

# Description of Bad-Signatures for Network Intrusion Detection

Michael Hilker

Christoph Schommer

University of Luxembourg, Campus Kirchberg  
Dept. of Computer Science and Communication  
6, Rue Richard Coudenhove-Kalergi, L-1359 Luxembourg  
Email: {michael.hilker, christoph.schommer}@uni.lu  
Phone: +352-420101-{234, 228}

## Abstract

Today, a computer network is under constant assault from attacks. In Computer Science, NIDS are used in order to protect a computer network against these intrusions. These systems normally use stochastic approaches or a rule-based system to detect intrusions and to describe the known intrusions. These systems have some disadvantages which we solve with a new approach called ANIMA. ANIMA stores bad-signatures of intrusions in directed and weighted graphs as well as returns for each checked-packet a value how malicious the packet is. The primary advantages of ANIMA are the online-system, adaptation, easy administration and storage-saving. In this article, we discuss the approach ANIMA for intrusion detection, the advantages and disadvantages, the implementation as well as the results occurred out of the simulations that ANIMA for intrusion detection works well in bad-packet-identification as well as the implementation substantiates the theoretical advantages.

## Keywords:

Network Intrusion Detection, Artificial Immune System, Data Streams, Anomaly Detection.

## 1 Introduction

In today's Internet and academic or commercial networks, the computers and servers are under constant assault from hackers, worms and viruses; these attacks are called *intrusions*. In order to secure a network or a network-node (e.g. server or computer), the administrators runs *NIDS* (Network Intrusion Detection Systems) as well as virus- and malware guards. Examples of NIDS are discussed in (Debar, Dacier & Wespi 1998, Roesch 1999, Snapp, Brentano, Dias, Goan, Heberlein, Lin Ho, Levitt, Mukherjee, Smaha, Grance, Teal & Mansur 1991, Staniford-Chen, Cheung, Crawford, Dilger, Frank, Hoagland, Levitt, Wee, Yip & Zerkle 1996, Porras & Neumann 1997, Janakiraman, Waldvogel & Zhang 2003). There also exist NIDS using Artificial Immune Systems, e.g. figured out in (Hofmeyr & Forrest 1999, Hofmeyr & Forrest 2000b, Hofmeyr & Forrest 2000a, Spafford & Zamboni 2000).

In all of these solutions finding intrusions, the system performs a string-matching. The system stores in an appropriate data structure the known bad-patterns of recognised intrusions. These bad-patterns are strings with various length. Afterwards, the system matches all packets against the bad-patterns in order to recognise the bad-packets. Therefore, the system examines the packets as strings and matches these strings against the bad-packets.

If the system identifies a packet as malicious, the packet is removed from the network and the network is secured from this attack.

Another approach for finding attacks is *anomaly detection*. Herein, the system tries to find behaviour of attacks which are called anomaly. One approach using finite automates is introduced in (Sekar, Gupta, Frullo, Shanbhag, Tiwari, Yang & Zhou 2002). A good overview-article is (Lazarevic, Ertoz, Ozgur, Srivastava & Kumar 2003) and a novel approach for anomaly detection is described in the article (Leung & Leckie 2005).

However, in this article we focus on intrusion detection. The main features of a NIDS are:

- Correct identification of known bad-signatures
- Tolerance against good-packets
- Adaptation: identification of novel intrusions, especially intrusions which have a signature similar to known bad-signatures
- Dynamic Behaviour: well working update for the set of bad-signatures
- Fast Behaviour: need few computational power for checking one packet-signature

Nowadays, nearly all systems use a simple rule in order to store one bad-signature and for each packet all rules must be checked. Furthermore, as the rules are static, it is not easy to update the rules if the system learns new bad-signatures and the duration time for the checking of one packet is pretty slow.

In this article<sup>1</sup>, we concern an information structure for bad-signatures based on ANIMA. The data structure is dynamic, storage space saving and adaptive. The main novel feature is that this version of ANIMA can be updated with new bad-signatures continuously.

ANIMA<sup>2</sup> is a system in order to store and to detect associative patterns from data streams (Schommer 2005, Schommer 2004, Schroeder & Schommer 2005). ANIMA processes a data stream and it is possible to receive the association rules and relationships at any time during processing - the online-concept. The idea of ANIMA is to store the transaction in an undirected network and to weight the nodes and connections. Consequently, it is possible to always extract the association rules and there originate skeletons which describe the main association rules with highest support and confidence. ANIMA for finding association rules will be introduced in the Section 3.

Moreover, ANIMA is not only a system for storing and detecting association rules, it is an idea of an approach for a lot of problems in computer science using the nature as archetype.

Copyright ©2006, Australian Computer Society, Inc. This paper appeared at the Fourth Australasian Information Security Workshop (AISW-NetSec 2006), Hobart, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 54. Rajkumar Buyya, Tianchi Ma, Rei Safavi-Naini, Chris Stekette and Willy Susilo, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

<sup>1</sup>This work is supported by the *Ministre Luxembourgeois de l'éducation et de la recherche* through the project INTRA (= *Internet TRAffic management and analysis*), which is currently performed at the *University of Luxembourg*.

<sup>2</sup>ANIMA is located in the project *EVO-BUSINESS - Evolutionary Algorithms in eCommerce* was introduced at the *University of Luxembourg* and founded by the *Ministre Luxembourgeois de l'éducation et de la recherche*.

Our novel approach is to use a modification of ANIMA in order to describe the bad-signatures of intrusions. Therefore, the signatures of intrusions are stored in a directed network. Each node represents one character and some nodes are connected by a directional connection. The system weights the nodes and connections in order to guarantee the proper storage of bad-signatures. The new version of ANIMA provides two different interfaces:

- **Learning:**  
This interface provides the behaviour of bad-signatures learning. Through this interface, a new bad-signature is stored in ANIMA.
- **Checking:**  
For packet-checking is this interface used. A packet-signature is feed in and ANIMA returns if the packet is malicious, malicious with high probability or non-malicious.

Consequently, ANIMA can be always updated, ANIMA is adaptive and tolerant. For a detail description we refer to the Section 4.

**Convention 1** *In order to distinguish between the two applications for ANIMA, we label ANIMA for Association Rules as ANIMA-AR and ANIMA for Intrusion Detection as ANIMA-ID.*

We implemented ANIMA-ID in order to substantiate the advantages figured out in the theoretical part of this article as well as to simulate scenarios in order to test ANIMA-ID. ANIMA-ID fortifies all advantages and operates well in the testing-scenarios.

The article is structured as follows. Section 2 gives an overview about Intrusion Detection, Section 3 describes ANIMA-AR and Section 4 discusses ANIMA-ID. Section 4 is divided into Subsections. Subsection 4.1 defines the storing of bad-signatures, Subsection 4.2 introduces the checking of bad-packets and Subsection 4.3 discusses the advantages as well as disadvantages of ANIMA-ID. Subsection 4.4 gives an example of a scenario for ANIMA-ID, Subsection 4.5 figures out the implementation of ANIMA-ID, Subsection 4.6 compares our approach with SNORT and Subsection 4.7 provides a proof for completeness. Thereafter, the Section 5 describes some examples of applications of ANIMA-ID and Section 6 concludes the article and gives an outline about the future work.

## 2 Current Situation

Nowadays, three different approaches for detecting bad-signatures in packets are mainly used:

1. **Rule-based storage of bad-signatures:**  
The rule-based storage approach saves the bad-signatures in rules like if *signature<sub>a</sub>* is found the packet is malicious. The advantages of this approach are that it is easy to implement and it is easy to check a packet-signature against a rule. The disadvantages are that it is time-consuming to check a packet-signature against all rules because there can be redundant rules as well as it is hard to update the rules if new bad-signatures are found - especially an online update on a running system.
2. **Stochastic approach:**  
In this approach, no bad-signatures are stored. The NIDS monitors the network traffic and analyses the traffic using stochastic methods. Consequently, it is another approach which is not proper comparable with the approach of saving bad-signatures. In our opinion, the two approaches must be combined in order to prevent intrusion using bad-signature-storing and detecting infections using stochastic-methods.

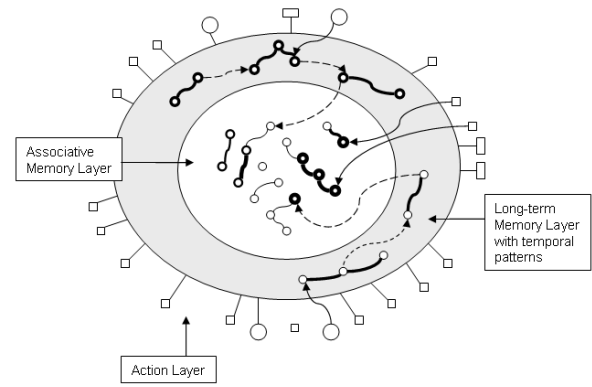


Figure 1: Visualisation of ANIMA-AR with the three layers: the short-term and the long-term memory as well as the action layer.

### 3. Expert system:

This approach uses expert systems like virus- or malware scanners in order to detect bad-signatures in packets. The advantage is that the expert systems work well and are up to date. Against it, the disadvantages are that the expert systems are not developed to detect intrusion in packets which are divided in a header- and a content-component as well as the packets can be fragmented. Also, the expert systems need too much computational power for checking all packets in a high-speed network (Antonatos, Anagnostakis, Polychronakis & Markatos 2004).

Consequently, it is important to find better approaches for bad-signature storage and string-matching between packet- and bad-signatures.

**Convention 2** *A network, e.g. the Internet, delivers packets from the source to the destination. These packets are checked by a system in order to remove bad-packets.*

*In this article, we talk about signatures of packets. A packet consists of a header- and a content-part. The signature of a packet is the string consisting of the header-string concatenated with the content-string. If we talk about checking a packet, the system creates the packet-string out of the packet and ANIMA-ID checks this string and returns a value which evaluates whether a packet is bad or not.*

## 3 Description of ANIMA for association rules

In detail, ANIMA-AR receives a constant data-stream of transaction and stores the transactions in an undirected network. Each node represents one entity or cell. All atoms, which appeared together in at least one transaction, are connected automatically. Furthermore, all nodes and connections are weighted in order to save confidence and support. This weight is set if the connection is set. Every time, the node or connection appears in a transaction, the weight is increased and the weight of all other nodes and connections, which does not appear in the transaction, is decreased - an adaptive system. Thus, ANIMA-AR provides association rules with confidence and support as well as ANIMA-AR provides skeletons which describe the main association rules of the data stream (Schommer 2004). Anymore, ANIMA-AR does not only provide association rules, it also provides relations between these entities and cells. The idea of ANIMA-AR is visualised in figure 1.

For the detailed explanation of the behaviour of ANIMA-AR, we refer to the article (Schommer 2005).

## 4 Description of ANIMA for network intrusion detection

This Section describes how we use ANIMA-ID in order to store and to detect bad-signatures of intrusions. A bad-signature is a string. Each character of this string is stored in a network-node. Thereafter, the network-nodes, which store the string, are connected through directed connections. Consequently, this new network represents the string. For an appropriate identification of bad-signatures and a proper tolerance of good-signatures, the nodes and connections receive a weight. In this case, ANIMA-ID stores a new signature which is not present, the weight of the nodes is set to  $\frac{1}{\text{length\_signature}}$  and the weights of the connections is set to 0. Hence, the weight of the signature is 1 because all weights are added up.

If ANIMA-ID stores a bad-signature and a part of the signature is already stored in ANIMA-ID, the non-stored-part of the signature is added to the stored-part in ANIMA-ID and the weights are set properly. Contrary to the rule-based-storing, ANIMA-ID reduces redundancies and, therefore, the used storage-space is decreased. ANIMA-ID weights the connection if a signature is stored but the signature is not marked as bad. If this task is performed, stored signatures receive a new weight and the signatures are maybe not identified as bad. However, the part of the signature, which is now weighted, identifies the old signature as bad because it is a substring.

We figure out the detailed rules for storing in the Section 4.1.

In the last paragraph, we figured out the storing of bad-signatures. Now, we discuss the idea of packet-checking whether a packet is bad or not. The signature of the packet is given to ANIMA-ID and it returns a value for the signature:

- If the value is equal to 1, the signature is definitely bad and must be removed from the network in order to prevent an intrusion.
- If the value is equal to  $1 - \delta$  and  $|\delta| \leq |\Delta|$ , the signature is bad - with high probability - and the system should remove the packet from the network for the purpose of preventing an intrusion. The parameter  $\Delta$  describes how affine the network reacts to similar intrusion.
- If the value is equal to  $1 - \delta$  and  $|\delta| > |\Delta|$ , the signature is good - with high probability - and the network should deliver the packet to the destination.

We discuss in-depth the checking in the Section 4.2.

In the checking-process, ANIMA-ID detects also similar intrusion; this is analogue to the immune system which also identifies mutated antigens. For the adaptation, ANIMA-ID provides two features:

1. If ANIMA-ID checks a signature which is similar to an existing signature, ANIMA-ID will return a value close-by 1 and the network will remove the packet using the parameter  $\Delta$ .
2. If the system inserts a lot of similar signatures in ANIMA-ID - the signatures must have the same length and the end of one signature is the start of another signature - ANIMA-ID will detect combinations and mutations of these signatures.

Therefore, ANIMA-ID is an adaptive and dynamic system.

### 4.1 Definition Storing a Bad-Signature

In order to define the storing process, ANIMA-ID performs up-to four tasks: We assume that ANIMA-ID stores a bad-signature  $S$ .

1. ANIMA-ID checks the signature  $S$  and if the value is equal to 1, ANIMA-ID already detects the signature  $S$  as definitively bad and the storing-process finishes.
2. ANIMA-ID contains the whole signature  $S$  but the value for  $S$  is not equal 1, the value is  $\hat{S}$ . Thereafter, ANIMA-ID weights the existing signature  $S$ :

- If all connection of the signature  $S$  are weighted or no connection and the ingoing and outgoing connection are not weighted, ANIMA-ID weights the connection as follows:

For each connection of the network of signature  $S$ , ANIMA-ID sets the weight for the connection to

$$\frac{1 - \hat{S}}{\text{length}_S - 1}$$

and the storing-process finishes.

- If not all connections are weighted or at least one connection is weighted or all connections are weighted and the ingoing and outgoing connection is weighted, ANIMA-ID inserts the signature  $S$  again with task 4.

3. ANIMA-ID does not contain the whole signature  $S$ . However, it contains a substring of the Signature  $S$ . Hence, string  $S = S_1 S_2 S_3$ , ANIMA-ID contains already the string  $S_2$  with value  $\hat{S}_2$  and ANIMA-ID does not contain the strings  $S_1$  and  $S_3$ . Also, the value  $\hat{S}_2 \neq 1$  because this would be a contradiction to task 1.

ANIMA-ID inserts two networks for the strings  $S_1$  and  $S_3$ , each node of these networks represents a character, connects the nodes with directed connections and weights the connection with the value 0. Afterwards, ANIMA-ID connects the end of the network of string  $S_1$  with the start of the network of string  $S_2$  as well as the end of the network of the string  $S_2$  with the start of the network of the string  $S_3$ ; the effect is that ANIMA-ID contains the whole signature  $S$ . Now, ANIMA-ID has to set the right weights. The value of the signature  $S_2$  is  $\hat{S}_2$ . Then, ANIMA-ID sets the value of each node in  $S_1$  and  $S_3$  to

$$\frac{1 - \hat{S}_2}{\text{length}_{S_1} + \text{length}_{S_3}}$$

and the storage-process finishes.

4. ANIMA-ID does not contain any substring of the signature  $S$ . ANIMA-ID inserts a new network, the nodes represents a character and ANIMA-ID connects the nodes using directed connections so that the path through the network is like the signature-string. Afterwards, ANIMA-ID weights the connections with 0 and the nodes with  $\frac{1}{\text{length\_signature}}$ .

Consequently, ANIMA-ID stores the bad-signatures of intrusions in networks and these networks represent the known world of bad-signatures. This is similar to ANIMA-AR where ANIMA-AR represents the current association rules with support and confidence, cp. (Schommer 2005, Schommer 2004, Schroeder & Schommer 2005).

### 4.2 Definition Checking a Packet-Signature

In this Section, we discuss the way how ANIMA-ID checks a packet. The system gives ANIMA-ID a signature  $S$ . The checking is divided into two tasks:

1. Calculating the value for the signature  $S$ :  
For calculating the value for a signature  $S$ , ANIMA-ID calculates the value of all substrings and returns

the value which is nearest to 1:

$$Value(S) = next1_{\bar{S} \subseteq S}(\bar{S})$$

where the function  $next1$  returns the value next to 1.

ANIMA-ID calculates for a string  $\hat{S}$  the value as accumulate the weights of all nodes and, if all connections of the string have a weight unequal to 0, adding the sum of all connections of the signature.

2. Evaluation of the value, decision whether a packet is good or bad:

As described above, there exist three different value levels:

- If the value given from ANIMA-ID is equal to 1, the signature is definitively bad and must be removed from the network in order to prevent the intrusion.
- If the value is equal to  $1 - \delta$  and  $|\delta| \leq |\Delta|$ , the signature is similar to an existing and known bad-signature;  $\Delta$  is a parameter of the checking system and describes how affine the network for mutated intrusions is. Therefore, the network should remove the packet in order to prevent an intrusion.
- Otherwise, the value is far away from 1, in detail the value is equal to  $1 - \delta$  and  $|\delta| > |\Delta|$ , and the packet is good - with high probability - and the network should deliver it to the destination.

Hence, ANIMA-ID identifies novel intrusions and adapts to the current intrusions without forgetting the old ones.

### 4.3 Advantages - Disadvantages of ANIMA-ID

In this Section, we present the advantages and disadvantages of our approach ANIMA-ID.

#### 4.3.1 Advantages

- **Adaptive:**  
ANIMA-ID recognises similar intrusions and it is able to prevent the system from similar attacks, e.g. if an attack is known, a small mutated attack is also removed.
- **Online System:**  
The system gives always a value about the criticality of a packet based on the actual set of signatures.
- **Easy to Administrate:**  
The administrator can easily add new signatures of intrusions because the process of storing the signature is implemented in ANIMA-ID.
- **Storage-Space-Saving:**  
ANIMA-ID saves storage space because little redundancy is used in the saved signatures of intrusions.
- **Checking-Time:**  
In contrast to a rule-based system, the time to check a packet is reduced because of the little redundancies.
- **Gives a current view how bad-signatures can be found:**  
In the storage of bad-signatures in ANIMA-ID, the nodes and connections have weights. If a short path through a network has a weight which is equal to 1, this (sub-)network identifies a lot of intrusions and is an important bad-signature.  
These short signatures are hard to compute if a rule-based system is used.

#### 4.3.2 Disadvantages

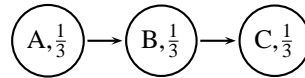
- **Removing bad-signatures:**  
ANIMA-ID provides an interface and logic for adding new bad-signatures. However, ANIMA-ID does not provide an interface and logic for removing bad-signatures. In this case, a new system must be used.  
A first and simple approach for this issue is a second version of ANIMA-ID wherein good-signatures are stored - similar to the white-lists of spam-blocker. These good-signatures describe packets which are always good and ANIMA-ID always returns the value 0, which means that ANIMA-ID identifies the packet as definitively good.
- **Stochastic-Approaches:**  
ANIMA-ID can only save strings. Consequently, there is no facility for adding other system, e.g. a stochastic-approach or a virus-/malware guard.

### 4.4 Example

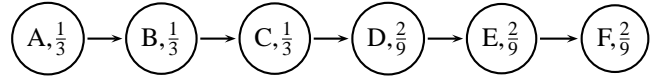
In this Section, we present an example of storing and checking in ANIMA-ID. In the storing-process, we use in this example all rules of the Section 4.1.

#### 4.4.1 Storing

Inserting the signature ABC using rule number 1:



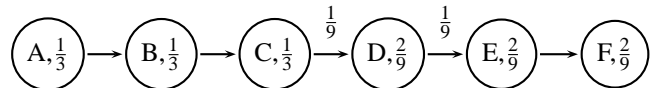
Inserting the signature CDEF using rule number 3:



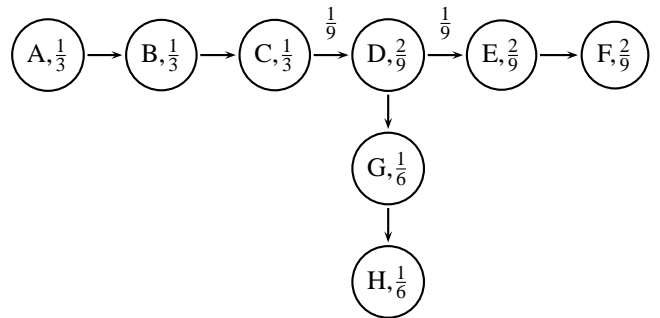
Inserting the signature ABC using rule number 1:

The signature already contains the value 1 and, hence, no changes in ANIMA-ID occur.

Inserting the signature CDE using rule number 2:



Inserting the signature CDGH using rule number 3:



#### 4.4.2 Checking

In this paragraph, we illustrate values for some signatures:

Signature	Value	Signature	Value
ABC	1	AB	$\frac{2}{3}$
CDEF	1	BCD	$\frac{8}{9}$
CDE	1	ABCDEF	$\frac{5}{3}$

## 4.5 Implementation of ANIMA-ID

After the theoretical phase - defining and analysing ANIMA-ID - finishes, we implemented ANIMA in order to test our approach. We implemented ANIMA in Java using OOP in order to keep platform-independent. However, we have to keep in mind that ANIMA-ID is research in progress and the implementation is an early prototype. The main aims of the prototype is to receive a stable version of ANIMA-ID as well as to test the idea. The scope of the implementation was not obtaining the fastest implementation.

The implementation provides all features described in this article as well as confirms the proof of completeness in Subsection 4.7. Furthermore, the implementation substantiates the advantages and disadvantages discussed in Subsection 4.3.

After implementation, we designed several scenarios in order to test ANIMA-ID. In all scenarios, we insert a number of bad-packet-signatures and, afterwards, we check packet-strings - good as well as bad packet-strings - in order to test if the returned values are correct. In all simulations, ANIMA-ID performs well without any problems or bugs. Furthermore, the storage-saving-feature is working well. Normally, ANIMA-ID saves about 10%-30% of storage compared to a rule-based system. Also, ANIMA-ID identifies reliable similar as well as mutated signatures of bad-packets.

### 4.5.1 Details of Test-Scenarios

The scenarios are quite similar and differ in the number of stored intrusion-descriptions as well as in the number of checked packets.

1. In the first scenario, we injected by hand some signatures of intrusions and afterwards checked some signatures of bad as well as good packets. In all situations, ANIMA-ID performs well; only some rounding errors occur.
2. In the second scenarios, we inserted automatically a lot of signatures of intrusions and afterwards we checked automatically several signatures of bad as well as good packets. Also in these scenarios, ANIMA-ID performs well and identifies all packets.
3. In the last scenarios, we inserted like in the last scenarios automatically a lot of signatures of intrusions and afterwards we checked automatically a lot of signatures of bad as well as good packets. Additionally, we also checked packets which contain a signature that is similar to an already stored signature. In this scenario, we test the adaptation of ANIMA-ID and it identifies the similar intrusions as well.

### 4.5.2 Details about the Implementation

We implemented ANIMA-ID in Java 1.4.2 using 545 code-lines and 313 additional code-lines for the GUI (Graphical User Interface). For the visualization of the graphs in ANIMA-ID, we used *jgraph* version 5.7.3 and *jgraphT* version 0.6.0.

We implemented four classes:

- **ANIMA\_Cell:**  
This class represents a cell of ANIMA-ID and stores the character, the weight of the cell and a list of references to the connections to the next as well as previous cells. The class is used as a simple *DTO* (Data Transfer Object).
- **ANIMA\_Connection:**  
A connection between two cells is represented by this class. It stores a reference to the source as well as destination cell and the weight of the connection. Like the *ANIMA\_Cell* class is this also a *DTO*.

- **ANIMA\_ID:**  
This is the class which implements ANIMA-ID and provides the following methods:
  - *Inserting\_Signature*: Insert a signature of a bad-packet.
  - *Check\_Packet*: Check a packet and return a value which describes how malicious the packet is.

This class obtains all cells and connections of ANIMA-ID and the only possibilities to access the networks in ANIMA-ID are the two methods above. It is possible to navigate through the networks using the classes *ANIMA\_Cell* and *ANIMA\_Connection*. Also, this class includes the networks.

- **GUI\_ANIMA\_ID:**  
This class visualises the networks and gives a front-end in order to test ANIMA-ID. For the visualisation of the networks, we use the java-projects *jgraph* and *jgraphT*. So, the handling of the GUI is intuitive.

### 4.5.3 Running-Time of the Implementation

The status of the project ANIMA-ID is that it is research in progress and the aims of this prototype is to test the basic idea as well as to visualize the idea. Furthermore, we used this prototype of ANIMA-ID in order to test the approach and to simulate scenarios. In the implementation, we focused on a fast prototype but the main aspects are to receive a stable version for good testing. Consequently, the running-times can be decreased.

Furthermore, a complete calculation of the best-, average- and worst-case running times are not possible by meeting the page-limitation of 10 pages. However, we can provide a basic analysing of the worst-case running-time in the implemented prototype:

We assume that we have  $c$  atoms in ANIMA-ID. This means that we saved signatures of bad-packets with  $c$  different characters. Now we distinguish between the checking of a packet and the learning of a new signature of a bad-packet.

#### Checking a packet:

We assume that the length of the signature of the checked-packet is  $n$ . Then, the running-time of the checking is bounded by  $O(n^3)$  and  $O(1)$  accesses to a hash-table with  $c$  items.

#### Learning a new signature of a bad-packet:

We assume that the length of the signature of the packet is  $n$ . Then is the running-time of the checking bounded by  $O(n^3)$  and  $O(n)$  accesses to a hash-table with  $O(n+c)$  items.

We have to keep in mind that the running-time depends on the implementation and our prototype is not revised to the highest performance.

## 4.6 Comparing the Storing of Bad-Signatures in SNORT with ANIMA-ID

SNORT uses another approach for storing the rules in order to check a packet. It differs between Chain-Headers containing Source/Destination IP-Address/Port and Chain-Options containing all other information of a packet, e.g. Content, TCP-Flags, Payload Size. SNORT checks the packet against the Chain-Headers and if the packet matches a Chain-Header, it will check the packet against the Chain-Options of the Chain-Header. If the packet matches one of the Chain-Options, an intrusion in the packet is identified. If the packet does not match any Chain-Header and Chain-Option, the packet does not include an intrusion.

The advantages of this approach are that the checking is divided into the fast header-checking and the not so fast complete packet-checking. Furthermore, SNORT

processes the checking after such a graph of Chain-Headers and -Options; this eliminates checking with uninteresting rules. The disadvantages are that the rules must be written by hand and SNORT does not eliminate redundancies in the rules which need running-time as well as storage-space.

ANIMA-ID will also only check the packet against the interesting parts of the network in ANIMA-ID and so there is a similar feature which eliminates uninteresting checks. Consequently, the running-time of the checking depends on the implementation which will be not focussed in this article.

A good approach is to combine the both system. This approach uses the Chain-Headers from Snort in order to classify the intrusions and the Chain-Options of a Chain-Header are replaced by an ANIMA-ID. Hence, both system are combined in order to receive the advantages of both systems.

#### 4.7 Proof for completeness of the rules

In this Section, we prove that the rules introduced in Section 4.1 are complete. This means that all added bad-signatures are identified as bad - value is equal to 1 - and all good signatures are not identifies as definitively bad - value is unequal to 1.

**Convention 3** Let  $S_1$  and  $S_2$  two strings. Thus,  $S_1 \subseteq S_2$  iff  $S_1$  is a substring of  $S_2$ . Also, let  $|S|$  be the length of the signature-string  $S$ .

The proof of completeness is not valid for all possible signatures. Firstly, we will show in the next paragraph that if ANIMA-ID saves a signature ANIMA-ID will identify this signature as definitively bad (value = 1). Secondly, we will show, that for mostly all signatures, which are not stored in ANIMA-ID, will not receive a value equal to 1. Unfortunately, in some cases if a signature is stored another signature, which is not stored in ANIMA-ID, receives the value 1. However, these cases are far away from practice. The signatures of these cases are short (at most six Bytes) and in practice the signature-length are normally at least  $1kByte = 1024Bytes$ . Consequently, the set of rules is complete for typical used signatures.

Anymore, if the systems inserts signatures  $S_i$  with  $i \in \mathbf{N}$ , each  $S_i$  has the length  $x$  and  $S_i = S_{i_1}S_{i_2}$  as well as  $S_{i+1} = S_{i_2}S_{i+1_2}$  with  $1 \leq |S_{i_j}| < x$ , then all  $S \subseteq \sum_i S_i$  with  $|S| = x$  are malicious. However, this is not a problem because these signatures  $S$  describe combinations and mutations of existing intrusions.

Now, we present the proof in detail:

##### 4.7.1 A signature $S$ is stored and, afterwards, not identified as definitively bad

ANIMA-ID stores the signature  $S$  in a time-step and sets the value to 1. Accordingly, ANIMA-ID returns for the signature  $S$  the value 1 which means that the signature is definitively bad.

Now, we assume that ANIMA-ID will at any time not identify the signature  $S$ , ANIMA-ID will return a value unequal 1 for the signature  $S$ . Hence, ANIMA-ID must increase or decrease the value using other weights. In order to contradict this, we discuss all rules of Section 4.1 if a rule can diversify the weights:

1. The first rule does not perform any action on the networks saved in ANIMA-ID. Consequently, the weights of the signature  $S$  are unvaried.
2. The second rule is the important rule in this proof because it varies weights of existing signatures.

If ANIMA-ID only weights a substring of the signature  $S$  the value of  $S$  is independent from the connection-weights in  $S$ , cp. the Section 4.2, or the new inserted substring is also definitely malicious.

If ANIMA-ID weights the whole string  $S$ , the system inserts a string  $\hat{S} \supseteq S$ . This is a contradiction to rule 1 because the value for the signature  $\hat{S}$  is 1 and ANIMA-ID does not weight the connections.

Hence, the weights of the stored signature  $S$  are unvaried.

3. In the third rule, ANIMA-ID stores a new signature affixing to a current existing network. Also, ANIMA-ID sets the weight of the new inserted nodes and connections in order to receive a value equal to 1 for the stored signature. Therefore, the weights of the stored signature  $S$  are unvaried.
4. The fourth rule inserts a new signature in a new network and sets the weights of the nodes and connections in order to receive a value of 1 for the new signature. Hence, the weights of the stored signature  $S$  are unvaried.

Thus, this part of the proof is shown.

##### 4.7.2 A signature $S$ is not bad and ANIMA-ID identifies it as definitively bad

In this Section, we prove that a signature  $S$  which is not saved in ANIMA-ID cannot receive the value 1; only in some non-practical cases can these occur. We assume that there is never a signature  $\hat{S} \subseteq S$  inserted. Consequently, there is no network which represents the signature  $S$  with a value equal to 1.

Now, we check all rules of the Section 4.1 in order to find a rule which can inserts or weights a signature  $S$  with value 1 when another signature  $\hat{S} \not\subseteq S$  is inserted.

1. The first rule does not change anything in ANIMA-ID and, consequently, there cannot emerge out of this rule a signature  $S$  with value 1.
2. The signature  $S$  is in ANIMA-ID and contains a value unequal to 1. Now, the connections of the signature  $S$  will be weighted.

Firstly, we note that ANIMA-ID must weight the connections of the whole signature  $S$  because a partly weighting has no consequence for the value of the signature  $S$ . Let  $\hat{S}$  the new signature and let  $\tilde{S} = S_1SS_2$ . Furthermore, let  $\tilde{X}$  the value of the signature  $X$  with  $X \in \{S_1, S, S_2, \hat{S}\}$ .

Then,

$$\tilde{\tilde{S}} = 1 = \tilde{S}_1 + \tilde{S} + \tilde{S}_2$$

and  $\tilde{S}_1 \neq 1$ ,  $\tilde{S} \neq 1$  and  $\tilde{S}_2 \neq 1$ .

Let  $\tilde{X} = N(X) + C(X)$  where  $N(X)$  is the weight of the nodes of the signature  $X$  and  $C(X)$  is the weight of the connections of the signature  $X$ . Then it follows that

$$\begin{aligned} 1 &= \tilde{S}_1 + \tilde{S} + \tilde{S}_2 \\ &= N(S_1) + C(S_1) + N(S) + C(S) + N(S_2) + C(S_2) \end{aligned}$$

If  $C(S_1SS_2) = x$  then  $N(S_1SS_2) = 1 - x$ . Also, because the weighting of the connections is uniform,  $C(X) = \frac{x}{|X|}$  with  $X \in \{S_1, S_2, S\}$ .

**Note:**

$N(X) \geq 0$  for all signatures  $X$ .

**Proof:**

If a signature is inserted in ANIMA-ID, ANIMA-ID sets the right values for the nodes. We discuss the four rules: The first and the second rule do not influence the weights of the nodes.

The third rule is valid because ANIMA-ID inserts a signature where a sub-signature is already stored. Hence, the value of the stored signature is in  $]0, 1[$  and the weights of the nodes of the inserted signature are also in  $]0, 1[$ .

The fourth rule sets the weights of the nodes equal to  $\frac{1}{|X|}$ , where  $X$  is the inserted signature-string. Hence, all weights are at least 0.

Hence, the note is valid.

Now, we assume that  $\tilde{S} = 1$  because then we receive a signature with value equal to 1 without saving the signature. Out of  $\tilde{S} = 1$  follows  $N(S) + C(S) = 1$ .

We know from above that

$$N(S_1) + C(S_1) + N(S) + C(S) + N(S_2) + C(S_2) = 1$$

and then

$$N(S_1) + C(S_1) + N(S_2) + C(S_2) = 0$$

From above we know that  $C(S_1) = \frac{x}{|S_1|}$  and  $C(S_2) = \frac{x}{|S_2|}$  and it follows that

$$N(S_1) + \frac{x}{|S_1|} + N(S_2) + \frac{x}{|S_2|} = 0$$

Also, we know from above  $N(S_1 S_2) = 1 - x$  as well as from our assumption  $S(S) + N(S) = 1$  and  $N(S) = \frac{x}{|S|}$ .

Hence,

$$\begin{aligned} N(S_1) + N(S_2) &= 1 - x - N(S) \\ &= 1 - x - 1 + \frac{x}{|S|} \\ &= \frac{x}{|S|} - x \end{aligned}$$

Thus, it follows:

$$\frac{x}{|S|} - x + \frac{x}{|S_1|} + \frac{x}{|S_2|} = 0$$

and

$$x(|S_1| \cdot |S_2| - |S| \cdot |S_1| \cdot |S_2| + |S| \cdot |S_2| + |S| \cdot |S_1|) = 0$$

From this equation it follows that either  $x = 0$  or

$$(|S_1| \cdot |S_2| - |S| \cdot |S_1| \cdot |S_2| + |S| \cdot |S_2| + |S| \cdot |S_1|) = 0$$

must apply. However,  $x = 0$  means that we do not set a weight on the connections and this is a contradiction.

Hence, we have to solve the diophantine equation

$$(|S_1| \cdot |S_2| - |S| \cdot |S_1| \cdot |S_2| + |S| \cdot |S_2| + |S| \cdot |S_1|) = 0$$

**Assumption:**

The equation

$$(|S_1| \cdot |S_2| - |S| \cdot |S_1| \cdot |S_2| + |S| \cdot |S_2| + |S| \cdot |S_1|) = 0$$

has the solutions:

$$\begin{array}{lll} 2 & 3 & 6 \text{ and it's permutations} \\ 2 & 4 & 4 \text{ and it's permutations} \\ 3 & 3 & 3 \end{array}$$

Proof:

We differ in the proof between some intervals for the variables  $S$ ,  $S_1$  and  $S_2$ . The parameters are in the interval  $[0, \infty[$ .

Firstly, we focus on the interval  $[1, 1000]$  for all three parameters. Then, we receive the solutions in the assumption above using the computer.

Secondly, we focus on the interval  $]1000, \infty[$  for all three parameters. Therefore,  $|S| \cdot |S_1| \cdot |S_2|$  is always greater than  $|S_1| \cdot |S_2| + |S| \cdot |S_2| + |S| \cdot |S_1|$  and we do not find any solution for the equation.

Thirdly, one parameter is in the interval  $]1000, \infty[$ , one parameter is in the interval  $[3, \infty[$  and one parameter is in the interval  $[2, \infty[$ . It is easy to see that the values 1001, 3 and 2 are interesting and all other higher variables have the same behaviour. If we use these values then  $|S| \cdot |S_1| \cdot |S_2|$  is equal to 6006 and  $|S_1| \cdot |S_2| + |S| \cdot |S_2| + |S| \cdot |S_1|$  is equal to 5011. Hence, these intervals do not provide any solution for the treated equation.

Fourthly, one parameter is in the interval  $]1000, \infty[$ , one parameter is in the interval  $[1, 3[$  and one parameter is equal to 1; w.l.o.g.  $|S_2| = 1$ . Thus, the equation has the form  $|S_1| \cdot |S| = |S_1| \cdot |S| + |S_1| + |S|$  and thus  $0 = |S_1| + |S|$  which is impossible.

Fifthly, the parameters are in  $\mathbf{N}_0$  and at least one parameter is equal to 0. W.l.o.g.  $|S_2| = 0$ . Then we receive the equation  $|S_1| \cdot |S| = 0$  and, hence,  $|S_1| = 0$  or  $|S| = 0$ . Accordingly, we only insert one string and this string must be according to the construction of this proof in ANIMA-ID. If two parameters are equal to 0, we receive the same result and, consequently, these intervals for the parameters does not provide any new solution.

Consequently, the second rule provides only the solutions of the assumption.

- The system inserts the signature  $\hat{S} = S_1 S_2 S_3$  in ANIMA-ID and the signature  $S_2$  is already stored. We note, that the value of  $S_2$  is unequal to 1 because then the first rule is valid because  $\hat{S}$  is definitively bad.

Let  $\tilde{S}_i$  be the value for the signature  $S_i$  for  $i \in \{1, 2, 3\}$ . Hence,  $\tilde{S}_1 + \tilde{S}_2 + \tilde{S}_3 = 1$  because  $\hat{S}$  will be inserted as well as  $\tilde{S}_1 = \tilde{S}_3$  because the weighting is uniform.

We say w.l.o.g. that  $S = S_1 S_2$ . It is clear that  $\tilde{S}_1 + \tilde{S}_2 = 1 - \tilde{S}_3$ . It follows that

$$\tilde{S}_1 + \tilde{S}_2 = 1 \longrightarrow \tilde{S}_3 = 0$$

and with  $\tilde{S}_1 = \tilde{S}_3$

$$\tilde{S}_1 = 0 \longrightarrow \tilde{S}_2 = 1$$

which is a contradiction to rule 1 the sub-signature  $S_2$  is already identified as malicious.

Hence, the signature  $S$  would not receive a weight equal to 1.

Anymore, ANIMA-ID detects also combinations and mutations of existing intrusions:

Let  $i, x \in \mathbf{N}$ ,  $|S_i| = x$ ,  $S_i$  signatures of intrusions and  $S_i = S_{i_1} S_{i_2}$  as well as  $S_{i+1} = S_{i+1_1} S_{i+1_2}$  with  $S_{i_2} = S_{i+1_1}$  and  $1 \leq S_{i_j} < x$ ; this means that the system inserts in ANIMA-ID a lot of signatures where the end of a signature is the start of another signature. Consequently, in ANIMA-ID evolves a network, each node

has the same value  $\frac{1}{x}$  as well as each connection has the value 0. Therewith, each signature  $S \subseteq \sum_i S_i$  with  $|S| = x$  is definitively malicious; this means ostensive that each sub-signature of the concatenation with length  $x$  is also malicious. However, these signatures belong to combinations and mutations of existing intrusions and should be detected by ANIMA-ID.

4. The fourth rule inserts a signature which is uniformly weighted. Hence, the signature  $S$ , which may be only a substring, cannot receive the weight 1.

Consequently, the second and last part of the proof is shown and the rules are complete for practical signatures.

## 5 Applications for ANIMA-ID

In this Section we present several applications for ANIMA-ID. However we have to keep in mind that ANIMA-ID is not a running NIDS, it is research in progress as well as just a component of a NIDS. Anyhow, we present some feasible applications which are currently not running.

Consequently from design, ANIMA-ID's main application is in the intrusion detection of computer networks. As well, ANIMA-ID also performs well in the area of intrusion detection in other networks like e.g. communication networks, data streams and satellite communications. Also, it is possible to use ANIMA-ID in order to detect intrusions in general networks, computer environments or data streams. Generalised, ANIMA-ID can be used in order to detect intrusions in all data-masses.

One practical example is to observe communication channels and the Internet in order to detect attacks, trojans and viruses. It is possible to use ANIMA-ID in an academic as well as commercial network as a part of the NIDS securing the Internet-Gateway in order to secure the network. Another practical example is to use ANIMA-ID as a lightweight agent in an artificial immune system which is used for intrusion detection in computer networks. We are currently building up such an artificial immune system in our group.

## 6 Conclusion

In this article, we presented a new system for saving bad-signatures of intrusions and for checking packets against these saved bad-signatures. ANIMA-ID has the advantages that it is fast, adaptive, online and storage-space-saving. Also, ANIMA-ID is easy to implement because it only consists of directed and weighted graphs. If ANIMA-ID is used for bad-signature describing and for checking packets, the NIDS will be enhanced and improved because current intrusion-describing systems use normally a rule-based-system. We implemented ANIMA-ID and simulated scenarios in order to test ANIMA-ID. However, it is interesting to enhance ANIMA-ID, e.g. implementing a short-term and a long-term memory for saving intrusions in the long-term memory and for saving short-term restrictions in the short-term memory. Anymore, we will use ANIMA-ID in an artificial immune system for network intrusion detection as a lightweight agent. Furthermore, it is also interesting to enhance ANIMA-ID in order to automatically detect abnormal packets which are different from the standard packets in a network. These next steps will be our challenge for the future.

## References

Antonatos, S., Anagnostakis, K., Polychronakis, M. & Markatos, E. (2004), 'Performance analysis of content matching intrusion detection systems', *SAINT* 4.

Debar, H., Dacier, M. & Wespi, A. (1998), 'Towards a taxonomy of intrusion-detection systems', *Computer Networks* 31(9), 805–822.

Hofmeyr, S. A. & Forrest, S. (1999), 'Immunity by design: An artificial immune system', *Proceedings of the Genetic and Evolutionary Computation Conference* 2, 1289–1296.

Hofmeyr, S. A. & Forrest, S. (2000a), 'Architecture for an artificial immune system', *Evolutionary Computation* 8(4), 443–473.

Hofmeyr, S. A. & Forrest, S. (2000b), 'Immunology as information processing'.

Janakiraman, R., Waldvogel, M. & Zhang, Q. (2003), 'Indra: A peer-to-peer approach to network intrusion detection and prevention', *Proceedings of IEEE WETICE 2003*.

Lazarevic, A., Ertoz, L., Ozgur, A., Srivastava, J. & Kumar, V. (2003), 'A comparative study of anomaly detection schemes in network intrusion detection', *Proceedings of Third SIAM Conference on Data Mining* 3.

Leung, K. & Leckie, C. (2005), 'Unsupervised anomaly detection in network intrusion detection using clusters', *Australasian Computer Science Conference* 28.

Porrás, P. A. & Neumann, P. G. (1997), 'EMERALD: Event monitoring enabling responses to anomalous live disturbances', pp. 353–365.

Roesch, M. (1999), 'Snort - lightweight intrusion detection for networks', *LISA* 13, 229–238.

Schommer, C. (2004), 'An incremental neural-based method to discover temporal skeletons in transactional data streams', *RASC, Proceedings of 2004 Recent Advances in Soft Computing, Nottingham, England*.

Schommer, C. (2005), 'Incremental discovery of association rules with dynamic neural cells', *ECAI, Workshop on Symbolic Networks, Valencia, Spain*.

Schroeder, B. & Schommer, C. (2005), 'Anima: Associate memories for categorical data streams. proceedings of the 3rd international conference on computer science and its applications', *ICCSA, Full Conference on Computer Science Applications, San Diego, USA*.

Sekar, R., Gupta, A., Frullo, J., Shanbhag, T., Tiwari, A., Yang, H. & Zhou, S. (2002), Specification-based anomaly detection: a new approach for detecting network intrusions, Vol. 9, pp. 265–274.

Snapp, S. R., Brentano, J., Dias, G. V., Goan, T. L., Heberlein, L. T., Lin Ho, C., Levitt, K. N., Mukherjee, B., Smaha, S. E., Grance, T., Teal, D. M. & Mansur, D. (1991), 'DIDS (distributed intrusion detection system) - motivation, architecture, and an early prototype', *National Computer Security Conference* 14, 167–176.

Spafford, E. H. & Zamboni, D. (2000), 'Intrusion detection using autonomous agents', *Computer Networks* 34, 547–570.

Staniford-Chen, S., Cheung, S., Crawford, R., Dilger, M., Frank, J., Hoagland, J., Levitt, K., Wee, C., Yip, R. & Zerkle, D. (1996), 'Grids - a graph based intrusion detection system for large networks', *National Information Systems Security Conference* 19.