Intrusion Detection Systems in Networked Embedded Systems

Christian Vestlund
chrve@ida.liu.se, M.Sc. student, TDDD17, ADIT, Linköping University

Abstract

Automotive systems are becoming more networked than ever before, with more electronics and computer systems embedded in them. The increased networking abilities for automotive systems implies a higher risk that someone with bad intentions can manipulate parts of the system. To avoid such manipulations, intrusion detection for networked embedded systems has to be deployed to ensure the integrity of the system. This paper provides a study of some intrusion detection system techniques that are used in ordinary computer systems and networks. This study is made to see if they are fit to deploy in an automotive networked embedded system or can be manipulated to fit the requirements of an automotive networked embedded system.

1 Introduction

Computer network systems have for a long time been subject for security research and many useful approaches have emerged over the years. One of the most used techniques today is probably the intrusion detection systems (IDS) which will be the subject for this paper. More specifically an approach to deploy desktop computer IDSs in networked embedded systems will be given.

In our age more and more embedded systems comes into our everyday life, lets take automobiles as an example. Features like intelligent parking assistants, blind-spot information, navigation systems with real-time updates are just some of the computer systems you may find in a modern automobile. Therefore it is crucial that these systems works as intended and that no one can intentionally insert malicious behaviour into the systems.

With the mentioned reasons in mind it is easy to see why there is a need for IDSs in embedded systems. In this paper a definition of what an IDS is and description and examination of different types of IDSs used in computer networks will be provided. A definition of what a networked embedded system is will also be provided so the target system where the IDS will be deployed on is defined. After the definitions and descriptions a section is devoted to define/construct an IDS that in theory would be able to fulfill the requirements for an IDS on a networked embedded system.

1.1 What is an intrusion detection system?

The easy and non-technical way of describing an IDS is to say that it is a "burglar alarm for computers that let someone know about the burglary". With that description most people can comprehend the function of an IDS. But that does not say anything about the functionality behind the IDS, how does it get to know about the "burglary"? How can it find out what was the vulnerability that is to blame for the possibility for the "burglary"? A vulnerability in the context of a house burglary would be for example an unlocked door or open window which leads to the possibility to commit a burglary. From here on the term "intrusion" is going to be used instead of "burglary".

An intrusion can be either physical or non-physical i.e. dropping a probe on to a circuit and read out the bits that flies by or writing malicious network packets. This paper will focus on the non-physical branch of intrusion detection, for physical intrusion detection readers are referred to papers on tamper evidence.

There are several different techniques that can be deployed when constructing an ordinary IDS for a computer system or network. Most of them, but not all, perform "real-time" detection which is going to be the focus for this paper. This means that physical and equipment under installation etc. is not considered when designing the IDS. In section 2 descriptions of several of the available techniques will be provided, one of the more important parts of intrusion detection, audit, will be presented in the next section.

1.2 Audit

Audit is probably the most important part of any security system. If there is not some kind of auditing, how can a system detect that something is wrong?

Audit is here defined as the process of generating, recording, and reviewing a chronological record of system events [1].
By using this definition it is very easy to see the importance of auditing. The most interesting part is the automation of the auditing process and also how to append an alert system after the reviewing of the events. The reviewing part of the audit process is often done with regards to certain rules or policys to determine if the events in a system are considered normal or not. If they are not considered normal by the reviewing process there should be a system to alert the management about the detected anomaly.

1.3 What is a networked embedded system?

Now there is a "definition" of what an IDS is and what the conceptual idea of it is. As earlier mentioned there exists several techniques to construct an IDS for a computer system or network. To be able to construct an IDS for a networked embedded system a definition of the term "networked embedded system" need to be given.

For that it is first needed to define what an *embedded* system is and secondly what a *networked embedded system* is.

Formal Definition: Embedded System

An embedded system is a computing system with a combination of hardware and software designed to perform one or a few dedicated functions [21].
Embedded systems can be a part of a larger system, e.g. in cars where the antilock braking system is only a part of the whole system in the car. The example with the car will return later on in this paper since it is a very good subject for this study. This leads to a definition of a networked embedded system.

Example: Embedded system

A mp3 player is an embedded system, it is a computing system that allows a user to store audio files and play them. A general purpose device such as a handheld computer is not truly an embedded system.

Definition: Networked Embedded System

The view used in this paper is that a networked embedded system is a system that consists of several embedded systems sharing communication resources in a larger system [21].

Example: Networked embedded system

A networked embedded system in the context of a car is several different systems that are interconnected. In a car there exists several different subsystems such as Controller Area Network (CAN) which handles real-time communication between controllers in a car and Media Oriented System Transport (MOST) which handles multimedia for in-car entertainment. These two systems are today interconnected and every controller, no matter on what subsystem, is able to send messages to any other controller in the car [21].

2 Types of IDS

As mentioned there are several different techniques for IDSs that can be used when constructing such a system. In this section some of the techniques (or at least their concepts) for IDSs will be explored and examination of their strengths and weaknesses will also be done.

In literature and in implementation the first difference people make between types of IDSs is if they are Host-based or if they are Network-based. This difference in practice depends on where they gather their information, if they gather it from the traffic in the network or from the host machines audit logs, system calls etc. There are some distributed IDSs as well, sensor networks is a good example of one. In a sensor network there are several sensors that provides a central unit with information, from that information the central unit can decide if a sensor has been broken or compromised [5].

For the sake of understanding the rest of the paper the following two definitions are needed:

Definition: Host-based IDS

Host-based IDSs (HIDS) typically resides on a host and examines the internal state of it by analyzing logs of system calls, access or modification of files etc.

Definition: Network-based IDS

Network-based IDSs (NIDS) examines data traffic in transit between hosts in the network.

These two kinds of IDSs does not serve the same purpose and therefore a NIDS can not be used instead of a HIDS for an example. HIDS are more suitable for use in non-networked environments to be most effective but they can be used together with NIDSs to increase the chances of detecting intrusions. Both these approaches will be discussed more in depth in section 3.2.4 and 3.2.5.

2.1 Techniques in IDSs

This section provides descriptions of the different techniques used in IDSs for ordinary computers. This includes signature-based, anomaly-based, hybrids etc. Strengths and weaknesses will also be discussed regarding their use in regular computer systems/networks.

2.1.1 Signature-based

A signature-based IDS makes use of rules, more specifically rules describing malicious behaviour. This means that a signature-based IDS compares sequences of events, patterns of data etc. to the rules it has stored in its knowledge database. These rules may only apply to a single packet from the network or a small sequence of system calls but it is possible to make more advanced rules that can apply to series of packets over the network. These advanced rules can evolve into more complex rules with hundreds if lines of code [22].

Example: Snort rule

```
alert tcp $EXTERNAL_NET any -> $HOME_NET
80 (msg: "BOTNET TESTING RULE: Candidate
to detect adorelyric.com-like malware";
flow:to_server; content:"POST /";
depth: 10;  content:".png HTTP/1.1";
depth: 30; content:  "Content-Type:
application/x-www-form-urlencoded";
depth: 200; sid: 1100001; rev:1;)
```

This rule checks for tcp packets flowing to a server on port 80 containing the string "POST /" in the first 10 bytes and the string ".png HTTP/1.1" in the first 30 bytes [23].

End of example.

When making use of a signature-based IDS there is a constant need to feed its knowledge base (the rule database) with new rules for every new exploit that is detected, and today there exists **a lot** of exploits for a wide variety of systems [24]. That said it is easy to understand that such an IDS needs to have an extensive knowledge base to cover most of the exploits, but that is not guaranteed to help.

A rule often makes use of pattern matching of data and are generally fixed for each case of an attack. The same attack can be done in several different ways or the same way but with different code/signature. That means that if there only is a slight change in the pattern of the attack, the IDS will miss it and the adversary will succeed in his task. Even more troublesome is that it is possible to desynchronize an IDS which tracks whole connections [4].

The biggest problem with a signature-based IDS it that it can not detect novel attacks because it does not have a signature stored in its knowledge base for any new attacks. This is a very disturbing fact when dealing with systems, mostly because there exists a huge amount of cases of anomalous behaviour compared to legitimate behaviour.

To wrap it up for the signature-based IDSs it can be seen that it is easy to understand how signatures work and are used but they are at disadvantage in an environment where that majority of the attacks can not predicted.

2.1.2 Anomaly-based

Anomaly-based IDSs typically creates profiles of usage over some time. After having created a profile an anomaly-based IDS can detect intrusion by examining how different the observed behaviour is from the produced profile. This might raise some questions about false alarms depending on how aggressive the IDS is, and that is also one of the drawbacks of this kind of IDSs. Depending on how maliciousness of an event is determined a lower or higher false positive rate is obtained. False positives are events that are reported malicious while they are completely innocent.

Aggressiveness of an IDS is mostly defined by threshold values, in statistical IDSs this can be how many connections that are established during a specific time period. An example of how this is done in data mining-based IDSs is given below.

Example: False positives and aggressiveness in Data mining-based IDSs

In a data mining-based IDS each event can be represented as a point in a n-dimensional space, each dimension corresponding to an attribute. When a profile is established points are placed in this n-dimensional space and a centroid is calculated. A centroid is a mean value of all the points in the space. When intrusion

detection is taking place with this model, every new event is compared to the centroid, if the difference is to big an alarm is raised. The difference between the centroid and a new event is the value used for detection in this case, and this value can be adjusted to create a strict profile or a loose profile.

Let the centroid be a vector consisting of five attribute values $< 3, 4, 12, 5, 7 >$ and the a new point with the attribute vector $< 6, 3, 3, 8, 1 >$ arrives to the IDS. The euclidean distance between the centroid and the new point is

$$\begin{aligned} &d(\text{centroid, new point}) = \\ &= \sqrt{(c_1 - n_1)^2 + (c_2 - n_2)^2 + \ldots + (c_5 - n_5)^2} = \\ &= \sqrt{(3-6)^2 + (4-3)^2 + (12-3)^2 + \ldots} = \\ &= \sqrt{136} \approx 11.7 \end{aligned}$$

This distance is the threshold value in the data mining case. This distance can be calculated in other ways than just euclidean distance depending on what variable types there is. That is beyond the scope of this paper and readers are referred to literature on cluster analysis.

End of example.

This problem has been subject to several research projects when the aim is to develop an anomaly-based IDS, the lower the false positive rate (while keeping the true positive rate high), the better the system. As an example of this the attention can be turned to Tan and Xi (2008) and Tandon&Chan (2006) [3, 2] where they examine the possibility of using Hidden semi-Markov model (HSMM) for anomaly detection in a specific program by looking at audit logs and analyzing system call sequences. Some more detailed examples of anomaly-based techniques will be provided in section 3.

In ordinary computer systems anomaly-based IDSs allow their pre-established usage profile to evolve with the legitimate usage of the system. After it has evaluated some events it will recalculate the profile values, thus letting it evolve with the evolving user of the system. This is a great way to get around problems with false positives while still letting the user change his behaviour bit by bit. But there is also disadvantages with this mechanism, a patient adversary can induce this evolving of the IDS to allow him to use the system for his malicious intents without any alarm going off.

Example: Evolving IDS

The same system as the previous example is used to illustrate the evolving of IDSs. When a new point is compared to the centroid and is found to be normal the centroid value is recalculated using the new point as well as the old points. Thus the centroid values changes towards the new normal behaviour.

Let a usage profile consist of 5 points with the attribute vectors $v_1 =< 4, 6, 1, 7, 8 >, v_2 =< 8, 11, 3, 2, 12 >, v_3 =< 5, 4, 8, 2, 9 >, v_4 =< 3, 1, 5, 2, 2 >, v_5 =< 10, 13, 3, 7, 9 >$. These 5 points gives a centroid with the attribute vector $c =< 6, 7, 4, 4, 8 >$ calculated as the mean value of the 5 points in the usage profile. A new point $n =< 12, 7, 4, 10, 14 >$ is accepted as normal and the centroid has to be recalculated as the mean of the 5 old points and the new point. The new centroid attribute vector will be $c =< 7, 7, 4, 5, 9 >$.

End of example.

The strengths of anomaly-based systems compared to the signature-based is that anomaly-based systems requires no predefined rules for detection of attacks, therefore they can also detect new attacks. The disadvantage is that anomaly-based systems need to establish a profile of normal usage. This often requires a training period where the system supposedly behaves as intended, without any anomalous behaviour whatsoever. This is a very hard requirement to fulfill in ordinary networked systems which often are connected to the Internet which is overflowing with malware. If it were possible to establish a profile for normal behaviour without the risk of malicious behaviour during the training period the effectiveness of an anomaly-based IDS increases significantly. This is because anomaly-based IDSs reacts on events that deviates from the established profile and if the profile contains malicious data then malicious events may not be registered as malicious [10]. Another disadvantage with anomaly-based detection is that it just detects anomalies, it does not necessarily give you any additional information on what went wrong. This can be solved using hybrid systems which can use signature-based classification, which will be discussed in the section 2.1.3.

There exists a great variety of anomaly-based detection schemes, to go through all of them is not essential for further understanding of this paper. The techniques that are discussed in the construction section will be described in more detail because it will be needed to understand the reasoning behind the choices during the "construction" of the new IDS for networked embedded systems. For those who are interested in anomaly-based IDSs a few additional techniques is mentioned

here; Bayesian networks, Markov models, Genetic algorithms, neural networks, clustering etc.

2.1.3 Hybrid Systems

Hybrid IDSs comes in a lot of shapes and sizes, a few of them will be described in the rest of this paper both in this section and section 3.

Decision Trees (DTs) are a good example of a hybrid system, DTs are often used for classification e.g. classification of behaviour, data or attacks [7, 8]. To be able to use a decision tree properly the data must be divided into several classes which can be labeled for human interpretability of the classification. For a human it can be hard to look at a cluster of data points and understand what they mean, therefore it can be needed to assign each cluster an appropriate label e.g. spam attacks, DoS attacks etc. An example of a decision tree could look something like in figure 1 where it can be seen that every node is analogue to a cluster/class of data, and every edge represents a rule from which the upper class can be further divided into smaller classes which can be labeled as "normal" or "anomalous" and also more specifically give the reason to why it is labeled as anomalous. This is something that is very useful in a lot of cases, to know exactly which parameter(s) gives the anomalous decision.

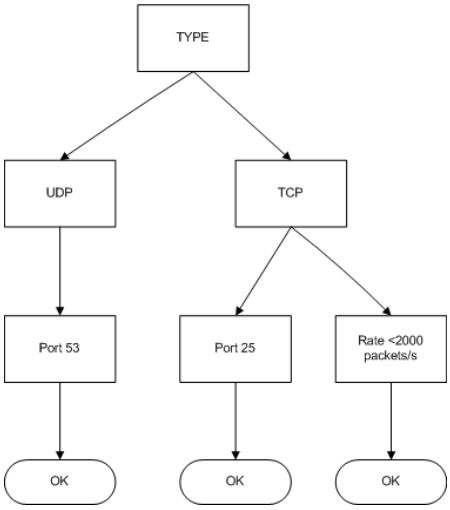

Figure 1: Example of a decision tree for network traffic.

2.1.4 Artificial Immune Systems

A special case of anomaly-based IDSs is the Artificial Immune Systems (AISs). When AISs are used they are mainly used to try to enforce the concept of self/non-self onto a computer system it is inspecting. The concept is ideal for intrusion detection, considering how immune systems work in biological organisms. The problem here is how to define what data belongs to the system and what is malicious data.

Several different papers state important principles for the design of an IDS to be optimal in the sense of mimicking a human immune system [15, 16, 17]. The properties stated in Kim et al. (2007) [15] will be used in this paper since they correspond to the best compiled list of the found properties for AISs.

Distributed. A distributed IDS supports several properties that are desirable to achieve when creating such a system; robustness, configurability, extendability and scalability. By not having a single point of failure the system is much more reliable and robust than a system with a centralized IDS. Another advantage of this principle is that several IDSs can operate concurrently and also co-operate and exchange information about attack signatures.

Self-organised. A self-organised IDS detects new attacks automatically and responds to compromised components by eliminating them without outside management or predefined attack signatures.

Lightweight. The lightweight principle says that the IDS should not impose large overhead or heavy burden on the components CPU and I/O. The principle of imperfect detection is embedded in the Lightweight principle. Imperfect detection allows the system to be very flexible and not consume a huge amount of resources by keeping the entire database of attack and/or non-attack signatures at hand. This can be achieved by only keeping a subset of detectors in memory and change the subset used during run-time [9].

Multi-layered. This principle also increases the robustness of the overall system and differs from the distributed principle by placing sensors at different levels in one monitoring place.

Diverse. This principle imposes a lot of difficulties for an attacker, by knowing the detection scheme at one place the attacker gains limited or no information about the detection mechanisms at other systems.

Disposable. This can be seen as a sub principle of the distributed principle, it states that the IDS

should not depend on a single component. That is, a component should be easy to replace with other components.

There are several different techniques used in AISs which have been presented over the years, most of the earlier works can be classified into three different categories:

- Conventional immune system inspired algorithms
- Negative selection
- Danger Theory

In the next section the danger theory will be explored, no other AIS category is explored since they are not essential for further understanding for this paper.

The Danger Theory

Aickelin & Greensmith (2007) use an approach which mimics the dendritic cells in the immune system, their approach is based upon the Danger theory in immunology which states that the immune system is not activated when a non-self entity is detected but when danger or damage is detected. Dendritic cells (DCs) acts as antigen-presenters, they are affected by signals in their surroundings which causes them to mature. When a DC has collected a sufficient amount of input signals it will presents its collected amount of antigens and together with the state of the DC it will be determined if it present a threat to the body or not [9].

Aickelin & Greensmith (2007) presents their theory about using the Danger Theory in intrusion detection by two algorithms; the Dendritic cell algorithm (DCA) and the TLR algorithm. Both of these make use of DCs as antigen presenters whereas the TLR algorithm also uses T-cells to match presented antigen [9].

Powers & He (2008) also describes the use of detector-antigen matching, which is analogue to the use of DCs and T-cells in immunology but their implementations in the computer world might differ [10].

3 Constructing an IDS for a networked embedded system

This section is devoted to the "construction" of an IDS for a networked embedded system. Firstly the problem will be defined as specific as possible together with the target on which the IDS is constructed for. Secondly IDSs will be "constructed" from several different approaches such as, network-based, host-based, anomaly-based, attack vs non-attack and so on. In section 3.3 the different suggested approaches in section 3.2 will be compared and a design for the IDS will be proposed.

3.1 Definition of problem

The aim for this paper is to provide a possible solution for an IDS in a networked embedded system, This section will specify the problem and the target for the IDS as detailed as possible.

3.1.1 The Networked Embedded System

The networked embedded system of choice is an automobile, there are several reasons for this choice. An automobile is a good representative for the computer systems which many humans interact with daily but never think about it much. It is also an example of a large system where the computer system inside it is responsible for a lot of the functionality and is liable for a major part of the total cost [21].

The security in these systems today is not as good as it should be. There exists possibilities to subvert/modify the systems behaviour in such a way that it would not be safe to use it anymore [21]. This is a very good example where an IDS would be useful to detect malicious tampering with the system and alert the user of the system that is everything is not alright. To get a grip of some of the embedded systems that can exist in an automobile, see figure 2.

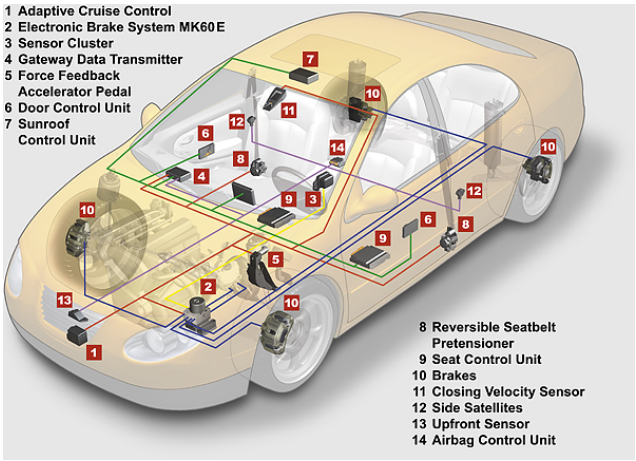

Figure 2: Example of an automobile with embedded computer systems [27].

There exists several different bus types inside an automotive system, table 1 gives some examples of bus types and a bus representative for each type.

The first three representatives LIN (Local Interconnect Network), CAN (Controller Area Network) and FlexRay handles car functionality such as door locking, power windows, real-time communication between controllers, engine management etc. FlexRay guarantees transmission times for controller information and is as such used in more safety-critical networks [21]. Thus, these networks should not really have to much connection with e.g. MOST (Media Oriented System Transport) buses that handles in-car multimedia.

Bus type	Bus representative	Speed
Subbus	LIN	20kbit/s
Event-triggered	CAN	1 Mbit/s
Time-triggered	FlexRay	10 Mbit/s
Multimedia	MOST	24 Mbit/s
Wireless	Bluetooth	720 kbit/s

Table 1: Bus types and representatives with operating speed for each bus.

3.1.2 General Assumptions

This section aims to make some assumptions that are going to be used during the construction of the IDS in the following section.

Firstly The systems behaviour is well-defined and uses well-defined ways of communication between components.

Secondly It is possible to have a training period of at least parts of the total system where there is a guarantee that it is only the intended data that comes in contact with the IDS.

3.1.3 Specific problems

There are some attacks for which the target is known, some attacks for each target are listed below [12]. These two targets, communication protocol and computation node, corresponds well to the view of HIDS and NIDS and it can be seen there is a need for both concepts in a systems where both types of attack targets are present.

Communication protocol

- Bogus synchronization/forged messages.
- Eavesdropping/Channel hijack.

Computation node

- Worms and viruses
- Buffer overflow
- Heap overflow

3.1.4 Aims of the construction

Because of the nature of the system that the IDS is designed for there are some goals for the system that needs to be thought of during the design process.

- Real-time detection
- Low resource consumption
- Continous data processing

These properties are some of those stated by Axelsson (2000) in his survey and taxonomy of intrusion detection systems at that time [20]. In an automotive embedded system the aim is primarily real-time detection because threats during run-time should be taken care of immediatly. Because the target is embedded system components low resource consumption should be aimed. Continous data processing by the IDS is desirable because the system probably do not have a sense of when it is important to look for intrusions i.e. a sharp turn or heavily trafficed roads.

3.2 Construction

This section is the main section of this paper, it aims to see if and in that case how an IDS technique can be applied to a networked embedded system, more specifically an automotive embedded system. Previously known techniques will be considered and tries to manipulate and combine them in the following sections will be made.

3.2.1 Signature-based

Signature-based IDSs have the drawback that they need resources for the knowledge database where all the rules are stored. It should be intuitive that the amount of rules can be vast [25] when the system gets more complex and since there are several different buses and ways of communication in an automotive system the probability that the amount of rules exceeds the available storage is rather high. The CPU usage should also be kept low because the IDS should not be the primary purpose of the components. Paulauskas Skudutis (2008) did a performance test on the popular NIDS *Snort* which showed that a PentiumIII with 450MHz CPU had over 30 % CPU usage already at 10 Mbps

traffic rate [26]. These reasons makes pure signature-based systems very undesirable in an embedded system so they will not be considered a valid option for this kind of IDS.

3.2.2 Attack vs non-attack

This approach is probably the most intuitive approach, there is a need to be able to detect attacks among sets of data containing normal data and attack data.

Decision Tree

This technique could be used without regard if it is a host-based or network-based IDS. To be able to use the DT at all there is a need to cluster the data (divide the data into classes) using some clustering algorithm. In Xiang et al. (2007) Bayesian clustering is used, it is an unsupervised classification method which divides the data into its "natural" classes [7]. This can of course be done by using any other clustering algorithm, but that is beyond the scope of this paper. In this case there will be access to some clean training data, i.e. data of normal usage not containing any attack data at all. Using a clustering method on that data will generate classes describing normal behaviour in different situations. A DT algorithm is then applied to the classes generated by the clustering algorithm and creates a DT which distinguishes between the different classes by looking at the used parameters.

Using the created DT, normal and anomalous behaviour can easily be distinguished. If the data that is examined follows the branches of the DT down to the leaf where it ends up, the data is most probably harmless for the system. But if the data does not conform to any of the branches that previously was established something is wrong.

Traversing branches in a tree is efficient and fast and it is easy to classify the data as normal or anomalous, this is an advantage when dealing with embedded systems. A disadvantage is that a DT might not be able to classify the anomaly properly, classification will be discussed in section 3.2.3. A problem is that there is a need of a good unsupervised clustering algorithm which can partition data in a good way for the IDS to use.

Control-Flow Checking

Control-Flow Checking (CFC) can be seen as a variant of DT's. This method is used for real-time evaluation of executing programs to see if they behave as intended. The basic idea of CFC is that a program consists of *basic blocks* which is branch-free code, i.e. code that does not contain any operations that changes the control flow by jumping several lines or calling other functions. Each basic block can contain a branch instruction as its last instruction which moves the program from one basic block to another, these transitions between basic blocks are evaluated. Each program is represented by a graph which contains basic blocks and branches that represents legitimate transitions between basic blocks [13, 14]. To connect to the earlier stated problems, e.g. buffer overflows is a typical attack where the program is tricked into making (mostly) illegal jumps.

Artificial Immune System

When creating an IDS for the networked embedded system there are several aspects to consider, first and foremost there is the resources. An embedded system component does not necessarily have a lot of CPU or storage available since it is built for a specific purpose, this corresponds well to the Lightweight property of AISs. Secondly, since this work is done witht a networked embedded system with several specialized components as a target there exists a high degree of Distributability and Diversity inside our system which also relates very well to the main properties for an AIS.

So if there were a working AIS with these properties it might have been a valid solution to the problems stated above. Today there exists several AISs created from different approaches of the human immune system [9, 10, 15] and some of them has been tested on the "usual" KDD 1999 cup data set to see how they work compared to other NIDSs [10].

This is an area where a lot of research remains to be done but if someone can make an AIS that satisfies most of the properties mentioned earlier in this paper there will be a great interest to use AISs for IDSs.

Statistical anomaly detection

Statistical anomaly detection is most often mentioned in the case of NIDS, used to create profiles of network usage. Typical metrics could be traffic rate, packets per protocol, number of IP addresses connected etc [19]. Statistical anomaly detection works by first creating a profile of usage on training data, which in this case can be done. Then the systemcan then duplicate this profile and update the second version over time with the new data that is inputted to it during runtime. This kind of systems also has the advantage that they do not need previous knowledge about attacks to detect them, and they can also evolve over time. The drawback with this kind of methods is that they can be

trained i.e. evolved by the attacker to accept behaviour that should not be accepted from the beginning [19].
One of the biggest drawbacks however, is the high rate of false positives which is still a challenge in the IDS area [19].

3.2.3 Classification of anomalies

It can be of equal importance to be able to classify the detected anomalies as it is to actually detect them, this section will be devoted to discuss some techniques used for classifying attacks.

Self Organising Map

Powers He (2008) [10] uses a Self Organising Map (SOM) to map connection vectors onto one of the clusters that describes different types of attacks. These clusters which defines attack type are created by unsupervised learning on a data set consisting only of different attacks. A SOM can be seen as a mechanism that takes points from an input space and maps them to coordinates in an output space [18, 10]. This output space can be partitioned into several different classes of attacks which in turn makes the SOM into an attack classifier.

To be able to use a SOM for this purpose it needs a training period where it is trained with attack data and then the different clusters in the SOM are labeled with an attack class. Intuitively the problem with this is that it is impossible to know every type of attack when conducting the training of the SOM.

Data Mining-based

Xiang et al. (2007) makes use of bayesian clustering and decision trees to be able to differentiate between different classes of attacks. The bayesian clustering algorithm clusters attribute vectors by looking at what class the vector should be assigned to and then looking at the probability density function for the class on the condition that the attribute vector is already assigned to that class. This approach might be very effective, but unfortunately the classification only works well for known attacks [7] .

Despite that the classification only works well for known attacks i.e. labeled data I think that data mining can be useful to detect the natural classes of the attacks. Even though they might not be intuitive for a human they can still provide information about the characteristics of attacks.

3.2.4 Host-based

According to the assumptions there exists a model of normal use even for individual components of the system (at least in a part of the total system, most probably in the more important parts that should be protected from input from e.g. MOST networks). That is there is a possibility to use anomaly detection schemes for hosts as in any experimental setting [2, 11]. The advantage here to use anomaly- and host-based IDSs is that an automotive system is a controlled environment and there is a knowledge of what should happen in each component for every step.

This approach allows the IDS to supervise the processing inside the host. There is still a need to sanity check every variable from the network, i.e. there is a for need an IDS looking at incoming traffic to determine if it is within the bounds specified.

As mentioned earlier in Section 3.2.2 this can be solved by using CFC which is very well suited for smaller, purpose specific components.

The advantages here being that it is possible to run an IDS on the individual components looking for anomalous sequences of system calls which is not that hard with a good model to derive normal behavior from. The disadvantages being that it is hard to know where it went wrong, it is only assumed that the network would behave in a well-defined way. That does not mean that the values coming in to the host is always being the correct values.

3.2.5 Network-based

The previous section about a host-based approach leads to the intuitive step to provide a discussion about network-based IDSs. In the previous section it was stated that pure host-based IDSs can not provide intrusion detection in the sense that it would be needed to know from where the threat emerged. Was it because of a transient fault? Or was it because another component has been compromised? Maybe it exists a bug within the network allowing components to address important components as they absolutely should not be able to do? These questions can be answered by using a network-based IDS.

In a normal computer network there is a great variety of protocols used which makes it very difficult to create an IDS to cope with all of them, which in the long term makes the computer networks vulnerable to different kinds of attacks. However, in a networked embedded system like in an automotive system there is not the

need for a great variety of protocol for communication. Taking it a step further, there should be explicitly stated rules of who can talk to who in such a system to avoid unnecessary resources being used and opening up the system for more attacks.

The less static properties of network communication also need to be considered. Traffic rate and unusual communication patterns are examples of attributes that can show anomalous behavior and can not be detected by using only rules. Hence there is need for anomaly-based intrusion detection.

The NIDS can introduce a need for a more centralized analyzing unit that checks all network traffic, this is because it can be very expensive to let every single component waste resources on checking all inbound-/outbound traffic [26]. A centralized unit is a disadvantage because it is more vulnerable if an attacker learns about the location the IDS has in the network and can deploy techniques to avoid it [4].

There exist several anomaly detection schemes for computer networks that can be considered when designing an IDS for networked embedded systems. Some of the techniques mentioned earlier such as DTs, statistical methods, clustering etc. can be used to solve the problem of illegitimate network traffic in a networked embedded system.

3.3 Summary

Starting of with the host-based approach it can be seen that the approach available today with CFC is well suited for detecting errors or attacks with the drawback that it can not by itself determine what is really happening. To be able to determine if there is an attack underway or if the system is exposed to transient faults there is a need to complement the CFC with a classification system. Classification without previous knowledge about attack is hard, probably one of the hardest tasks when creating an IDS. Using a data mining approach on the clean data and then compare anomalous attribute vectors with the pre-established model can probably be a way to solve this problem. Looking at what parameters giving the anomalous behaviour and using a domain experts knowledge classification can be made.

Continuing onto the network-based part, the easiest solution would probably be a hybrid system of statistical anomaly detection to take care of traffic rates etc and a signatures to take care of sanity checking of the data values that are sent over the network. However, there is a the risk of high false positives when using statistical methods if not used carefully so this part might need some more thorough research to come up with the optimal solution for this particular problem. If however recovery from anomalies can be extended to be able to cope with some false positives, the rate can be allowed to be a little higher.

What would have been the most optimal is an IDS that combines NIDS and HIDS into one singular system. In the existing literature there doess not exist a system that today successfully combines these two concepts into one system.

4 Conclusion

To conclude this review of IDS techniques and construction of an IDS for networked embedded systems the aims of the system is restated here because they have had a big influence on the design process.

- Real-time detection
- Low resource consumption
- Continous data processing

From the summary in section 3.3 it can be seen that techniques used in todays IDSs can be used and adapted to form an IDS for a networked embedded system but that there still exists a lot of work that can be done. What would be really nice to see is a system deployed at a single host that can take care of the tasks of both NIDS and HIDS and combine the systems and look for correlations between events in a better way than what is done today.

Another interesting approach is the AISs which is probably going to solve some of the problems of todays IDSs. They offer a good potential of being flexible, lightweight, diverse and disposable that no other IDS today does. The results of Aickelin & Greensmith's work with the danger theory and the DC algorithm [9] will be an interesting read.

This paper has not covered all techniques there are out there, and there could be other IDS techniques that can be adapted to embedded systems.

References

[1] Bace Rebecca, *Intrusion Detection.* Macmillan Technical Publishing, Indianapolis, USA, 2000.

[2] Tandon G., Chan P. K., *On the learning of system call attributes for host-based anomaly detection.* In International Journal of Artificial Intelligence Tools, pages 875-892, 2006.

[3] Tan X., Xi H., *Hidden semi-Markov model for anomaly detection.* Applied Mathematics and Computation 205, pages 562-567, 2008.

[4] Ptacek T. H., Newsham T. N, *Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection.* Secure Networks, Inc., 1998.

[5] Basile C., Gupta M., Kalbarczyk Z., Iyer R. K., *An Approach for Detecting and Distiguishing Errors versus Attacks in Sensor Networks* IEEE Computer Society, 2006.

[6] Barr Micheal, *Embedded Systems Glossary.* Netrino Technical Library, http://www.netrino.com/Embedded-Systems/Glossary-E, Retrieved 2009-02-20.

[7] Cheng Xiang, Png Chin Yong, Lim Swee Meng, *Design of multiple-level hybrid classifier for intrusion detection system using Bayesian clustering and decision trees.* Pattern Recognition Letters 29, pages 918-924, 2008.

[8] Jiawei Han, Micheline Kamber, *Data Mining, Concepts and Techniques, Second ed..* Morgan Kaufmann Publishers, San Francisco, USA, 2006.

[9] Uwe Aickelin, Julie Greensmith, *Sensing danger: Innate immunology for intrusion detection* Information Security Technical Report 12, pages 218-227, 2007.

[10] Simon T. Powers, Jun He, *A hybrid artificial immune system and Self Organising Map for network intrusion detection.* Information Sciences 178, pages 3024-3042, 2008.

[11] Abimbola A.A., Munoz J.M., Buchanan W.J., *NetHost-sensor: Monitoring a target host's application via system calls.* Information Security Technical Report 11, pages 166-175, 2006.

[12] Izosimov V., *Introduction to Our Work on Fault Tolerance Towards Embedded Security.* Linköping University, 2009.

[13] Goloubeva O., Rebaudengo M., Sonza Reorda M., Violante M., *Soft-error Detection Using Control Flow Assertions.* Proceedings of the 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2003.

[14] Oh N., Shirvani P. P., McCluskey E. J., *Control-Flow Checking By Software Signatures.* IEEE Transactions of Reliability, Vol. 51, No.2. March, 2002.

[15] Kim J., Bentley J. B., Aickelin U., Greensmith J., Tedesco G., Twycross J., *Immune system approaches to intrusion detection - a review.* Natural Computing 6, pages 413-466, 2007.

[16] Somayaji A., Hofmeyr S., Forrest S., *Principles of a Computer Immune System.* New Security Paradigms Workshop Langdale, Cumbria UK, 1997.

[17] Kim J., Bentley P., *The Human Immune System and Network Intrusion Detection.* 7th European Conference on Intelligent Techniques and Soft Computing, 2001.

[18] Kohonen T., *Self-organization and Associative Memory, 2nd Ed..* Springer-Verlag, 1988.

[19] Garcìa-Teodoro P., Díaz-Verdejo J., Macià-Fernàndez G., Vàzquez E., *Anomaly-based network intrusion detection: Techniques, systems and challenges.* Computers & Security 28, pages 18-28, 2009.

[20] Axelsson S., *Intrusion Detection Systems: A Survey and Taxonomy.* Chalmers University of Technology, Gteborg, Sweden, 2000.

[21] Lemke K., Paar C., Wolf M., *Embedded Security in Cars.* Springer-Verlag, Berlin Heidelberg, 2006.

[22] Porras P., Saidi H., Yegneswaran V., *Conficker C P2P Snort Detection Module.* [www] http://mtc.sri.com /Conficker/contrib/Conficker_C_P2P_Detector.patch, SRI International, 2009.

[23] Hassmund J, *Studying IDS signatures using botnet infected honeypots.* ADIT, LiU, 2009.

[24] [www] http://www.milw0rm.com/, fetched April 2009.

[25] [www] http://www.activeworx.org, fetched May 2009.

[26] Paulauskas N., Skudutis J, *Investigation of the Intrusion Detection System "Snort" Performance.* Electronics and Electrical Engineering, 2008.

[27] N. Eyal Co. Vehicle Lab [www] http://www.vehicle-lab.net/services.html fetched May 2009.