# Airids Architecture

# And

# Methodology

Rev 1.0

Prepared by
Thomas Munn

The Goal:

To make an Open-Sourced IDS that can intelligently react to threats without causing denial of service conditions, and reduce the workload of IDS analysts so they can concentrate on less mundane threats.

The Problem:

Current IDS implementations lack one critical ability: The ability to react intelligently. They are very happy to warble, chirp and scream that there has been an intruder, but they don t DO anything, other than just annoy the Jailer, er, security person. Worse than this, with signature based IDS, there are many false alarms. Error rates of upto 60% have been seen by this analyst. Current active methods of altering firewall rulesets, session sniping and the like are just too primitive to be trusted. All an intruder has to do is send a barrage that looks like it came from your DNS server, and you are in a far worse situation than if you were just watching. In addition to this, session sniping doesn t work very well with pesky ICMP, IGMP and UDP.

Network based NIDS are the second problem: Alone they cannot see what is going on behind an encrypted tunnel. They are subject do dropping packets on high-speed links. They are subject to being blinded , faked out, and just too annoying so that the become ignored. Signature maintenance is a nightmare in that tuning the IDS requires countless hours of finding out what signatures are stupid and need to be terminated, and which ones are good and need to be kept.

Host based IDSES also have their shortcomings:

Can t see network based attacks
Limited to only one host
Can affect operation of busy servers
Managing multiple hosts rapidly becomes a problem
Must be installed on each host
Must be updated regularly
Must be baselined.

Also current IDS implementations require extraordinary ability and expertise of the people who are going to be deploying it. Many people deploying IDSes have no idea how to properly implement and maintain rulesets place sensors, etc.

No ids currently takes a correlative approach out of the box. They don t correlate what other sensors are saying and provide for a simple, intuitive

display as to whether or not you are in fact in any real danger. There are add on products that do this, Motorola intrusion vision, for example, but this product costs nearly $10,000.

This brings me to my greatest gripe: I shouldn t have to pay $10,000 for an Ids that works. I should be able to deploy sensors for the cost of the hardware alone, so that I can spend valuable money on important things like personnel training.

The Solution:

The Automated Intelligently Reactive Intrusion Detection System (AIRIDS henceforth) hopes to address some of these shortcomings. In short, it seeks to provide for a system where a user can rate threats, and have the machine respond based on these ratings. It will use the principles of rule-based AI to select which signatures and data sources are used in making decisions. It also will use open-sourced components (with the ability to tack in commercial ones, should they output in text or a reasonably understandable format) so as to minimize work on development. It will not seek to re-invent the wheel, or to limit itself to one particular network-based or host based IDS. It will be o/s agnostic, at least from the user s perspective, and will have the ability to be packaged in a nifty ISO, one for each component of the architecture. It will have permanent memory in the shape of an RDBMS, and short term memory in the form of fast response mechanisms. It will be extensible, well documented, and an ongoing work. It will employ flexible response mechanisms such as bridging, command routing, and firewall ruleset changes to actively respond. It will employ a flexible licensing model: E.G. Each component will be separate, and will have the ability to be under its own license model. So the core engine could be one license, and the response module could be another. To avoid gpl virus like bleeding to other components, no source code will be shared among the different modules to prevent conflicts between the different module portions of AIRIDS.

A word on licensing

The last sentence will undoubtedly make some people very angry. Why not release this project under ONE license. Well, I originally thought that this would be a good idea, but then I thought, what if I want to make a living off of this project? What if others do, do I have the right to FORCE a project to be all free ? Do I have the right to restrict the creative expression of others? Do I even want to get into the silly bsd vs gpl arguments that always seem to plague other projects? No. I want to invent a new term: melting-pot licensing. Just like America, which has dozens of nationalities living in relative harmony, so should differing licenses live together in harmony. Each module will be a

neighborhood in that it must be under ONE license. Separate but equal will be our motto. Only things that leave neighborhoods (e.g. people) will communicate in common areas like downtown. The Common module, e.g. the inference engine will be released under the BSD license. This is to allow for flexibility and the ability for the work to go mainstream and still remain in the public domain. The flexibility of the Melting-pot license will allow different projects and philosophies to co-exist in relative harmony. If you don t like my neighborhood, just go to the next one with more agreeable occupants. AIRIDS might be seen as a large city, with constituent modules representing various Structures that support the city.

Overview of AIRIDS

Here is the current secret formula for the AIRIDS city:

Physical Components:

Sensor/reaction module
Analysis station
Firewalls
Hosts themselves
Data sources (such as syslog server)

Modules:

Courts (Intelligence, storage)
Police (reactors/sensors)
The Press (communicating state of system to user)
Post Office (for official correspondence between user and courts)

Courts (Inference Engine)

These are the heart of the AIRIDS architecture. They are where packets, and events are taken, tried, and either acquitted or executed. The laws of the system are determined by a ranking based on user responses to a questionnaire that is sent out on a daily basis until training is complete.

The questionnaire would be an easy to read summary with events correlated and counted so that the citizen could respond with 1- Never show me this alert again to 9 this alert is very meaningful and takes precedence over other alerts. These responses could, of course be changed at any time.

In addition to this, there will be a fact finding portion of the court. It will go out an probe a network for devices, machines, etc. with their OS types, ports that are open, and banner scans of all web servers, etc. Once this fact finding phase was complete, the user could label machines in terms of criticality, e.g. never block traffic from our dns server, just send an alert or log it. Of course, it would be much simpler to have proper rules to prevent such things as spoofing (get rid of packets pretending to come from outside that have ip address ranges that the site owns). The fact finding committee would interact with the user (citizen), finding out what ip address ranges belong to partners, which ones were trusted and which ones should NEVER be blocked, except under extreme circumstances. The fact finding commission would be able to tailor rulesets and eliminate signatures, or host-based information that do not apply to the current situation. This reduces the number of things we have to watch, and bother the citizen with.

The questionnaire is analogous to common law wherein a court determines acceptable outcomes based on past experiences and decision of judges . The weight and scope of this body of common law can change with time, and new decisions can take precedence over past decisions, or even remove past decisions entirely. The key here is context. The common law (hereafter known as knowledge base) base shouldn t change if it is working, but if it isn t there should be a facility to alter past decisions to represent current threat models. Above all, a judge (citizen) should have the right to determine the laws for his or her city.

Included would be a starting knowledge base, that a judge (analyst) could implement and then use if they found it satisfactory.

Police (Sensors and Reaction modules)

The police are the enforcers of the courts system. They escort defendants (packets) to either the gas chamber, or glorious freedom to roam the network. They look for suspicious packets, have regular beats in the form of being on the host, watching for suspicious activity. When such activity is spotted, it is brought to the attention of the court, which makes an instantaneous decision based on common law as to the fate of the event, as well as the punishment. Like police, the sensors and reaction modules run into two types of situations: those such as fifi being stuck in a tree, that can merit a more measured and slow response, and those such as being shot at by thugs, or seeing a murder in progress. These require instant decisions, so each officer has a copy of the common law with them, so they can independently act from the court when necessary. In addition to this, police have radios so that they can talk to each other and tell details about a suspect, his or her direction, type of car, etc. and correlate information to act more effectively as a team.

The Press (The presentation layer to user)

In any free society, or even in a totalitarian one, the press serves the important function of keeping its citizens informed of news (either created or real). News must be concise, make sense of diverse events, and must be understandable by a fourth grader. It must also accurately reflect what it is reporting.

The presentation layer in the AIRIDS project is one that fulfills these requirements. Basically, events are correlated and sent to a bullseye as events are correlated, tallied, and reported, the bullseye gets shot either closer or farther away from the center. Hosts that are most important, are near the center, and have a higher weight than unimportant hosts which will show up on the periphery. The events represent aggregate number of events, so that greater frequency of events will have larger holes, and smaller number of events will have smaller holes. It will have drill down detail available by clicking on a dot, and then a summary of events will be displayed for the host in question. (This could also be a network). More detail is possible by clicking further and further down, until the level of individual events is reached. If an attack is stopped the hole goes away (e.g. if the attack has been neutralized by the police force, then the threat is removed from the display.), however, the record of the account is stored.

The news also has other communication methods, e.g. radio, pager, and cell phone. Using talking technology the IDS should be able to tell you what is happening in an audible format. This allows cell phones to interact with the system. It would only call one number, and would ask for a pin (5 digits minimum), and then deliver the news. It would also have response options, e.g. could say that it has a condition that it doesn t know what to do with, read the data off, and then you can say 1 for kill the connection 2 log the connection 3 the connection is O.K etc. It would allow for remote operation anywhere in the world there is wireless phone access.

So a citizen could respond to threats WITHOUT having to be at the console.

Finally the press would use have a research mode, in that all old news would be archived and searchable for data mining, and finding out who is being naughty and nice.

The press also serves the function of interviewing the user. It could ask a user how he or she feels about a news item (aka event). The summarization of the days news could be polled by the user, thusly help building the common law that was talked about earlier in this paper.

Post Office Module (communications architecture)

Finally, the Post office is for sending all correspondence between various parties. The polls that are carried out via the news, the communications between the sensors, the stamping of messages with cryptographic hashes to protect them from tampering, etc occurs here. The court in a sense is the co-ordinating body that communicates out of band via a covert communications channel. All the modules involved in the city talk through this cb like except that no one can listen in.

Modules communicate in real-time about threats, network outages (if an officer gets shot for example), citizens being attacked (hosts), etc.

Technical Considerations for the AIRIDS model

The paper presented here has tried to conceptualize how the AIRIDS model will work. What follows is a more technical discussion of its operation and technical problems faced by any IDS.

Problem of too much bad data

This is perhaps the worst problem facing any system which hopes to automate response to attack. Signatures are simply too unreliable in and of themselves to trust an automated system to. Just ask any user of such features as realsecure s session sniping , and have the CEO s computer be blocked because the IDS thinks he is an intruder.

There are several ways around this problem:

1. Correlate data between different segments of the network. By correlation, this means sensor 1 and sensor2 compare their signature outputs. By intelligently comparing the differences, or even the false positives, we can use the data about the data to tell what is REALLY going on. Obviously, the user probably couldn t do this, but an analyst could.
2. Apply statistical analysis for signature probabilities--E.G. for any given packet, which signatures are the most probable to be correct (e.g. any signatures related to IIS shouldn t show up on a UNIX box)? I also read an interesting article on using batesian mathematics for some of this.

3. Use more than network based IDS. Windows, UNIX, routers, SNMP, syslog, tripwire, firewalls, etc. These host, system, and network based protocols and programs provide a wealth of data to help us. For instance, if a windows box sees an invalid login attempt, we can weight the sensor to stop connections if it sees more than say 4 per second from a given host, or could even, temporarily, block the host (provided that it is not a partner) for a given time period, or even re-route the host to another web server for honeynet functionality. Tripwire signatures are another example. If, say an INDEX.HTML file is changed without proper md5summing, things could be rooted and appropriate action taken.
4. Anomaly detection--Things that vary from the norm. ICMP ranges out of the norm , high traffic volumes (be careful!), large number of errors e.g. http 404 errors , people trying to login to servers that shouldn t be logged into, strange userids, etc.

The problem of Encryption

Any network based IDS requires the ability to inspect packets to compare them to a signature database. If packets are encrypted, they cannot be inspected. As the Internet goes IPV6, we are going to see more and more end-to-end encryption, so this is only going to get worse. VPN traffic, SSL, SSH, etc. all are very difficult problems to go around, by the very nature of their encryption.

Some ways around the encryption problem:

Man in the middle -- for TLS aware applications we can have an IDS serve as a proxy and serve encrypted streams in both directions. We can inspect packets in the middle of the encryption process, and encrypt them again once we see they aren t nasty. The same could be done for SSH with some source code editing.

Put a sensor at the end of a tunnel-- For VPNS this is the only option. The sensor looks at the unencrypted tunnel.

Put the sensor on the VPN device itself--This allows you to see unencrypted traffic

Put the sensor on a remote desktop or host-- You can see all data coming from the host, and appropriately react to it, obviously host must be in your control, this may be the only option as nic to nic encryption becomes standard.

The problem of defining normal

In any IDS system, defining what is, and is not, hostile is quite a difficult issue. Network spikes, backups, maintenance or changes to hosts, etc. all present a very complex set of conditions to define.

Some methods around the normal problem:

1. Have a learning mode, one that looks at network traffic for some time and models based upon this. The only problem is that if you are hacked during the training period .hacking becomes normal!
2. Use intelligence and change-control. By keeping your maintenance scheduled, the ids can be tuned not to squawk during scheduled update times or routine system maintenance. By forcing humans who maintain systems to be predictable, you can make your ids more effective.
3. Look at function of host, and what it does. This allows us to say for example that a web server should NEVER have port 31337 open. It also allows us to

say things like PDC s in our dmz (god help you!) shouldn t be getting anonymous registry manipulation requests from outside ip addresses.

The problem of people

Any technology or detection methodology is only as good as the people who implement it, and will be evaded by people who are smarter than you. The problem with IDS is that it is trying to predict what people will do. People are inherently unpredictable, especially on computers. While you can see trends predicting what an intruder will do to your hosts can be a very difficult science. Certainly one can look at the standard model of intrusion, reconnaissance, target selection, attack, and exploit. But things are more complex than this.

Most attacks don t come from some evil hacker outside but by insiders. By people we trust. Attacks could originate from a trusted host on the inside of our network.

An IDS that helps educate its GENERAL population will enable us to get around this problem. By seeing the kinds of attacks going on, a general user population can be EDUCATED about what a threat constitutes, and can TELL US when they see such activity. Kind of like when the burglar alarm draws attention to a car thief (although too much of this makes you ignore such things!) People themselves can become sensors. When they see people acting suspiciously, or plugging into network jacks that they shouldn t. People definitely have to be part of the solution, not part of the problem.