LAN ABNORMALITIES THREAT DETECTION: AN OUTLOOK AND APPLICABILITY ANALYSIS

A.M.Modorskiy[1], A.S.Minzov[2], O.R.Baronov[3], A.Y.Nevskiy[4].

In this article contemporary LAN threats are considered in scope of signatures, where a pile of polymorphic elements, not applicable for diverse analysis by itself are combining along with preambles, traffic strains and flows, as well as protocols and ports, could be reviewed to tell if there are threats detected. The point is to compile a standalone system, capable of scoping and triaging diversified elements on a LAN Core, giving a system owner an opportunity to early detect, prioritize and workaround threat that standard security systems allow by default. However, it is wrong to consider such mechanism a classic signature–based, where traffic dump is investigated for known issues. Conversely, the system in scope should act proactive, which require shaping of basic, likewise ideal, traffic flow, seeking for abnormalities in an early threat occurrence. For this occasion, the neural network should step in, utilizing the vector comparison for the abnormalities detection process being effective.

***Keywords**: Security, Networks, LAN, Threat Detection, Cisco, Neural Networks, Signature, Firewalling, Traffic dump.*

DOI: 10.21681/2311-3456-2018-1-11-18

Introduction

A contemporary world of Data Security could seemingly have a whole set of defensive mechanisms: depends on a current need we have firewalls, IDS/IPS, antiviruses, integrated security solutions etc. However, could we suppose this set complete? As Cisco CEO John Chambers recently said: «There are two types of companies: those that have been hacked and those who don't know they have been hacked». Couldn't agree with him more; despite constant improvement in data security field, despite a bunch on new technologies, all the researches and product range we still suffer severe attacks and, as a consequence, casualties.

Why is this topic a thing? Well, the answer is simple, yet catchy. IT world has no current systems capable of acting fully proactive. The closest to this we have is a signature-based defense systems. The point is once each signature met, the prevention mechanism is triggered, so a full attack is mitigated (or neglected) at the very beginning acting so-called «proactive». In fact, this is not 100% true. The signature is yet to be found, established, processed and spread until it became functional. Suppose we have no defense against not-well-known threats, making a zero-day and system vulnerabilities a thing we may only overcome once recorded and studied. In fact, if a hacker (intruder, attacker) is first to find a weak point, we may only pray our system is sufficiently protected on access level otherwise attack is predefined successful.

What can we do with this state? Obviously, we need a system to have a fully proactive mechanism, meaning, if we are focusing on a network POV and LAN specifically, some sort of traffic flow control, which allowing us to detect any abnormalities, thus decreasing both 1st and 2nd type errors.

Chapter 1. Abnormalities thread detection method overview

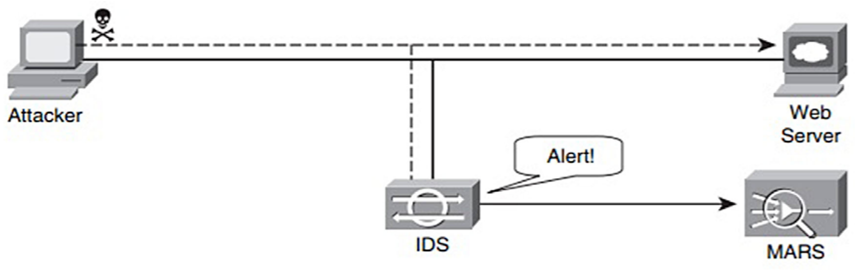
Leading to contemporary world techniques and defense systems, signature threat detection systems are not a newbie to a data security. The most common solution closest to considerate topic is IDS/IPS class systems. IDSs are devices that in promiscuous mode detect malicious activity within the network. IPS devices are capable of detecting all these security threats; however, they are also able to drop noncompliant packets inline. Traditionally, IDS systems have provided excellent application layer attack-detection capabilities; however, they were not able to protect against day-zero attacks using valid packets. The problem is that most attacks today use valid packets. On the other hand, now IPS systems such as the Cisco IPS software Version 6.x and later offer anomaly-based capabilities that help you detect such attacks. This is a big advantage, since it makes the IPS devices less dependent on signature updates for protection against DDoS, worms, and

1 Alexey Modorskiy, Master degree student Information and Economic Security Institute,, CCNA, CCNA Instructor, National research University «MPEI», Moscow, Russia. E-mail: alexeymodorskiy@gmail.com

2 Anatoliy Minzov, Dr.Sc. Professor, National research University «MPEI», Moscow, Russia. E-mail: aminzov@mpei.ru

3 Oleg Baronov, Associate Professor, Ph.D., National research University «MPEI», Moscow, Russia. E-mail: baronovor@mpei.ru

4 Alexander Nevskiy, Associate Professor, Ph.D., National research University «MPEI», Moscow, Russia. E-mail: nevskyay@mpei.ru



Variances of IDS/IPS systems defined under MARS Monitoring System control

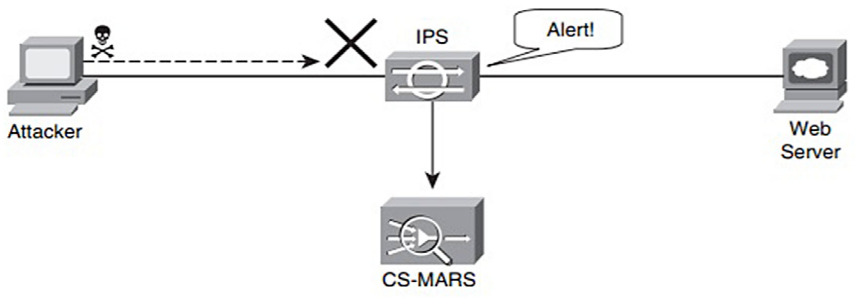


Figure 1-2. *IDS/IPS solution based on MARS example*

any day-zero threats. Just like any other anomaly detection systems, the sensors need to learn what is «normal.» In other words, they need to create a baseline of legitimate behavior [1].

Looks a rather close to a topic being reviewed, isn't it? The basics are solid milestone: everything we need to protect our network undercover a convenient and reliable vendor. Let's dig a little deeper into a underlying mechanism. Turns out, as the most of IDS/IPS, Cisco utilizes a monitoring mechanism – Netflow – as shown on Figure 1. Other vendors often rely on vendor-independent technologies, yet the outcome is still: monitoring sys tem is a key to work on anomalies.

The figures above only gives us a stratified and simplified look to the technology, which is, of course, more complicated and advance.

Allow us to have a look at Neflow:

Each packet that is forwarded within a router or switch is examined for a set of IP packet attributes. These attributes are the IP packet identity or fingerprint of the packet and determine if the packet is unique or similar to other packets, as presented on Figure 3.

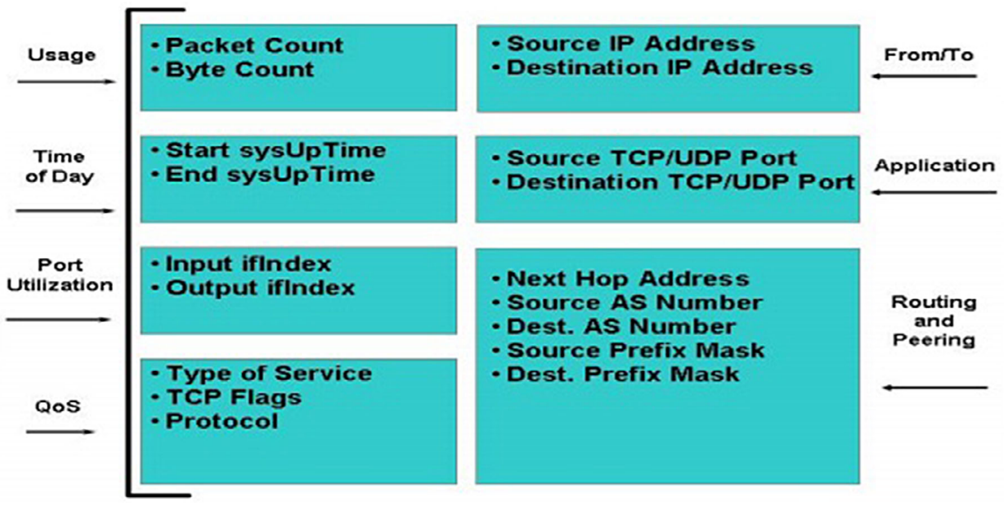
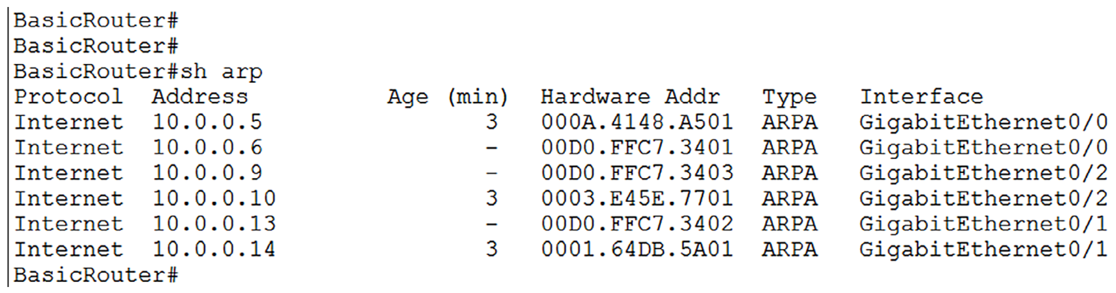


Figure 3. *Cisco NetFlow v5 network utilization data report example [3]*



```
BasicRouter#
BasicRouter#
BasicRouter#sh arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  10.0.0.5                3   000A.4148.A501  ARPA   GigabitEthernet0/0
Internet  10.0.0.6                -   00D0.FFC7.3401  ARPA   GigabitEthernet0/0
Internet  10.0.0.9                -   00D0.FFC7.3403  ARPA   GigabitEthernet0/2
Internet  10.0.0.10               3   0003.E45E.7701  ARPA   GigabitEthernet0/2
Internet  10.0.0.13               -   00D0.FFC7.3402  ARPA   GigabitEthernet0/1
Internet  10.0.0.14               3   0001.64DB.5A01  ARPA   GigabitEthernet0/1
BasicRouter#
```

Figure 4. *ARP table overview, as issued on Cisco Router Series 2900*

Traditionally, an IP Flow is based on a set of 5 and up to 7 IP packet attributes.

IP Packet attributes used by NetFlow:

- IP source address;
- IP destination address;
- Source port;
- Destination port;
- Layer 3 protocol type;
- Class of Service;
- Router or switch interface [2].

Away with the Cisco technologies, the rest of network-based IDS/IPS systems are quite alike, for example, Juniper SRX utilizes PCAP Syslog along with Juniper Secure Analytics (JSA) appliance, which basically is the same filtering solution [4]. Same may be found under Checkpoint, PaloAlto and other market leading vendor solutions.

Is this really enough? Well, to find out we need again to dive into the underlying principle, which is an OSI packet flow relation and processing ability. Turns out that mentioned traffic flow control protocols are strictly limited to routing and transport layers of OSI, since classic routers, as well as firewalls or gateways are operating at four below OSI layers (it's often misunderstood those devices are only operating at layers 3-4, however it's obviously has a physical layer, and those devices does have a Layer 2 operational units since ARP tables are exists and default routing contains MAC-address changes from one routing device to another – otherwise neighbours could not know each other), as shown on Figure 4 and Table 1.

As we can see, such IDS/IPS solutions are limited by design, since they utilize embedded, built-in routing mechanism. The advantages of these solutions are considerable: significant decrease of additional load on active net device, standard architecture and configuration, optimization, topology-independency etc.

However there is certainly the flip side of a coin. For the default router, as well as classic firewall, each OSI layer above 4th is just a payload with no considerate

Table 1. *OSI standard representation as presented in LAN*

Layer		Protocol data unit (PDU)	Functions
Host layers	7. Application	Data payload	Application internal information, often described as payload.
	6. Presentation		Translation of data from application to network format and vise versa
	5. Session		Session control between two endpoints including time sync
	4. Transport	**Segment (for TCP) or Datagram (for UDP)**	**Reliability control between endpoints, segmentation and fragmentation control.**
Media layers	**3. Network**	**Packets**	**Routing between either endpoints or subnets using routing protocols**
	2. Datalink	**Frames**	**Switching between two or more endpoints (connected to the same switching infrastructure) and reliability control**
	1. Physical	**Bits**	**Transmission of electrical signals**

indexes/preambles etc. Hence, it couldn't be investigated for further decision-making process and is not considerate useful for mentioned IDS/IPS solutions.

If we need a solution to scope a whole packet into investigation and filtering, we obviously have to have a Layer 7 device, either a same called firewall or server, capable of running appropriate software and equipped with a set of required hardware/firmware. This is not a new solution to a market, those devices are called host-based IDS/IPS and they utilize a variety of advantages:

- Full 7 layer OSI coverage;
- Any IP-demanded filtering;
- Flexibility of use.

As far as we have advantages, disadvantages are also in place:

- Dedicated environment demand;
- Low speed of filtering;
- Network traffic is not counted for host-based solutions;
- Costs etc.

Those solutions are extremely protective yet expensive, hard to tune and support. Host-based IDS often require a small-cell diversion of LAN, since are only capable of carrying application traffic parameters [5]. Layer 7 Firewalls (as well as Multilayer Firewalls) are way more advanced, yet extremely costly, and often work as a transparent devices (it is recommended to implement transparent firewall mode on a network, if firewall is implemented along router [6]), which means they are routing-insensitive.

As a result, there is currently no end-to-end solution to cover a whole scope of LAN network security. Since valid packet threats are a thing we cannot only rely on integrated network IDS/IPS [7], while filtering is a target of networking devices. A complete solution is a more like a compilation of Host-based IDS, Network-based and a Multilayer (7th layer) Firewall. This is a compilation of disadvantages as well: a poor performance, costs and a rather challenging support.

Chapter 2. How to perform

Clearly, the solution to original agenda should aggregate a whole scope of technologies to perform a complete investigation of traffic. Yet the most efficient way is to have an ideal dump, therefore having the non-standard traffic analyzed separately. There are multiple advantages to this solution: we do not need to have a filtering device working 24/7, inspecting a whole flow of payload, but enabling specifically at the time anomaly detected, improving performance and boosting the routing; the analysis itself becomes more efficient due to a considerate decrease on a data marked to investigate. As a consequence, a LAN may miss a whole set of infrastructure dedicated to act IDS-alike, while bearing specific device, say, a server, performing on-demand with a few recourses allocated at the time.

This raises a couple of reasonable questions:

1. How do we have a normal traffic dump idea?
2. How to analyze an ideal dump for abnormalities?
3. Where to have analyzing equipment installed?

The first question is basically a matter of modeling. Since only a LAN traffic is in scope, we are able to decrease the area to the data being send and received between local resources. Nevertheless, the problem of modeling this traffic flow is a thing. To solve this task it is better to have original traffic decomposed to several components easier to analyze [8-9].

Firstly, a service traffic – data, required by network devices to communicate between each other and function around dedicated mechanisms (e.g. routing, fail-proof, redundancy etc.) [10]. A formalizing of this traffic could be done by simply listing a used technologies or sniffing traffic in a «silent mode», where no payload is neither sent nor received. It is worth saying such test should be done on an isolated LAN where no suspicious traffic is presented, and there is only one way to have it done: on the network cut-over, when LAN is initially disconnected from unprotected environments such as Internet or adjacent LANs.

Having a listed scope of service data circling on a network we may proceed to determining user traffic and at this point we may need to divide payload flow from servicing traffic to have a clear representation of ideal (or normal) dump. How can we have this done? Modeling is the best way to perform in this case.

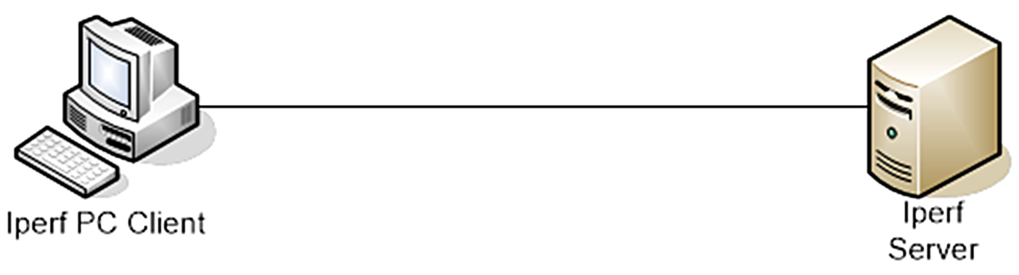


Figure 5*. Iperf relations schema*

Table 2. Iperf general options

GENERAL OPTIONS	
Command line option	Description
-p, --port n	The server port for the server to listen on and the client to connect to. This should be the same in both client and server. Default is 5201.
--cport n	Option to specify the client-side port. (new in iPerf 3.1)
-f, --format [kmKM]	A letter specifying the format to print bandwidth numbers in. Supported formats are ‹k› = Kbits/sec ‹K› = KBytes/sec ‹m› = Mbits/sec ‹M› = MBytes/sec The adaptive formats choose between kilo- and mega- as appropriate.
-i, --interval n	Sets the interval time in seconds between periodic bandwidth, jitter, and loss reports. If non-zero, a report is made every interval seconds of the bandwidth since the last report. If zero, no periodic reports are printed. Default is zero.
-F, --file name	client-side: read from the file and write to the network, instead of using random data; server-side: read from the network and write to the file, instead of throwing the data away.
-A, --affinity n/n,m-F	Set the CPU affinity, if possible (Linux and FreeBSD only). On both the client and server you can set the local affinity by using the n form of this argument (where n is a CPU number). In addition, on the client side you can override the server's affinity for just that one test, using the n,m form of argument. Note that when using this feature, a process will only be bound to a single CPU (as opposed to a set containing potentialy multiple CPUs).
-B, --bind host	Bind to host, one of this machine's addresses. For the client this sets the outbound interface. For a server this sets the incoming interface. This is only useful on multihomed hosts, which have multiple network interfaces.
-V, --verbose	give more detailed output
-J, --json	output in JSON format
--logfile file	send output to a log file. (new in iPerf 3.1)
--d, --debug	emit debugging output. Primarily (perhaps exclusively) of use to developers.
-v, --version	Show version information and quit.
-h, --help	Show a help synopsis and quit.

First of all, we need to exclude service traffic, and there's only one way to do it with 100% efficiency: getting rid of network device, stratifying a host-server relations. There are several techniques to do it, let's consider the simplest: a traffic generator [11].

In this example we will use iperf traffic generator as a simple, free-based software available online.

For the correct usage we will need to consider a following simple topology presented below on Figure 5:

The point is to have a both way relations required to establish a model which is maximum close to a real one, excluding any service-related flow. Syntax is clear and easy to use (refer to Table 2).

Having this utility settled and tune we may proceed to collecting a dump of ideal, or normal, traffic. For this matter we may use either iperf embedded output or Wireshark as a sniffer. To use this application we would need to adjust and convert original topology (Figure 6):

This application allows engineer to sniff packets flowing through a networking interfaces while not interrupting the flow itself [12]. Clearly, to built alike topology we will need to have a machine with at least 2 NW cards installed. Original user interface of Wireshark is rather clear and straight-through, which makes sniffing easy. Outcome of sniffing process of-

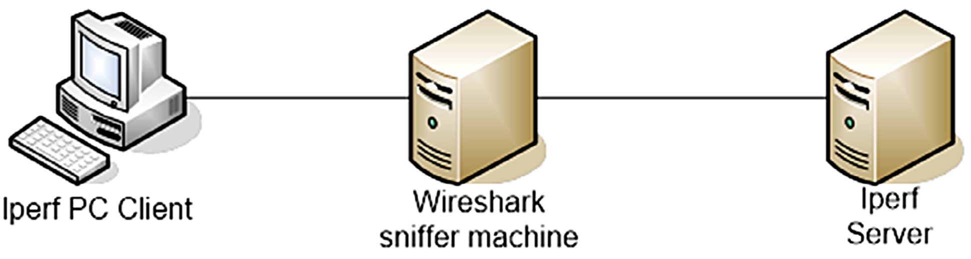
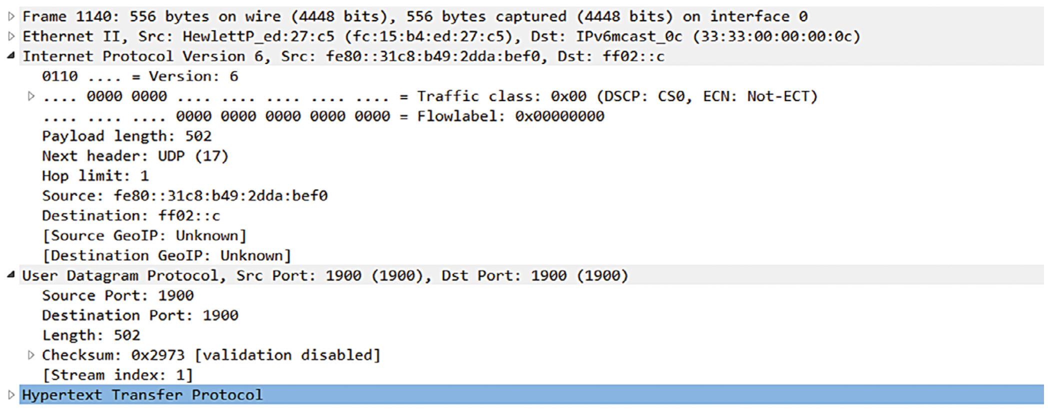


Figure 6. *Iperf realtions schema with sniffer integrated*



```
▷ Frame 1140: 556 bytes on wire (4448 bits), 556 bytes captured (4448 bits) on interface 0
▷ Ethernet II, Src: HewlettP_ed:27:c5 (fc:15:b4:ed:27:c5), Dst: IPv6mcast_0c (33:33:00:00:00:0c)
◢ Internet Protocol Version 6, Src: fe80::31c8:b49:2dda:bef0, Dst: ff02::c
    0110 .... = Version: 6
  ▷ .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 502
    Next header: UDP (17)
    Hop limit: 1
    Source: fe80::31c8:b49:2dda:bef0
    Destination: ff02::c
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
◢ User Datagram Protocol, Src Port: 1900 (1900), Dst Port: 1900 (1900)
    Source Port: 1900
    Destination Port: 1900
    Length: 502
  ▷ Checksum: 0x2973 [validation disabled]
    [Stream index: 1]
▷ Hypertext Transfer Protocol
```

***Figure 7**. Example of Wireshark sniffer output*

ten presented as a combinations of packets tied by some of crucial preambles and service markers, as shown on Figure 7 above.

A compilation of iperf and Wireshark is a reasonable way to have a payload traffic flow modeled, since an output is a scalable model considering input data and case-sensitive, yet not the only.

For the SOHO and middle Enterprise LAN we may also find heuristic analysis as a suitable form of modeling [13]. For example, a secure environments requiring DMZ are built based on a compilation of ACLs following a traffic audit or predictions [14].

Let's now switch to the second stated question – how to analyze an ideal dump for abnormalities. This is the main point of research since there are currently no systems capable of determining deviations from a clean dump. The modern systems are working vise versa, looking for known signatures on unknown flow.

This brings us to the point where we need to develop such system, yet we require it to be based on standard solutions for the sake of stability and readiness to go for production. As a first step, we need to compile and represent each packet properties to the form appropriate for further analysis. To simplify the input data, we may create a table of listed packet properties, marking each property as «1» if included or «-1» otherwise (bi-polar standard representation [15]). This brings every packet to following view as presented on Table 3.

***Table 3**. Binary algebra issued to a packet*

properties	index
VLAN flag	1
QoS tag	-1
TCP identifier	1
UDP identifier	-1
…	…

The main issue following this schema is a non-binary properties such as IP addresses and alike parameters. It is supposed we still may formalize such cases using a number of variables from the left pane.

How to fulfill such table? Well, the easiest way is to combine a database table, having those variables sent via simple procedure: a Wireshark converts variables to .CSV format, while a script (supposing we are running WIN workstation or server) inputs it to database:

```
CREATE TABLE packet_prop (
  VLAN DECIMAL(10,2) NULL,
  QoS DECIMAL(10,2) NULL,
  TCP DECIMAL(10,2) NULL,
  UDP DECIMAL(10,2) NULL,
  PRIMARY KEY (id)
);
LOAD DATA INFILE ‹c:/tmp/current_packet.csv›
INTO TABLE discounts
FIELDS TERMINATED BY ‹,›
ENCLOSED BY ‹«›
LINES TERMINATED BY ‹\n›
IGNORE 1 ROWS;
```

Above is represented only a simplest case, while an adequate datastore should include a variety of parameters depends on a network in scope.

Having a main table fulfilled with first packet we may switch over to analysis. Obviously, as valid traffic attacks are already mentioned, we would need more than one sample to compile a full ideal traffic dump. Its size may vary depending on an environment and audit performed. At this moment, we need to decide what component may we use to analyze the dump while comparing it to suspicious traffic.

Let's get input parameters together: having an ideal traffic dump in place we would need to compare any traffic marked as «unusual» to the gold probe, hence desired analyzing tool should have an embedded mechanism to compare every each of current traffic dump to a packet (or a group of packets) from the original dump. There we can use a neural network adhesive to original issue. Basically we need quite the same mechanism applied on ATM to check whether a cash bill is recognized correctly and available to use within certain environment. This is a Hopfield neural network, or, to be more specific, a Little's neural network as a derivative to original one.

The point of using such method is having a complete set of input parameters as it is: a database table described earlier could compile a summary of ideal probes required for Hopfield network to properly function, as present ed at (1-3):

$$\overrightarrow{x_1} = [x_1, x_2, \dots, x_n]; \quad (1)$$

$$\overrightarrow{x_2} = [y_1, y_2, \dots, y_n]; \quad (2)$$

$$\overrightarrow{x_3} = [z_1, z_2, \dots, z_n]; \quad (3)$$

$$\vec{s} = [s_1, s_2, \dots, s_n]; \quad (4)$$

Where $\overrightarrow{x_1}, \overrightarrow{x_2}, \overrightarrow{x_3}$ – packets being originally captured at the modeling stage, and $\vec{s}$ represents a packet (a group of packets) under consideration.

Now we may compile a matrix using standard Little's network principle:

$$W = \sum_{k=1}^{3} (\vec{x}_k^t \, \vec{s}) \; ; \quad (5)$$

Where W is the matrix and 'k' is a counter for current vector being used. For the vector it is easier to use it to avoid original idea's violation, since, in fact, we need to compare each parameter with its own baseline.

A comparison happens as each vector marked suspicious is multiplied repeatedly with a whole set of baseline vectors, which outputs as either inverted packet original packet (from ideal scope, depends on which packet's variation being considered by neural network) or a complete unknown vector, which may point out an anomaly.

The suggested solution is a new direction to Data Security, yet it has some lookalikes when compared to CM systems. However, it does differ starting from underlying concept: unlike CM, it works with a specific modeled ideal dump rather than assuming some point where traffic is clean while comparing it to any given moment. This allows system administrator to collate a suspicious object to an ideal data stamp, not specific state of LAN traffic flow defined formerly. It is also concentrates on a network components rather than standard server-client infrastructure.

Conclusion. Implementation and future development.

Having established an environment and a set of instrumentals we, however, does not getting a complete product. To make it usable and user-friendly we still strive to have a GUI, sufficient modeling constructor, plug-and-play database and so on.

As a result to this research we offered a new solution, which, if correctly compiled and implemented, may considerably increase overall LAN security while sharing existing infrastructure and cutting the costs. A solution offered combines both Network- and Host-based IDS/IPS principles, while having a case-sensitive reaction system at the same time. The key advantages to issued solution may vary depending on a LAN infra, scoping data processing speed, self-learning mechanism, simplicity, and, at the end, this is a brand new solution completely unknown to modern hacking community.

However there are still some blind spots to this topic: the system is incomplete and for this moment is database-dependent. This means a neural network should be used based on a DB table, using a built-in multiplication, transpose and summation methods. Those factors may impact performance and self-learning abilities, decreasing the LAN throughput if attached to Core layer.

It is predictably more efficient to use programming languages for neural network platform while having a joint with DB, so future development already has some areas to evolve. Nevertheless, main components should seat still: a neural network, sniffer, modeling tool and database have to be installed for proper functioning. The rest may vary depending on a certain cases and preferences.

***Reviewer**: V.L. Tsirlov, Ph.D., Associate Professor, Information Security Department, Bauman Moscow State Technical University, Moscow, Russia. E-mail: v.tsirlov@bmstu.ru*

References

1. End-to-End Network Security: Defense-in-Depth By Omar Santos. Published Aug 24, 2007 by Cisco Press, pp. 19-22
2. Introduction to Cisco IOS NetFlow - A Technical Overview, Cisco White papers, issued May 29, 2012. [https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html]
3. NetFlow gives Network Managers a Detailed View of Application Flows on the Network, Cisco® IT Case Study/Cisco Network Management/NetFlow, Case study, issued © 2004 Cisco Systems, Inc,[http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6555/ps6601/prod_case_study0900aecd80311fc2.pdf], p. 4.

4. Junos® OS IDP Series Appliance to SRX Series Services Gateway Migration Guide. Copyright © 2017 Juniper Networks, Inc, pp. 3-7.
5. Host-based vs Network-base Intrusion detection systems, SANS institute article, [https://cyber-defense.sans.org/resources/papers/gsec/host-vs-network-based-intrusion-detection-systems-102574], 2005, pp.3-5
6. CLI Book 1: Cisco ASA Series General Operations CLI Configuration Guide, 9.4, Cisco inc. Issued Dec 4, 2017.
7. K. Scarfone, P. Mell, Special Publication 800-94: Guide to Intrusion Detection and Prevention Systems (IDPS), National Institute of Standards and Technology (NIST) (2007), pp 4-6.
8. Two-layer modeling for local area networks Authors: M. Murata, Comput. Center, Osaka Univ.,Japan, H. Takagi, pp 1-10
9. Modeling and Analysis of Wireless LAN Traffic, JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 25, 1783-1801 (20090, DASHDORJ YAMKHIN AND YOUJIP WON+, Hanyang University, Seoul, 133-791 Korea
10. Enhancing LAN Performance, Gilbert Held, Fourth Edition, CRC Press, 18 March, 2004 , pp 130-140 ISBN 978-0-203-49605-3
11. A Network Traffic Generator Model for Fast Network-on-Chip Simulation, IEEE Article, Shankar Mahadevan, Federico Angiolini, Michael Storgaard, Rasmus Grøndahl, Olsen Jens Sparsø, Jan Madsen, Informatics and Mathematical Modelling (IMM), Technical University of Denmark (DTU), Richard Petersens Plads, 2800 Lyngby, Denmark - Dipartimento di Elettronica, Informatica e Sistemistica (DEIS), University of Bologna, Viale Risorgimento, 2 40136 Bologna, Italy. Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'05), 1530-1591/05 IEEE, pp 1-6.
12. Wireshark - The best open source network packet analyzer(Part I), Himanshuz.chd | Sep 23 2012, IBM DeveloperWorks Electronic Archive.
13. Traffic inspection for visibility, control and new business opportunities, Ericsson White papers, 284 23-3112 Uen Rev B | September 2010, pp 2-12.
14. The Science DMZ: A Network Design Pattern for Data-Intensive Science, Eli Dart, Lauren Rotman, Brian Tierney, Mary Hester Jason Zurawski, Lawrence Berkeley National Laboratory under Contract No. DE-AC02-05CH11231 with the U.S. Department of Energy. The U.S. SC13 November 17-21, 2013, Denver, CO, USA Copyright 2013 ACM 978-1-4503-2378-9/13/11, page 5
15. Mathematical morphology on bipolar fuzzy sets: general algebraic framework, IsabelleBloch Télécom ParisTech, CNRS LTCI, Paris, France, May, 2012, pp 2-3