Anderson. Computer security threat monitoring and surveillance. April 1980.

[8] Christian Kreibich. Broccoli - the bro clinet communication library. URL, March 2011.

[9] Gregor Maier, Robin Sommer, Holger Dreger, Anja Feldmann, Vern Paxson, and Fabian Schneider. Enriching network security analysis with time travel. *SIGCOMM Comput. Commun. Rev.*, 38:183–194, August 2008.

[10] Open Source Initiative OSI. The BSD License. URL, March 2010.

[11] George Orwell. *1984 Nineteen Eighty-Four*. Penguin Classics, new ed edition, January 2004.

[12] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, pages 2435–2463, 1999.

[13] Vern Paxson. Bro Workshop 2009, Design. URL, October 2009.

[14] Vern Paxson. Bro Workshop 2009, Scripting Language. URL, October 2009.

[15] F. Risso and L. Degioanni. An architecture for high performance network analysis. In *Computers and Communications, 2001. Proceedings. Sixth IEEE Symposium on*, pages 686 –693, 2001.

[16] Tcpdump/Libcap. Tcpdump/libcap public repository. URL, September 2011.

[17] The Bro Project. Bro workshop 2009, the 2nd. URL, October 2009.

[18] The Bro Project. Signatures. URL, October 2009.

[19] The Bro Project. The Bro Network Security Monitor. URL, September 2011.

[20] The Bro Project. Research. URL, September 2011.

[21] Jian Zhang and A. Moore. Traffic trace artifacts due to monitoring via port mirroring. In *End-to-End Monitoring Techniques and Services, 2007. E2EMON '07. Workshop on*, pages 1 –8, 21 2007-may 21 2007.