Protocol Detection Capabilities in Bro

ROGER LARSEN

Project - Spring 2012
IMT4022 Digital Forensics II
Gjøvik University College

Thursday 21st June, 2012

Abstract

Network Intrusion Detection Systems (NIDS) focus generally on 3 main detection methods; (i) signature, (ii) anomaly network traffic behaviour, and (iii) protocol analyses. The challenge in protocol analyses is to detect the correct protocol used and initiate the proper analyzing method(s). The TCP/IP suite have a standard scheme which predefines port numbers for each protocol by IANA. However, both benign and evil software are continuously getting more and more sophisticated and do not follow these predefined rules. Bro is a open source framework for network traffic analyses[1] [33]. We will in this paper focus mainly on protocol analyzing mechanisms in Bro. With Bro we can analyze network traffic and detect odd protocol/port pairs usage. We will also look at what Bro can contribute regarding digital forensics. Does the present version of Bro[2] hold any machine learning functionality?

1 Introduction

Traditional security mechanisms like antivirus and firewall are today not enough in the battle against malware and malicious attacks. We need smart network filtering mechanism that can detect unknown attacks both from Internet (untrusted) and internal (trusted). Network intrusion detection system (NIDS) will definitely play an important role in future information security mechanisms.

1.1 Motivation / Limitation / Outline

Our motivation for this paper is to explore Bro's protocol analyzing and digital forensics capabilities. The amount of research projects that have used and use Bro tells us that this is a interesting piece of software. Hopefully we will also find some interesting machine learning functions.

We focus mainly on Bro and its capabilities. This project is part of a course in Digital Forensics II, spring 2012 (5 ECTS [3]).

[1] Bro (or the long version: "The Bro Network Security Monitor") are Unix-based software that are capable of passively monitor network traffic for suspicious behaviour.

[2] Bro version 2.0 released Jan 11, 2012.

[3] ESTC - European Credit Transfer and Accumulation System `http://ec.europa.eu/education/lifelong-learning-policy/ects_en.htm`

We start this article with a general description of the NIDS and especially Bro. Further we explain Bro's protocol analyzing functions and contributions in digital forensics (including possible machine learning functions).Finally we discuss and summarize our findings.

1.2 Background

NIDS

NIDS are network based IDS systems that filters and/or check the computer traffic for unwanted activity/threats. We often classify the NIDS in two kind of main categories; (i) misuse/signature and (ii) anomaly [24] [20] [2].

The misuse/signature based NIDS works principally much like an antivirus software. The misuse/signature NIDS are preloaded with known suspicious patterns/signatures. They are known to be very efficient and with low false positive. It is important to notice that this kind of NIDS may not prevent/block new, unknown network attacks. A well known misuse/signature based NIDS are SNORT®[4]. Snort are much used and exists in both community and commercial versions[27].

The anomaly based NIDS collect networks statistics and define the most normal network behaviour as a baseline. When this baseline are challenged in some degree (by deviation), the alarms go off. This kind of NIDS can produce large number of false positive, but may be very efficient against new, unknown network attacks. Bro [33] is a often referred to as anomaly based NIDS.

Machine Learning (ML)

Intelligence in technology may sometimes be a subjective opinion - but in this context we will use the term to describe computer software that have learning capability [25]. We often use the term ML when computer equipment and/or software have learning capabilities [7] [3]. We split ML into two main categories: (i) supervised and (ii) unsupervised.

In supervised learning we have a great deal of knowledge of the data we are dealing with. We can pre-set labels versus parameters in these learning functions and use statistical approach to classify/analyze data to decide what actions to execute.

In unsupervised learning we only have minor knowledge of the data we are dealing with. We often use complex statistical algorithms, pattern recognition and data mining techniques to label the data we analyze. [36][10].

A typical example where ML have succeeded are the process of grouping email into unwanted/unsolicited email (SPAM) and normal email (non SPAM).

1.3 Bro Design

Bro is a framework for network analyses and are very popular in many research groups[30]. Bro's design principals are flexibility and efficiency regarding traffic analyses [19]. We have included a figure to illustrate the architecture of Bro with explanation below. See figure 1.

Bro's pure and simple design (without challenging flexibility) have for many years resulted in many scientific articles/projects in research communities [30].

[4]Snort is a trademark of Sourcefire, Inc. USA.

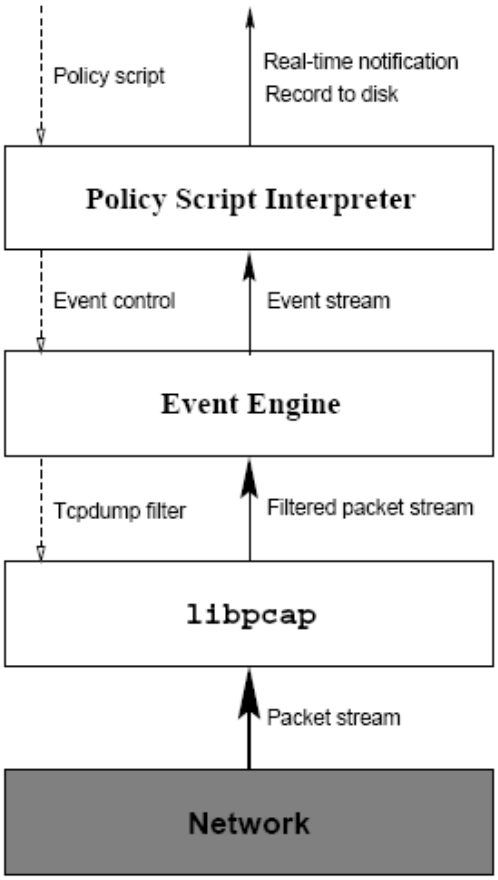

Figure 1: Bro's Principal Design.

Network Bro needs a physical network connection to get a copy of the network traffic it will analyze. This is normally done by the use of port mirroring functionality in switches/routers or a TAP device [37] [23].

Libcap `Libcap` is the application `tcpdump`'s packet filtering library [28]. This isolate Bro from the physical network medium in the operating system.

Event Engine The filtered network data packages from `libcap` are 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 TCP/IP OSI model[8] - see figure 3. 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 [31].

The event engine performs several (packet) health checks and try to reassemble the packet:

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 + drop packet (from further analyses in Bro)
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

For every TCP packet that arrives the event handler the *TCP connection handler* performs the following actions:

1. Verify the TCP header
2. Verify the TCP packets checksum for header
3. Verify the TCP packets checksum(s) 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)

For every UDP packet that arrives the event handler the *UDP connection handler* reacts. This handler are similar to TCP but much simpler because it is connection less (e.g. no connection state). However, UDP sessions use different ports when starting a UDP packet stream then replying this UDP stream. These states are called pseudo connection states.

The event engine creates a `tcpdump` network trace file. The connection handlers will acknowledge to this trace-file their packet status when processed; (i) highly interesting (the whole package are then saved), (ii) medium interesting (header of package saved) or (iii) not interesting (package history dropped).

Policy Script Interpreter If the event handler raise the interesting flag (medium or high), the *Policy Script Interpreter* will further examine/analyze the packets. Bro have specialized language (Bro language) written scripts that it will execute. These actions will typical be one or several of the following; (i) execute analysis, (ii) log event, (iii) raise an alert and/or (iv) update statistics [19].

2 Protocol Analysis

2.1 Challenges

Protocol analysis is a challenge for many reasons: (i) protocols have no standard identification, (ii) protocols may use other ports then defined by IANA[5][5] and (iii) encapsulation. Traditionally, protocols have been identified my their port usage. Today, ports and applications are not used as defined by IANA both by benign and malicious reasons. Network administrators change the default ports configuration for their applications/servers in their firewalls and other network equipment to hopefully increase their security level.

Some protocols can be very easily detected (e.g. FTP) but other can be more difficult and more analysis are needed (e.g. telnet/DNS). We need in general a signature database to be able to detect different protocols.

New application tend to be very intelligent in finding open ports from an users point of view. A typical example on this is the popular Internet VoIP application Skype©[6] [22]. However, we need some ports open to use basic services in Internet. Typical TCP port 80 are always available (HTTP) and Skype will quickly find this port open and use these common ports for its services.

[5]Internet Assigned Numbers Authority (IANA) is globally responsible of coordinating Internet protocol resources.

[6]Skype is a chat, voice and video telephone software owned by Microsoft©.

2.2 Related Work

Protocol analyzing function in computer network software is a hot topic in many research communities.

Michael Mai finished in 2005 his Master Thesis with the title *"Dynamic Protocol Analysis for Network Intrusion Detection Systems"*[13]. This thesis describes an approach called *Application Layer Switch Analyzer* (ALSA) using Bro as a platform for their implementation.

Dreger et.al wrote in 2006 an article called *Dynamic Application-Layer Protocol Analysis for Network Intrusion Detection* [6] which describes another *protocol identification analyzer* (PIA) which most likely are implemented in Bro (much similar name, functionality and description in Bro documentation and source code).

2.3 Protocol Analysis in Bro

Supported Protocols/Applications Bro supports the following protocols/applications due to Thursday 21st June, 2012 (in alphabetic order): BITTORENT, DHCP, DNS, FTP, FINGER, HTTP, IRC, MIME, NETBIOS, NETFLOW, NFS, NTP, POP3, RLOGIN, RPC, RPC PORTMAPPER, SMB, SMTP, SSH, SSL, SYSLOG, TELNET, X.509.

Bro's Dynamic Protocol Detection

Bro has a dynamic analyzer framework which associate an analyzer tree with every connection. This design is described in the article *"Dynamic application-layer protocol analysis for network intrusion detection"* by Dreger et al. in 2006 as mentioned above [6]. This tree can enable or disable sub-trees "on the fly" and contain an arbitrary number of analyzers in various constellations. This tree structure can exist the whole lifetime of a connection. This three structure giver high flexibility and an effective model for protocol analyses:

— *Bro can perform protocol analysis independently of ports.* With the use of signatures which match typical protocol dialogues, Bro can look at payload for the correct analyzer and activate it.

— *Bro can disable analyzers if they are parsing the wrong protocol.*

In this way Bro can loosely check for what protocols that are used, and if unsure, use multiple analyzers in parallel. Figure 2 shows an example of an analyzer three.

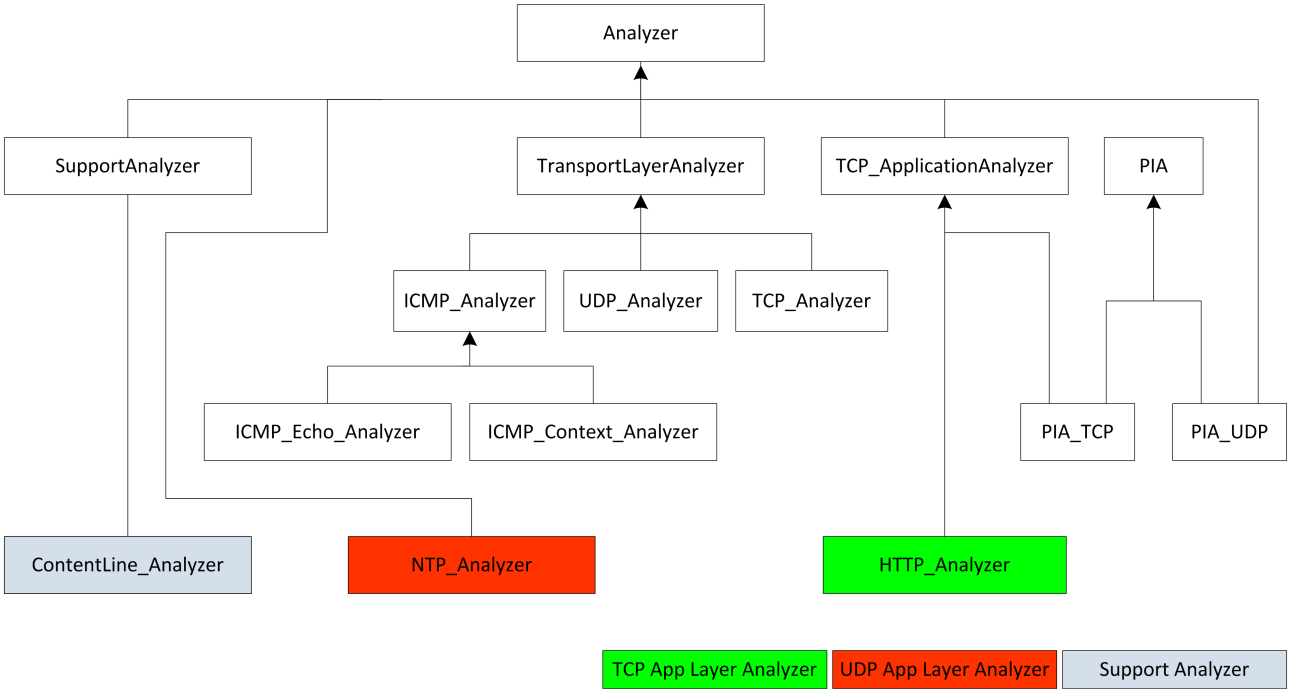

Figure 2: Example of an analyzing tree in Bro. Source: Bro's website [32].

This analyzing tree approach is a per-connection data structure that represents a *data path*, which extracts and track information about what analyzers that should be used. If, typical, Skype are using TCP port 80 this analyzing process will initially check for HTTP traffic (regarding IANA's port and service standard scheme). However, when the protocol signature do not match HTTP, the analyzing process will switch to other protocol signatures/detecting methods.

Every branch under the class Analyzer are different analyzers, they are child processes of the class `Analyzer`. For each connection a so-called *Analyzer Tree* is created. The packets are analyzed by the analyzer's child processes and the results are passed to its successor. Every analyzer process can activate and deactivate other analyzers.

The Protocol Identification Analyzer (PIA) process are identifying the network packets/data stream to initiate the correct protocol analyzer. The PIA process is using a so-called protocol heuristic detection method. When the PIA process finds any match regarding protocols it initiate a new child analysis process. An PIA process and its child processes may continuously change in extent the whole lifetime of the connection (to the actual connection is closed).

For every new connection, a new analyzer tree are initiated with the corresponding TransportLayerAnalyzer. For TCP and UDP there are initiated signature matching processes in search for known protocols.

The analyzers are capable of handle either (i) packet-wise, (ii) stream-wise or (iii) both in combination.

The analyzers in Bro are highly customizable. One can easily change or add new analyzers without great programming knowledge [32].

Bro's BinPAC

The Bro distribution consists of several individual components. `binpac` is a so-called protocol parse generator. We can use high level semantic language to describe our protocol analyzer and let `binpac`[7] generate the C++ code. This makes the urge for continuous change and modifica-

[7] `binpac`'s resent version 0.31 was released 2012-01-09

tion in protocol analysis an mush easier task for us without deep programming knowledge.

Bro source code comes with the following existing `binpac` scripts (in alphabetic order): BitTorrent, DCE RPC Simple, DCE RPC, DHCP, DNS, HTTP, NCP, NetFlow, SMB, SMB-EndPointMapper, SMB-Mailslot, SMB-Pipe, SSL, SYSLOG, SYSLOG[21].

The BinPAC script language is very powerful and flexible and specialized for network packet analyzing purpose. The table 1 shows a summary of the language functions.

Language Construct	Brief Explanation
`%header{ ... %}`	Copy the C++ code to the generated header file
`%code{ ... %}`	Copy C++ code to the generated source file
`%member{ ... %}`	C++ declarations of private class members of connection or flow
`analyzer ... withcontext`	Declare the beginning of a parser module and the members of `$context`
`connection`	Define a connection object
`upflow/downflow`	Declare flow names for two flows of the connection
`flow`	Define a flow object
`datagram = ... withcontext`	Declare the datagram flow unit type
`flowunit = ... withcontext`	Declare the byte-stream flow unit type
`enum`	Define a "enum" type
`type ... =`	Define a `binpac` type
`record`	Record type
`case ... of`	Case type—representing an alternation among case field types
`default`	The default case
⟨type⟩`[]`	Array type
`RE/.../`	A string matching the given regular expression
`bytestring`	An arbitrary-content byte string
`extern type`	Declare an external type
`function`	Define a function
`refine typeattr`	Add a type attribute to the `binpac` type
⟨type⟩ `withinput` ⟨input⟩	Parse ⟨type⟩ on the given ⟨input⟩ instead of the default input
`&byteorder`	Define the byte order of the type and all enclosed types (unless otherwise specified)
`&check`	Check a predicate condition and raise an exception if the condition evaluates to false
`&chunked`	Do not buffer contents of the bytestring, instead, deliver each chunk as `$chunk` to `&processchunk` (if any is specified)
`&exportsourcedata`	Makes the source data for the type visible through a member variable `sourcedata`
`&if`	Evaluate a field only if the condition is true
`&length = ...`	Length of source data should be ...
`&let`	Define derivative types
`&oneline`	Length of source data is one line
`&processchunk`	Computation for each `$chunk` of bytestring defined with `&chunked`
`&requires`	Introduce artificial data dependency
`&restofdata`	Length of source data is till the end of input
`&transient`	Do not create a copy of the bytestring
`&until`	End of an array if condition (on `$element` or `$input`) is satisfied

Table 1: `binpac` language summary.

In table 2 we can see how HTTP protocol are analyzed regarding version and status codes (the table show only a small part of the actual BinPAC script). Pang et al. have written an article called *binpac: a yacc for writing application protocol parsers*[18] which is a good complement to Bro's documentation.

```
type HTTP_PDU(is_orig: bool) = case is_orig of {
        true ->         request:        HTTP_Request;
        false ->        reply:          HTTP_Reply;
};
type HTTP_Request = record {
        request:        HTTP_RequestLine;
        msg:            HTTP_Message(BODY_MAYBE);
};
function expect_reply_body(reply_status: int): ExpectBody
         %{
         // TODO: check if the request is "HEAD"
         if ( (reply_status >= 100 && reply_status < 200) ||
              reply_status == 204 || reply_status == 304 )
                 return BODY_NOT_EXPECTED;
         return BODY_EXPECTED;
         %}
 type HTTP_Reply = record {
         reply:          HTTP_ReplyLine;
         msg:            HTTP_Message(expect_reply_body(reply.status.stat_num));
};
type HTTP_RequestLine = record {
         method:         HTTP_TOKEN;
         :               HTTP_WS;
         uri:            HTTP_URI;
         :               HTTP_WS;
         version:        HTTP_Version;
} &oneline;
```

Table 2: A small part of the `binpac` script `http-protocol.pac`.

2.4 ML functions in Bro

We will in this section look for ML functions in Bro. Sommer and Paxson published in 2010 an interesting article regarding ML in NIDS called *"Outside the Closed World: On Using Machine Learning for Network Intrusion Detection"* [26]. Here they discuss the use of ML functionality in NIDS systems (or rather NIDES system[8] that specially point out that there are ML functionality present. The essence of this article is that ML implementation in NIDS have the following weakness; (i) the need for outlier detection rather then ML classification/similarity function, (ii) very high cost on classification errors, (iii) great variation in adversary data (network traffic) makes solutions unstable and finally: (iv) in general evaluation challenges. They seem in general rather sceptical, but pragmatic to ML in NIDS systems. The article describes recommendations to implementation and evaluation of ML in NIDS.

The ML Hunt

We started our hunt for Bro's ML functions in several stages; (i) read a lot of documentation and articles concerning Bro, (ii) searched on Internet, (ii) searched Bro's community forum [29], (iii) downloaded the source code and read/searched it, (iv) practical test of Bro [12] and finally we ended up asking Bro's community forum [29] directly regarding ML.

The read/parsed pile of C++ code in our search for ML functions gave us the algorithms listed below. The developers of Bro have been good in commenting their coding. We did not use any advanced software in our search for ML functions. The search was mainly done by a Microsoft Windows utility similar to `grep`[9] with keywords like (i) *learning*, (ii) *algorithm*, (iii) *feature*, (iv) *vector*, (v) *heuristic* etc.

The following interesting algorithms were found:

[8] Network Intrusion Detection Expert System

[9] `grep` - a generic text matching command line tool standard installed in Linux/Unix/BSD

- **Radix or Patricia Tree algorithm** — a search algorithms (PATRICIA - Practical Algorithm to Retrieve Information Coded in Alphanumeric. This piece of software is developed by The Regents of the University of Michigan[15]. Bro uses this PATRICIA algorithm to subnet lookup.

- **The Smith-Waterman algorithm** — a maximum likelihood estimation often used in data mining to compare patterns/sequence of data[16]. Bro uses Smith-Waterman algorithm to find overlapping substrings (a built-in function for string search and manipulation).

- **Threshold Random Walk (TRW) algorithm** — an on-line detection algorithm that identifies malicious remote hosts using Sequential Hypothesis Testing[9].

- **Monte Carlo algorithm** — an randomization algorithm used for testing entropy (the average uncertainty).

However, the algorithms above are not categorized as ML functionality. At June 13, 2012 we got an answer from one of Bro developers, Robin Sommer. Unfortunately he could tell us that Bro is lacking ML functionality with operational reliability as their argument. Please find the whole answer (an email) fully referred to in Appendix C. Sommer mention an article we have described in the start of this section[26].

So - we got the not-so-sexy answer: **today, there are no ML functionality in Bro** ...

If further interest, please read our practical tests of Bro in project report, course IMT-4641 Computational Forensics 2012, fall [12].

3 Digital Forensics

3.1 Challenges

The technical challenge in digital forensics in networking are in general the huge amount of data. In our context the more specific sub challenges may typical be; (i) log credibility (ii) clock/timeline credibility (iii) encapsulated/enciphered data and of-course (iv) managing large logs[4] [17].

3.2 Logging

Bro is a very powerful tool regarding live forensics due to its great flexibility in filtering and strong script functionality. We can narrow down and focus on small fragments of large network traffic with Bro's strong filtering capabilities. The extensive logging give us valuable data in historical perspective (though in this context we may talk of seconds). We will focus on logging as Bro's contribution to digital forensics.

Bro have very flexible log functionality. As long as Bro can detect traffic - it can produce logs. Detailed logs of traffic analyzes are generated from the default installation. When Bro is running the log files are located in folder `$ PREFIX/logs/current/`. Here are the logs found when running Bro:

`communication.log` – Bro's own log over main and child processes in addition to some statistics
`conn.log` – connection log: log over every completed connections
`dns.log` – DNS[10] service log

[10]DNS = Domain Name System.

`dpd.log` – log containing Dynamic Protocol Detection log
`http.log` – log containing HTTP activity
`ssl.log` – log containing the analyzing results regarding SSL/TLS handshaking and enciphering establishment process
`weird.log` – log containing unknown/strange activity that is logged for possible later analyzing
`known_hosts.log` – log containing list of hosts that had a complete TCP handshake that actual day.
`known_services.log` – log containing list of IP addresses that had complete TCP handshakes with other hosts on known services.
`software.log` – log containing list of known software detected

The log files are in clear text/ASCII and are organized with columns headings with space/TAB between.

Bro and DataSeries Logging

Bro Team plan to support the Hewlett-Packard *DataSeries*[11] logging format in Bro's next version 5.1[12]. The DataSeries support will get much faster and effective binary logging (than today's ASCII format) in large volumes. DataSeries are a well documented logging standard and comes with libraries/API's for easily analyzing its log in retrospect[1].

3.3 Time Machine

An external software project "closely connected" to Bro is *The Time Machine* (TM) [34]. The TM are able to record raw network traffic for later replay and analysis. The TM are a recording mechanism that can manage gigabit environments. The term "closely connected" refers to that the developers of Bro and TM have coupled their work so Bro can control/operate TM. In this way we can first use Bro to filter out the interesting data traffic we will record and later use Bro to replay this data for further analyzing[11][14].

This is a joint project of the Technische Universität Berlin, the Technische Universität München, and the ICSI (University of California Berkeley). It is open-source and published under the BSD license.

4 Discussion and Further Work

4.1 Protocol Detection Analyzing

Protocol ID Standardization

ML have been introduced with success in many areas - especially medicine. Why are ML so slowly introduced into the computer network area? Our hypothesis is that we (the industry and research communities) try to serve network protocol standards that have a large gap in age/development. A lot of protocols in TCP/IP are almost unchanged since they where developed in early 1960. However, several of TCP/IP protocols have been updated to today's demands. We should develop a protocol standard identification system that would help all of us in this struggle of detecting protocols.

[11]HP DataSeries is a tookit developed by Hewlett-Packard Labs (`http://http://www.hpl.hp.com/`and licensed under BSD license (same as Bro).
[12]According to Bro's BLOG: `http://blog.bro-ids.org/2012_05_01_archive.html`

Controlling Adversary Network Traffic and add ML Functionality with Success?

With our experiences in protocol analyzing and NIDS in this article we are tempted to look at adversary network traffic as a huge unhandy pile of data we desperately need to control. What if we limited traffic from trusted area to untrusted/Internet and visa verse to the kind of traffic that we know/can analyze? In a perfect world we may find several human organized system that can work as an analog to this thought; (i) only certified car drivers and cars are allowed on public roads, (ii) only certified officers may navigate ships in international sea.

NIDS together with firewalls are brilliant filtering mechanisms to control what network traffic that are allowed in an organizations different security zones (DMZ[13], guest hot-spot, internal production network, control rooms etc.). We can have different levels of security filtering with several NIDS slave configured for inline traffic that have the appropriate security policy level for that department. In this way we have a almost controlled environment regarding what protocols that are used. We can now add ML functionality with a much more certainty of success.

However, encapsulation/enciphered traffic may still be a challenge - but we may deny encapsulation and/or enciphered traffic for unusual TCP/IP ports.

4.2 Digital Forensics

4.2.1 Privacy

In NIDS logging the most challenging part is privacy. If we sanitize (wash for privacy) our logs we often loose important information/knowledge. The Europe Parliament decided in March 2006 that they would fight cybercrime with the use of enforced traffic logging for every ISP[14] [35]. Are we actually improving the general information security by this? This is a great dilemma in privacy that is somewhat outside the scope of this article. We do collect a lot of sensitive data using NIDS and need to control access and the use of this kind of huge data logs.

5 Further work

We would like to collect a new up-to-date network traffic dataset to use in the evaluation process. A proper evaluating of a NIDS with ML functionality is crucial for next generation of NIDS/NIDES.

We would like to do a project that challenge odd behaviouring clients with the use of CAPTHA functions - are they humans or evil software? Will we humans cope this kind of redirection when we have installed new software e.g? This kind of interrupt in a clients work both have a technical and a psychological challenge.

6 Summary

We have in this article described Bro NIDS in general and its protocol detection analyzing process (PIA) especially. Bro is a flexible framework for network analysis and have interesting capabilities in digital forensics science. Bro have extensive logging mechanisms and with the third party software "The Time Machine" we can record raw traffic data for later analyzes. However, we did not find any machine learning functionality in present version. The Bro Team

[13]DMZ - DeMilitarized Zone. A medium secured area/zone where we place public available services like; web-, email- and FTP-servers.

[14]ISP = Internet Service Provider

seems to be a competent group of developers, so with a growing community and founding this project will sure be followed with great interest.

7 Acknowledge

Thanks to Professor Slobodan Petrović and Professor Katrin Franke (both from Gjøvik University College) for their patience and supervising. Thanks to Ernst Kristian Henningsen for supporting feedback.

References

[1] Eric Anderson et al. "DataSeries: an efficient, flexible data format for structured serial data". In: *SIGOPS Oper. Syst. Rev.* 43.1 (Jan. 2009), pp. 70–75. ISSN: 0163-5980. DOI: 10.1145/1496909.1496923. URL: http://doi.acm.org/10.1145/1496909.1496923.

[2] Rebecca Gurley Bace. *Intrusion detection*. Indianapolis, IN, USA: Macmillan Publishing Co., Inc., 2000. ISBN: 1-57870-185-6.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN: 0387310738.

[4] Eoghan Casey. "Error, Uncertainty and Loss in Digital Evidence". In: *IJDE* 1.2 (2002).

[5] M. Cotton et al. *Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry*. Internet Draft (draft-ietf-tsvwg-iana-ports-10.txt). Feb. 2011. URL: http://tools.ietf.org/id/draft-ietf-tsvwg-iana-ports-10.txt.

[6] Holger Dreger et al. "Dynamic application-layer protocol analysis for network intrusion detection". In: *Proceedings of the 15th conference on USENIX Security Symposium - Volume 15*. USENIX-SS'06. Vancouver, B.C., Canada: USENIX Association, 2006. URL: http://dl.acm.org/citation.cfm?id=1267336.1267354.

[7] Katrin Franke. "Machine Learning and Pattern Recognition. Lecture Notes". Gjøvik University College. Mar. 2011.

[8] ISO. *ISO/IEC 7498-3:1997 - Information technology – Open Systems Interconnection – Basic Reference Model: Naming and addressing*. 2012. ISO - International Organization for Standardization, May 2006. URL: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=25022.

[9] Jaeyeon Jung et al. "Fast Portscan Detection Using Sequential Hypothesis Testing". In: *IEEE Symposium on Security and Privacy 2004*. Oakland, CA, May 2004.

[10] Kononenko, Igor and Kukar,Matjaz. *Machine learning and data mining : introduction to principles and algorithms*. Chichester: Horwood Publishing., 2007, p. 454. URL: http://lkm.fri.uni-lj.si/xaigor/.

[11] Stefan Kornexl et al. "Building a time machine for efficient recording and retrieval of high-volume network traffic". In: *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*. IMC '05. Berkeley, CA, USA: USENIX Association, 2005, pp. 23–23. URL: http://dl.acm.org/citation.cfm?id=1251086.1251109.

[12] Roger Larsen. *Protocol Detection Capabilities in Bro - The Practical Approach*. Tech. rep. Course IMT-4641 Computational Forensics. Gjøvik University College, June 2012.

[13] Michael Mai. "Dynamic Protocol Analysis for Network Intrusion Detection Systems". Diplomarbeit. Munich, Germany: Technische Universität München, Sept. 2005. URL: http://www.net.t-labs.tu-berlin.de/papers/M-DPANIDS-05.pdf.

[14] Gregor Maier et al. "Enriching network security analysis with time travel". In: *SIGCOMM Comput. Commun. Rev.* 38.4 (Aug. 2008), pp. 183–194. ISSN: 0146-4833. DOI: 10.1145/1402946.1402980. URL: http://doi.acm.org/10.1145/1402946.1402980.

[15] Donald R. Morrison. "PATRICIA - Practical Algorithm To Retrieve Information Coded in Alphanumeric". In: *J. ACM* 15.4 (Oct. 1968), pp. 514–534. ISSN: 0004-5411. DOI: 10.1145/321479.321481. URL: http://doi.acm.org/10.1145/321479.321481.

[16] Richard Mott. "Maximum-likelihood estimation of the statistical distribution of Smith-Waterman local sequence similarity scores". In: *Bulletin of Mathematical Biology* 54 (1 1992). 10.1007/BF02458620, pp. 59–75. ISSN: 0092-8240. URL: http://dx.doi.org/10.1007/BF02458620.

[17] Adam Oliner, Archana Ganapathi, and Wei Xu. "Advances and challenges in log analysis". In: *Commun. ACM* 55.2 (Feb. 2012), pp. 55–61. ISSN: 0001-0782. DOI: 10.1145/2076450.2076466. URL: http://doi.acm.org/10.1145/2076450.2076466.

[18] Ruoming Pang et al. "binpac: a yacc for writing application protocol parsers". In: *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. IMC '06. Rio de Janeriro, Brazil: ACM, 2006, pp. 289–300. ISBN: 1-59593-561-4. DOI: 10.1145/1177080.1177119. URL: http://doi.acm.org/10.1145/1177080.1177119.

[19] Vern Paxson. "Bro: a system for detecting network intruders in real-time." In: *Computer Networks* (1999), pp. 2435–2463.

[20] Slobodan Petrovic. "Intrusion Detection and Prevention. Lecture Notes". Gjøvik University College. Oct. 2011. URL: http://www.hig.no.

[21] The Bro Project. *Downloads*. The Bro Team. May 2012. URL: http://www.bro-ids.org/download/index.html.

[22] P. Renals and G.A. Jacoby. "Blocking Skype through Deep Packet Inspection". In: *System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on*. Jan. 2009, pp. 1 –5. DOI: 10.1109/HICSS.2009.90.

[23] F. Risso and L. Degioanni. "An architecture for high performance network analysis". In: *Computers and Communications, 2001. Proceedings. Sixth IEEE Symposium on*. 2001, pp. 686 –693. DOI: 10.1109/ISCC.2001.935450.

[24] F. Sabahi and A. Movaghar. "Intrusion Detection: A Survey". In: *Systems and Networks Communications, 2008. ICSNC '08. 3rd International Conference on*. Oct. 2008, pp. 23 –26. DOI: 10.1109/ICSNC.2008.44.

[25] M. K. Smith. *Learning theory, the encyclopedia of informal education*. Accessed 2011-01-24. Sept. 1999. URL: http://www.infed.org/biblio/b-learn.htm.

[26] Robin Sommer and Vern Paxson. "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection". In: *Proceedings of the 2010 IEEE Symposium on Security and Privacy*. SP '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 305–316. ISBN: 978-0-7695-4035-1. DOI: 10.1109/SP.2010.25. URL: http://dx.doi.org/10.1109/SP.2010.25.

[27] Sourcefire, Inc. *Snort HomePage - What is Snort?* Accessed 2011-09-22. Sourcefire, Inc. Sept. 2011. URL: http://www.snort.org/.

[28] Tcpdump/Libcap. *TCPDUMP/LIBCAP public repository*. Accessed 2012-05-01. Tcpdump/Libcap. May 2012.

[29] The Bro Project. *Bro Mailing List*. Accessed 2012-06-12. The Bro Project. June 2012. URL: http://mailman.icsi.berkeley.edu/mailman/listinfo/bro.

[30] The Bro Project. *Bro Research*. Accessed 2012-05-01. The Bro Project. Sept. 2011.

[31] The Bro Project. *Bro Workshop 2011*. Accessed 2012-04-29. International Computer Science Institute, University of California, Berkeley / National Center for Supercomputing Applications, University of Illinois. Nov. 2011. URL: http://www.bro-ids.org/bro-workshop-2011.

[32] The Bro Project. *Dynamic Protocol Detection*. Accessed 2012-05-21. International Computer Science Institute, University of California, Berkeley / National Center for Supercomputing Applications, University of Illinois. May 2012. URL: http://www.bro-ids.org/development/dpd.html.

[33] The Bro Project. *The Bro Network Security Monitor*. Accessed 2011-09-14. URL: http://bro-ids.org/.

[34] The Bro Project et al. *The Time Machine*. Accessed 2012-05-29. Technische Universität Berlin, the Technische Universität München, and the ICSI (University of California Berkeley). May 2012.

[35] The Europe Parliament. *Directive 2006/24/EC of The European Parliament and of The Council*. English. Electronic. Journal, Online. Accessed: 2012-05-25. Mar. 2006. URL: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2006:105:0054:0063:EN:PDF.

[36] Theodoridis S. and Koutroumbas K. *Pattern Recognition, Fouth Edition*. ACADEMIC PRESS, 2009. URL: http://cgi.di.uoa.gr/~stheodor/.

[37] 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*. May 2007, pp. 1 –8. DOI: 10.1109/E2EMON.2007.375317.

Appendix A — The TCP/IP and OSI model compared.

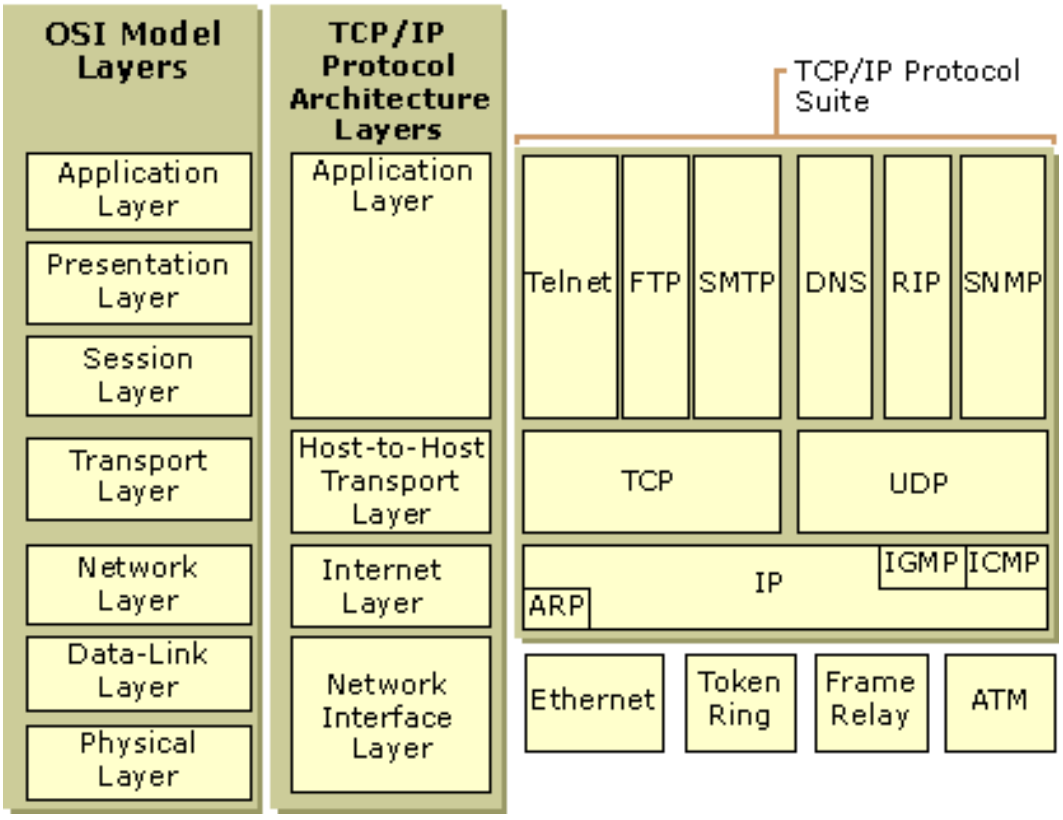

Figure 3: The OSI and TCP/IP model layers with some popular protocols illustrated.

Appendix B — The Time Machine Architecture.

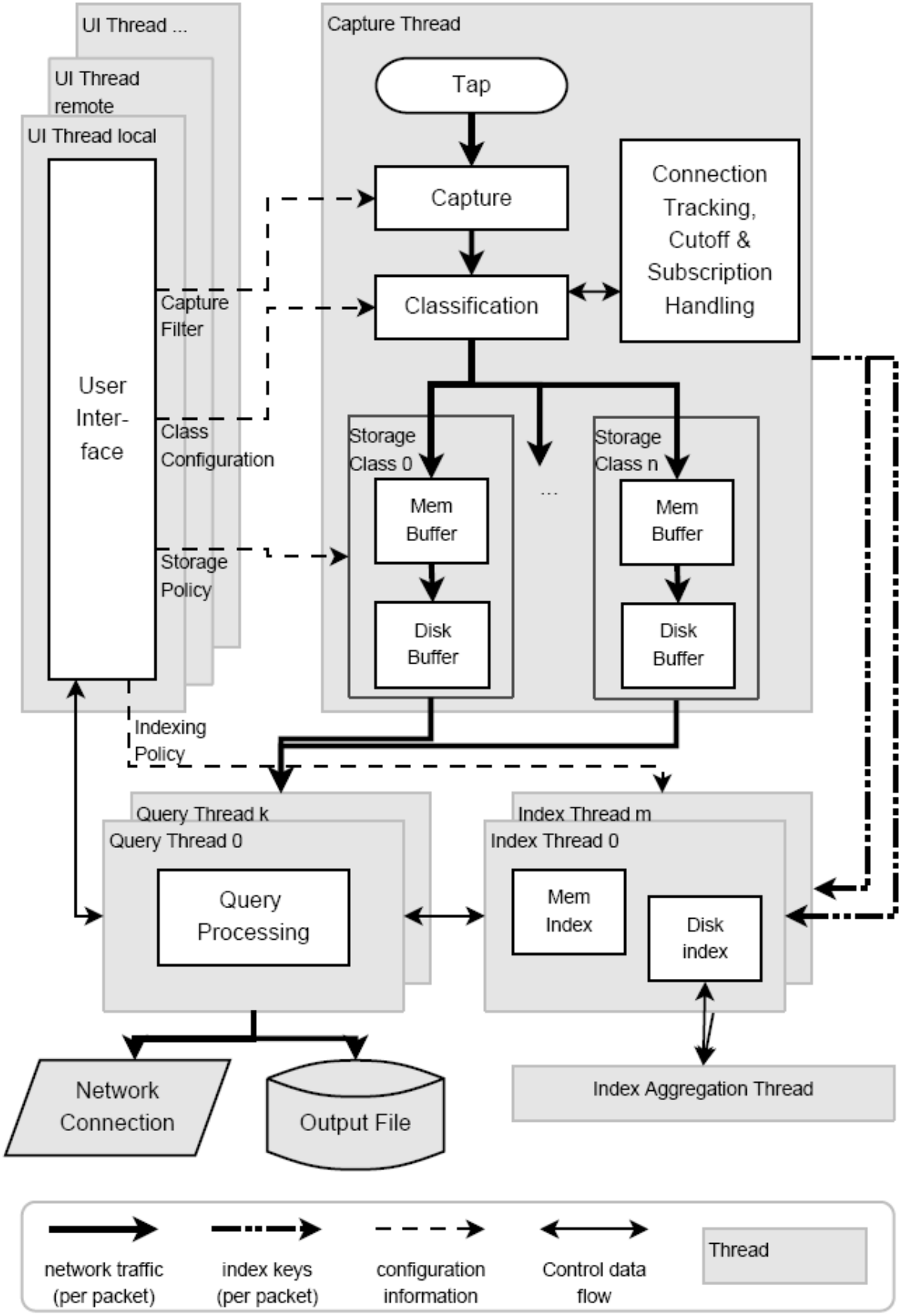

Figure 4: The Time Machine architecture.

Appendix C — Email from The Bro Team

-----Original Message-----
From: Robin Sommer [mailto:robin@icir.org]
Sent: 13. juni 2012 16:45
To: Roger Larsen - Hï¿½gskolen i Gjï¿½vik
Cc: bro@bro-ids.org
Subject: Re: [Bro] Are there any machine learning functionality in Bro?

On Wed, Jun 13, 2012 at 09:56 +0200, Roger Larsen - H gskolen i Gj vik wrote:

> Threshold Random Walk algorithm (TRW).

(TRW doesn't use machine learning.)

> Are the learning capabilities in Bro well hidden   or missing?

Missing. Not by design, but out of operational concerns: it's extremely hard
to get ML approaches to work reliably. You may be interested in this paper
we wrote a little while ago on this topic:

    http://www.icir.org/robin/papers/oakland10-ml.pdf

Robin

--
Robin Sommer * Phone +1 (510) 722-6541 * robin@icir.org
ICSI/LBNL   * Fax   +1 (510) 666-2956 *   www.icir.org