Trend Center
Final Report

31 Oct 2003

Todd Heberlein
Net Squared, Inc.

Abstract

This report summarizes the work performed under the TrendCenter SBIR contract. We begin by summarizing the change in the threat environment that threatens the traditional view of intrusion detection, and then we provide some simple but powerful statistics to argue that the traditional "detect and respond" model needs to be replaced with a "predict and prepare" model. We introduce the original vision for TrendCenter, how that vision changed, and why. We provide a tour of the TrendCenter portal, identify difficult technical issues that still need to be addressed, and provide a "to do" list for future work. Finally we provide the documentation for the two tools most system administrators will use to interact with TrendCenter: one tool to sanitize and summarize sensor logs to be submitted to the TrendCenter portal, and one tool for system administrators to configure a vulnerability scanner to look for specific vulnerabilities in their network that attacks will most likely exploit over the next few days.

Table of Contents

List of Figures

1 Introduction

When intrusion detection systems were initially funded in the mid-1980s, the environment in which the systems were expected to operate was very different than the today's typical environment. The following few paragraphs summarize the threat environment back then.

Actual attacks against the information systems were assumed to be rare. Indeed, the approach used by most systems was to look for rare or unexpected events, and the term "anomaly detection" was synonymous with "intrusion detection".

The numbers of known vulnerabilities at any given time were extremely rare, and new ones emerged very slowly. In the 2-3 year period of an intrusion detection research project, the vulnerabilities at the beginning of the project were generally the same as the vulnerabilities at the end of the project: the same default passwords (i.e., backdoors) to allow the vendors into the machines, the '+' in a SunOS hosts.equiv file, easily cracked passwords, and a small number of vulnerabilities in SUID programs such as gcore and rdist.

Few important systems were connected to the Internet. Statistics were always thrown about in papers and presentations stating that *insiders* were the source of the overwhelming number of cases of attacks or misuse. What these papers and presentation typically failed to state was that the statistics were often gathered on machines with no outside connections. There could be no *outsiders*. The important machines were kept off open networks.

Computer systems and process interactions were much simpler. Many machines were self contained and never connected to the network. And even when machines were connected to the network, they still largely acted as stand-alone systems – they contained their own disks, login and password files, and any external systems that the computer was expected to connect with was kept in a local address file (e.g., /etc/hosts for UNIX systems – there was no wide spread DNS system available then). Most programs were designed to be run by users directly logged in to the system, and the program rarely communicated with other systems on the network.

Even when systems were connected to the Internet (or other public networks), these networks were very small by today's standards, and access to these public networks were still largely limited to scientists at universities, governments, and other technical organizations.

Finally, when attacks did occur, either by an insider or outsider, they were generally performed manually, with the attacker typing each instruction at a keyboard. The typical threat model, at least for outsiders, was Cliff Stoll's "Wiley Hacker" as described in his book *The Cuckoo's Egg* – a lone hacker patiently but persistently winding his way through computers, one system at a time [Stol 88].

Today's environment has changed dramatically. Attacks are no longer rare but are a regular and continuous threat. If you install from a CD a fresh operating system on a computer directly connected to the Internet (e.g., at a University or on a home DSL or cable modem connection without a firewall), within a day the system will probably be

successfully hacked. At the UC Davis Security Lab mean time between probes on a system is measured in minutes.

Whereas in the past new vulnerabilities were rarely discovered, today thousands of new vulnerabilities are discovered and publicly reported every year. System and network administrators simply cannot keep up with the latest potential challenges to their system's security.

In the past, while only a small number of computers were connected to the Internet and even fewer important systems were, today mission critical computer systems in most organizations are directly connected to the Internet. Power grid systems, DOD war planning systems, dams, and payroll systems are typically connected to the Internet.

Also, today's systems are much more complex and interdependent than those of the past. Login to your workstation, and your system may make a request to a DNS server to find the IP address of a Kerberos or NIS name and password server before authenticating you. Once authenticated your system may make another request to the DNS server to discover the IP address of file server, and then your computer mounts your home directory from that remote file system. To perform your work you may open a web browser and connect to a web server, and that server connects to one or more back-end databases to provide you with the information you need. This complexity and interdependence means that a vulnerability in one piece of the system can cascade through many other components in our network.

And whereas the early Internet was small and limited to technically oriented users and organizations primarily operating in the United States, today's Internet reaches into every corner of the globe touching people from almost every strata of society. A student in elementary school in Pakistan has as much of an opportunity to attack your critical systems as the employee sitting next to you or the MIT graduate student.

Finally, few attacks today are launched one at a time from commands typed into a computer. Typically attacks are bundled into automated tools that can scan thousands of systems per minute, and these tools are often simple enough to be run by amateurs, so called "script kiddies".

Thus, intrusion detection systems, originally conceived during a time when attacks were rare and slow, have become overwhelmed and are nearly useless in today's environment of continuous and fast moving attacks.

TrendCenter, originally conceived in 2000, was designed to address this mismatch between the original intrusion detection concept and today's threat environment. Instead of treating an intrusion detection sensor as a device designed to protect a site, TrendCenter uses a large number of sensors and multiple sites as a means to measure the threat environment, and then we couple the threat environment with an analysis of your site to arrive at a specific set of recommendations to secure your site *before* it is attacked.

In Section 2 we dive deeper into the threat environment, examining the issues a typical system or network administrator faces today. In Section 3 we present the TrendCenter vision, both the original concept and the final vision that changed due to a number of external factors. In Section 4 we look in detail at the TrendCenter work flow and the prototype web portal we developed. In Section 5 we discuss some of the

"gotchas" we ran into when deploying the prototype. In Section 6 we list some of the additional near-term work we think needs to be added to TrendCenter. In Section 7 we summarize our efforts on this project. Finally, there are two appendices that document the two tools system administrators will use to interact with and take advantage of the TrendCenter portal.

2 Threat Environment by the Numbers

The birth of the intrusion detection concept is usually traced back to James Anderson's 1980 paper *Computer Security Threat Monitoring and Surveillance* [Ande 80] and then in the mid-1980s with the first development effort of the Intrusion Detection Expert System (IDES) [Denn 87]. This was a time when few if any statistics were kept regarding the numbers of vulnerabilities, attacks, attack tools, etc. But in this section we have identified some statistics that illustrate the threat environment and how it has changed over time.

Figure 1 shows the number of vulnerabilities reported per year as recorded by CERT [CERT 03]. The earliest year for which they have statistics is 1995, and only 171 vulnerabilities were reported that year. By 2002 the numbers soar to 4,129 – on average, over 11 new vulnerabilities reported per day. With all their other responsibilities, few system administrators have the time to continuously track the constant stream of new reports, scan all their systems for presence of the vulnerabilities, and keep all of their systems fully patched against all known vulnerabilities.

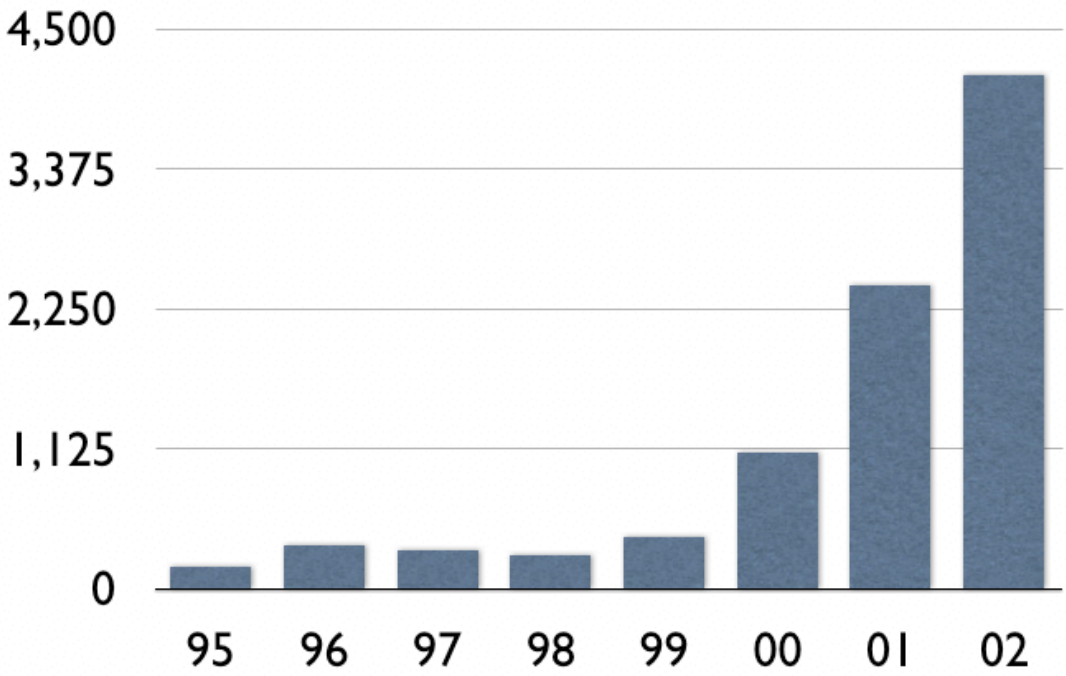


Figure 1: Reported Vulnerabilities Per Year

In addition to the tremendous growth in vulnerabilities reported each year, as shown in Figure 2 the number of attacks reported to CERT each year is also growing rapidly [CERT 03]. In the first full year of data, 1989, only 132 incidents were reported

to them, but by 2002, 82,094 incidents were reported in a single year. Furthermore, this growth rate underestimates the actual number of systems attacked because by CERT's counting standards each incident may involve hundreds our thousands of systems or even sites. And as we mentioned previously, early attacks tended to involve a small number of hosts attacked manually, so a single incident in 1989 probably involved a small number of hosts. Now, with most attacks automated, a single incident can easily involve thousands of systems. Thus, while according to CERT the number of incidents is growing rapidly, so also is the number of machines involved in each incident.

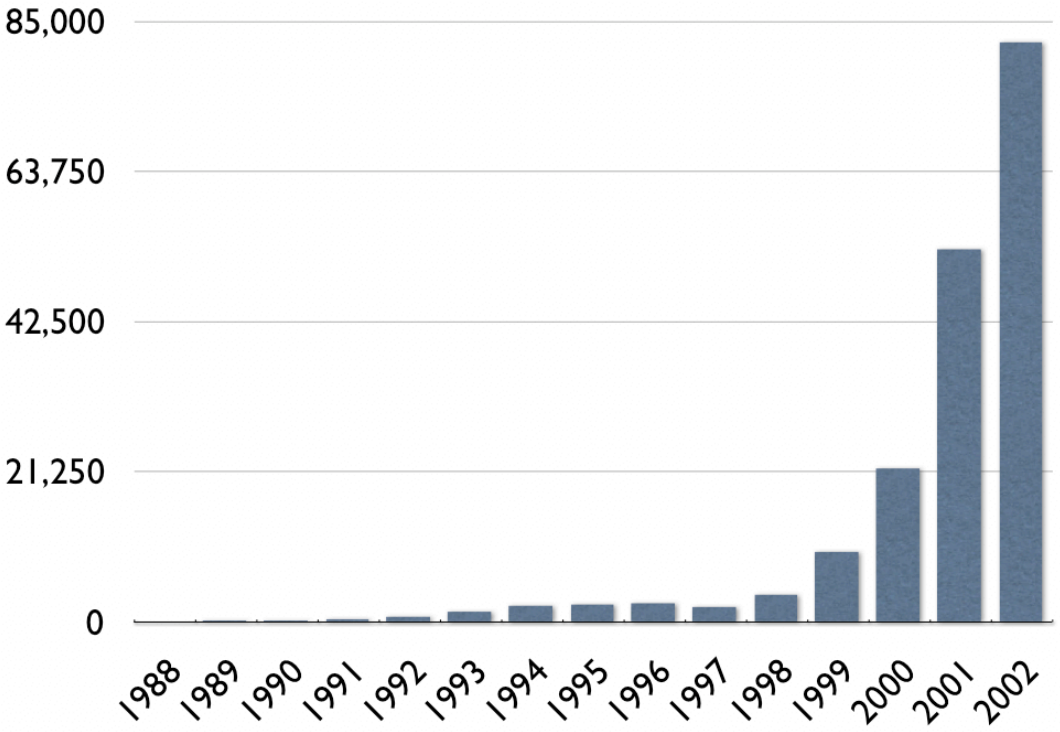


Figure 2: Growth in Number of Incidents

To put this in concrete terms for a single site, Figure 3 shows the number of alerts generated per day by a Snort 2.x sensor at UC Davis. On average, the sensor reported 801,414 attack attempts per day. This translates into 557 attack reports per minute, or almost 10 reports per second. And this is for only one site. Even more importantly, the scan detector was turned off on the sensor, and we used only a fraction of the Snort rules – the ones that specifically identified the vulnerability exploited by the attack with a CVE identifier. Since the CVE-based rules were generally "content" type rules, they only fired when a full connection was established and the data was sent to the server, so attempted connections to ports that had no server (the vast majority of the cases) would never generate an attack report. Thus in the final analysis, the 800,000+ attacks per day probably vastly undercounted the number of attempted attacks on the network.

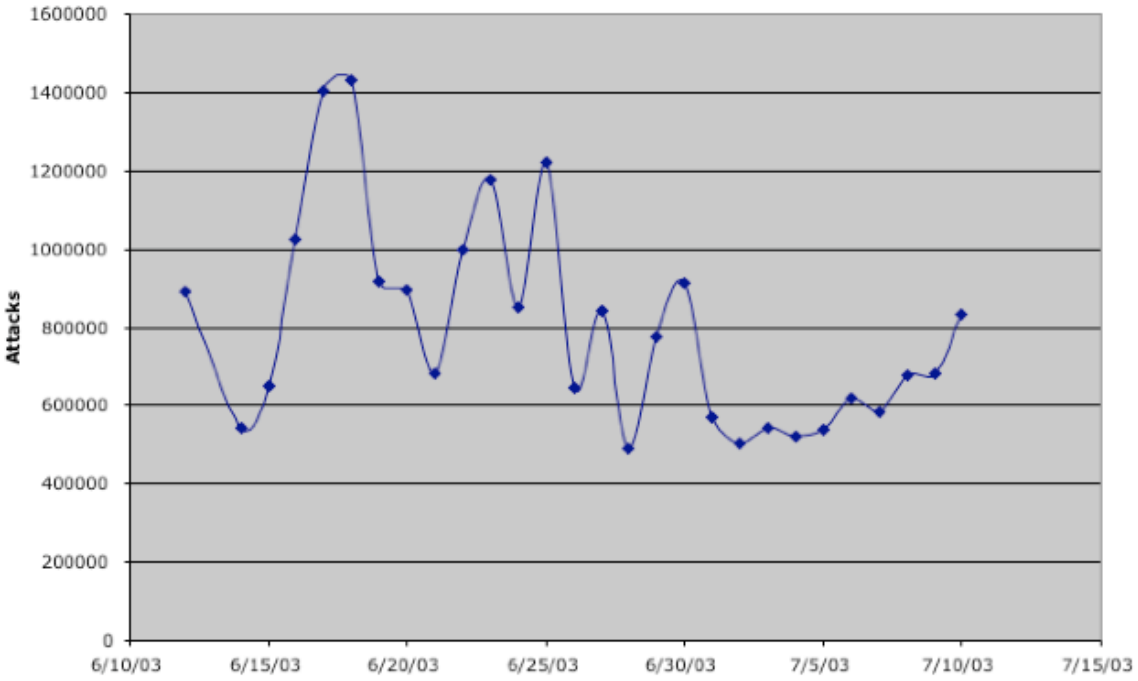


Figure 3: Attacks per Day At One Site

Finally, for a third data point we looked at a commercial monitoring service, Counterpane [Coun 03], to determine the number of sensor events they were recording. Counterpane's home page (see Figure 4) includes a small script that reports the number of events they have processed for the year[1]. As of Oct 30, 2003, over 400 billion events have been processed this year. This works out to over 17,000 events per second.

Figure 4: Clip From Counterpane's Web Site

In conclusion, the explosion in the number of new vulnerabilities discovered and reported every day and the explosion in the number of attacks as recorded by various organizations, the operating environment for intrusion detection systems today is very

[1] The actual number is an estimate. From their web documentation: "This counter estimates the total number of events collected by Counterpane, using a baseline and rate that are updated weekly from actual data."

different than the operating environment when intrusion detection systems were first conceived and developed.

3 TrendCenter Vision

In this section we look at TrendCenter vision, both as originally conceived in 2000 and how the vision evolved as external conditions changed.

3.1 Motivation

As the numbers presented in Section 2 demonstrate, the threat environment faced by system administrators was changing rapidly in 2000. We recognized that system administrators could not keep on top of the blizzard of reports they were receiving from the security community and their own sensors. In short, the *detect and respond* model originally envisioned for intrusion detection systems was dying quickly. TrendCenter proposed to replace this dying model with a *predict and prepare* model.

The "predict and prepare" model was based on two key observations. First, while the number of vulnerabilities being reported each year was in the thousands, attackers were exploiting only a small number of them. For example, in 2001 CERT identified 2,437 vulnerabilities, but less than 1% were exploited in actual attacks [Schw 02]. Second, with thousands of sites on the network, the probability that any particular site would experience the first attack exploiting a new vulnerability is very small. The TrendCenter goal was to coordinate sensors from hundreds or thousands of sites to determine which threats each specific site was likely to see over the next few days (the "predict" part), and then identify where in each site those attacks are likely to succeed so system administrators can focus their patching attention to those specific systems (the "prepare" part).

3.2 Original Models

Being a Small *Business* Innovation *Research* (SBIR) project, we had to concentrate on both a business model for the project as well as a technical model with original research. We discuss those two issues next.

The original commercial model was simple: we were going to create a web portal. In January 2000 portals were a hot topic. They attracted press attention, venture capital funding, and, if they went public, their market capitalization were incredibly high. And the portal model was simple, attract "eyeballs" and sell advertising space; a potential sideline was a subscription service for premium offerings.

We envisioned the TrendCenter portal would be the homepage of a significant portion of system administrators around the world. A system administrator would login to his workstation in the morning, launch his web browser, and the TrendCenter page would load identifying the vulnerabilities most likely to be attacked in the near future. It would be like a weather forecast for system administrators, and they would check the forecast first thing every morning.

The technical model was largely based on techniques Amazon.com used to sell books. The argument was as follows: the average visitor at Amazon.com was only going

to spend a limited time at their web site, so Amazon.com had to maximize the probability that they could present a book to the visitor that the visitor would buy. In our case, the system administrator would spend only a limited amount of time at our portal, so we had to maximize the probability that we would present to the visitor a vulnerability that would likely be attacked at their site.

Amazon.com used three ranking system to find books they think the visitor would buy: best sellers' lists, personal recommendations, and purchase circles. The best sellers' list concept was simple – if a lot of other people liked this book, you might like it as well. The personal recommendations concept analyzes your past purchases, finds a group of other customers who bought similar things, and then identifies books that a large number of people in that group also bought that your may not have. Finally, purchase circles defined certain customer bases such as Air Force customers, or customers living in the San Francisco Bay Area, and identified what was uniquely popular in those areas. For example, a book on hiking trails in the San Francisco area might be popular in the San Francisco area but would probably sell poorly in the rest of the country. Thus, if the visitor came from San Francisco, Amazon, through the purchase circle approach, would present the visitor this particular book on hiking trails.

When mapping these to TrendCenter, the general top sellers list was the attacks most people sites experienced, so your site would probably experience them as well. These attacks may only be random acts of violence, but you should be prepared for them. Our personal recommendations would be determined by identifying sites with similar victim profiles – sites that historically experienced the same type of attacks or the same attackers. If a victim with a similar profile saw a new attack, you might as well. Finally, our purchase circles would be based on specific industries (e.g., the power grid industry, the Air Force, or Department of Energy). Thus, attacks that appear uniquely popular at power facilities will be highlighted for power facility customers.

3.3 Environment Changes

Unfortunately the macro trends of increasing numbers of vulnerabilities and attacks were not the only changes in the environment during the TrendCenter development. Business environments changed, technical aspects of the threat environment changed, and some of our assumptions turned out to be incorrect, and during the course of developing TrendCenter we had to change our direction. The next few paragraphs describe some of those changes.

One of the most important changes we faced was the change in the business climate epitomized by the stock market bubble bursting. In the winter of 2000 when TrendCenter was conceived, web portals were considered an important business model for the future – portals attracted substantial press, venture capitalists poured money into them, and when a portal company's stock went public its capitalization was very high. Unfortunately, all that began to change for the negative in 2000, and web portal business conditions continued to slide for the next three years. Money for on-line advertising dropped through the floor forcing even the largest portal, Yahoo!, to drop its CEO and change its business model. As the stock markets dropped (Nasdaq dropped to less than 1/3 of its March 2000 peak), companies stopped going public, and this stopped the flow

of money into venture capitalists' pockets. This in turn curtailed new investments, especially in Internet concepts such as web portals.

In addition to the business environment changes, technical aspects of the threat environment changed. In particular, worms, virtually unseen on the Internet when TrendCenter was first conceived, became a regular occurrence on the Internet. The TrendCenter model assumed new attacks would have a relatively slow ramp-up period, so that once we detected the new threat, most system administrators could be given enough warning to patch those systems. With worms, which during the life of the project went from off the radar screen to the greatest concern of most system administrators, most systems are infected within hours and possibly minutes, so there is effectively no warning time.

SANS [Sans 03] also changed directions during the course of our project. As mentioned in our proposal, the TrendCenter concept was largely inspired by SANS's Y2K rollover center, a manual effort to collect sensor logs from around the Internet in order to spot expected attacks coinciding with the change in the millennium. However, following Y2K, instead of terminating the Y2K rollover center, SANS decided to rechristen it the Global Incident Analysis Center, or GIAC. For a variety of reasons we believed (and still do) SANS was in the best position to run a TrendCenter-type analysis center, and so we decided to team with them. People would send their sensor logs to GIAC, we would process the data and generate the recommendations, and GIAC would host the portal.

Unfortunately, as the business environment for web portals changed, SANS shifted its emphasis for GIAC away from an "incident analysis center" to an education and certification organization – an effort that is (from what we have heard) very financially successful. GIAC now stands for "Global Information Assurance Certification" [Giac 03]. The original GIAC concept was shifted to incidents.org [Inci 03], and the sensor data was collected by another organization DShield.org [Dshi 03]. Furthermore, because of technical limitation of the DShield data we could not perform the level of analysis we wanted. For example, DShield only recorded that a port number was probed, and with the case of web servers, with dozens upon dozens of known vulnerabilities, knowing someone was attacking port 80 did not provide enough detail to allow us to specify exactly which vulnerabilities were being attacked. Also, as of the summer of 2002, even though a number of people contributed to DShield's data, most data feeds represented a small number of computers, including a large number of "single IP address" feeds. For example, UC Davis, with multiple class B networks and many class C networks, had more hosts than were represented in the DShield data.

Because of GIAC's change in direction from an incident analysis center to a certification organization, DShield's limited information in the data they collected, and the relatively small size of the DShield data feed, we decided to pursue TrendCenter independently. This change in direction forced us in the summer of 2002, rather late into the project, to develop our own web portal and collect our own data.

During our change of direction we discovered a second problem, and perhaps the reason DShield's data did not represent the large organizations we had hoped for – organizations did not want to share their sensor data. For example, in the summer of

2002 the UC Davis Vice Provost of Information Technology, John Bruno, approached the lead of the UC Davis Compute Security Laboratory (SecLab), Professor Karl Levitt, and told Prof. Levitt that he would like the security lab to help perform intrusion detection work for the campus. Since we worked very closely with the UCD SecLab, we thought the timing was perfect. However, it took another 9 months of negotiations to push through a Memorandum of Understanding (MOU), which left us with only a few short months to collect data for experiments.

As we discovered, this desire not to share was not unique to UC Davis. The Department of Energy's Office of the Chief Information Officer is funding a DOE-wide effort very similar to TrendCenter called Cooperative Protection Program (CPP). Lawrence Livermore National Laboratories (LLNL) Computer Incident Advisory Capability (CIAC) organization is leading the processing of the information and identifying threat trends. However, despite the mandate and the funding coming from the Office of the CIO, many DOE sites are still refusing to provide CIAC with their data. In another government effort, the United States Air Force Information Warfare Center (AFIWC) is developing and testing standard techniques for the Air Force to perform vulnerability scanning; however, we have heard that individual Air Force sites are hesitant to share this vulnerability information with a central organization such as AFIWC (and its operational sister organization, AFCERT).

The TrendCenter is based on the concept that by sharing information, each organization can be more secure than they would be otherwise. However, convincing organization of this advantage is proving harder than we expected.

Our experiences with respect to changes in the environment or incorrect assumptions (such as the belief that organization would share data in order to make everyone more secure) were not unique. As mentioned, SANS changed GIAC's name from the "Global Incident Analysis Center" to the "Global Information Assurance Certification". PricewaterhouseCoopers also established a network security portal, only to abandon it later. SecurityFocus.com tried to establish a global monitoring and analysis center called Aris, but that effort appears to be largely abandoned.

3.4 New Directions

With all of the changes we faced during the development of the effort, we shifted our emphasis from developing a portal to developing technologies for enterprises to deploy their own portal or use with their existing portal. For example, we are currently in the process of taking pieces of TrendCenter and applying it to the DOE's CPP program with LLNL's CIAC organization.

We developed a complete end-to-end solution, including tools for sites to process and sanitize of sensor logs, a portal for uploading and downloading information, back-end analysis tools to spot trends and identify vulnerabilities that are frequently exploited, and tools to automatically pull down the results from the analysis, configure a vulnerability scanner, and scan your site for the existence of those vulnerabilities in your network. In actual operations (see a description of the tools in Section 4.3 and Appendices A and B) the "portal" essentially disappears from users' normal experience, and much of the work is automated. The typical user, a system administrator who want

to protect his network, simply checks an HTML file on his system, and it identifies specific vulnerabilities on specific systems in his network that will probably be exploited.

4 TrendCenter, The Web Site

While the portal concept has been deemphasized in our new TrendCenter model, we still support one. Furthermore, it provides a convenient means for describing the larger process. This section provides a brief tour of the portal we assembled. The figures are presented at the end of this section.

4.1 Front Page

Figure 5 shows the home page when a user visits the portal. Across the top are the tabs to jump to the primary sections of the portal. The left most tab with the label "Attack Trends" and the Air Force logo brings you back to this page, the home page for the site.

The primary information element on the page is a graph showing which vulnerabilities are being exploited the most. As shown, the vulnerability identified by the CVE identifier CAN-2000-0071 was exploited the most (as least according to the sensor), with nearly one million instances observed during the most recent period of analysis. The number in parenthesis at the end of the CAN-2000-0071 label, "(80)", indicates the service port for this vulnerability. In this case, port 80 is the web server port. The next most frequently exploited vulnerability is identified as CVE-2000-0884, and is also a web vulnerability.

Below the primary graphic is a list that provides essentially the same information as the graphic. Each row in the list includes the vulnerability identifier, the port typically associated with that vulnerability (e.g., port 80 is used for web traffic), a relative score, and links to additional information. On the actual web page, the full top-20 vulnerabilities are displayed. The Score column provides the user with a relative likelihood that a vulnerability will be exploited. The top vulnerability is always given a score of 1000, and all other vulnerabilities are measured with respect to that score. For example, the second vulnerability has a relative score of 256 indicating that for every 1000 attacks against the first vulnerability, you will only see 256 attacks against this one. The relative scores drop off quickly, with the fifth most exploited vulnerability (not shown) having a relative score of only 10, indicating that it is exploited only 1% as often as the top most vulnerability.

The "Details" column on the far right of the list provides links to NIST's ICAT web site [Icat 03] and Mitre's Common Vulnerabilities and Exposure (CVE) web site [Cve 03a]. Clicking on the links will pull up each organization's information on that particular vulnerability. Typical information includes the platforms on which the vulnerability exists, a description of what an attacker can do to you by exploiting this vulnerability, and link to vendor patches for the vulnerability.

4.2 Analysis Page

Figure 6 shows the analysis page, which plots the number of times each vulnerability was attacked each day. This view shows whether a particular vulnerability is being exploited more or less today than it was in the recent past.

4.3 Tools Page

Figure 7 shows the Tools page where visitors can download tools to help them protect their network and help us gather information about common exploits. The first tool shown is a Perl script that will configure the Nessus vulnerability scanner to scan your site for the top exploits. The script, cverc_v3.pl automatically downloads from the portal a text file identifying the vulnerabilities currently being exploited the most, generates a configuration file for the Nessus vulnerability scanner, and optionally runs the vulnerability scanner. The result of the vulnerability scan is placed into an HTML file. After pulling down this Perl script, a system administrator can create a simple cron job to automatically run this script once a day, then he will always have an HTML file on his machine identifying the specific vulnerabilities on each machine in his network that are most likely to be exploited.

Also on the Tools page, but not shown in this figure, is the tool ats.pl, a Perl script that automatically sanitizes and summarizes Snort 2.x alert logs. A system or network administrator who wants to contribute to our analysis runs this tool on his log file and uploads the results to our portal. ats.pl sanitizes the log file by removing any information about the targets of the attacks, and the tool summarizes the log file by telling us how many individual systems were targeted by a specific attack. The resulting file is roughly 0.2% of the original file size. For example, a typical Snort alert file for a day at UC Davis is 350 MB, while the sanitized version is only 630 KB. Furthermore, since we only collect the log files in batch mode (e.g., only once a day as opposed to a continuous stream of reports), we can compress the resulting file. Thus, the original 350 MB of attack alert records becomes a 106 KB compressed file, a size that is only 0.03% of the original size. This much smaller size allows us to support a security portal without requiring a massive amount of bandwidth to our site.

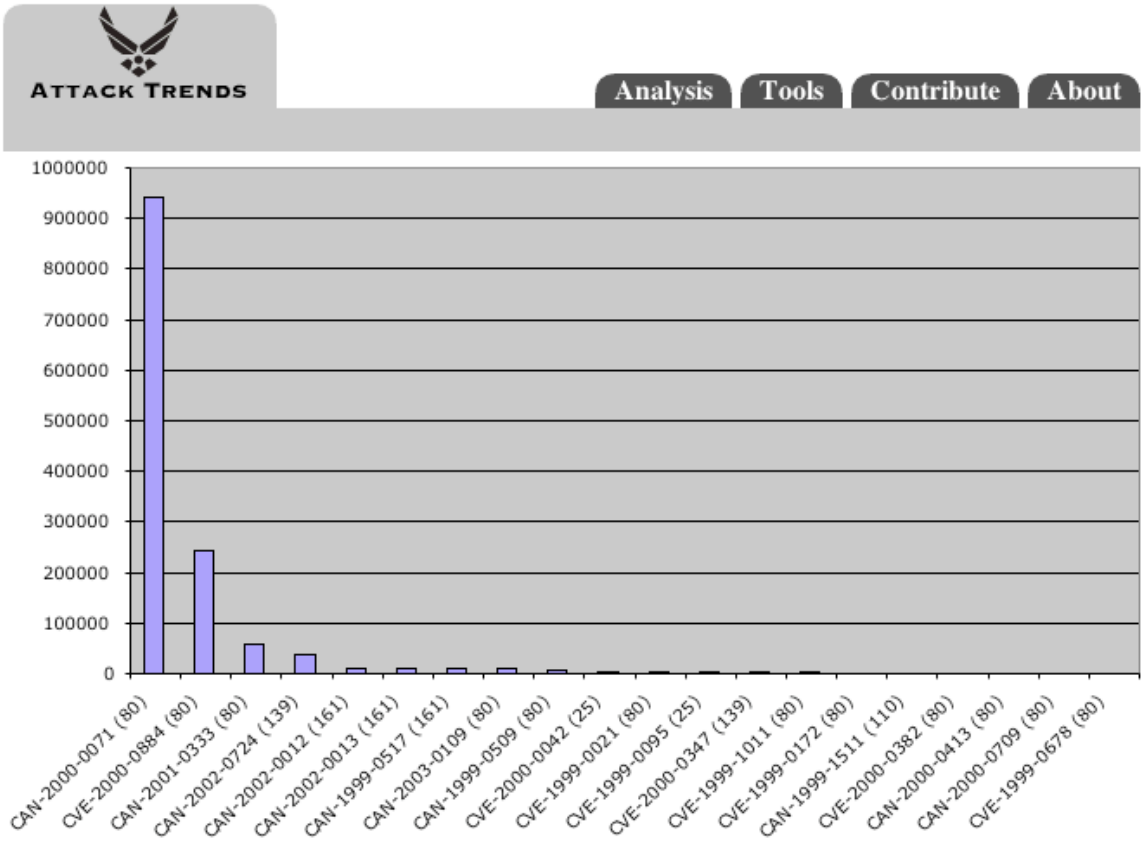
4.4 Contribute Page

Figure 8 shows the Contribute page, which guides a user through the steps of contributing a log file (not all the steps are shown in the figure). One of the most important steps any system or network administrator should take is to get full management support (Step 3), and we provide a link to the article “Bad Raps for Non-Hacks” [Rasc 03], a cautionary tale about how a system administrator can think he is doing the right thing but get into big trouble instead.

Once the system administrator has properly configured his sensor, runs the sanitizing and summarizing tool we provide, and receives support from his management, he can use a standard web form to upload his data through our secure web server. However, we also show the system administrator how to use the *curl* command to automatically upload the file. This way the system administrator can create a simple cron job that will automatically run the tool on the current Snort sensor log and upload the

resulting file. Thus, once set up, the system administrator never needs to manually process any files or visit our portal; everything is handled automatically and in the background.

4.5 Web Portal Snapshots



Vulnerability	Port	Score	Details
CAN-2000-0071	80	1000	icat, cve
CVE-2000-0884	80	256	icat, cve

Figure 5: Front Page

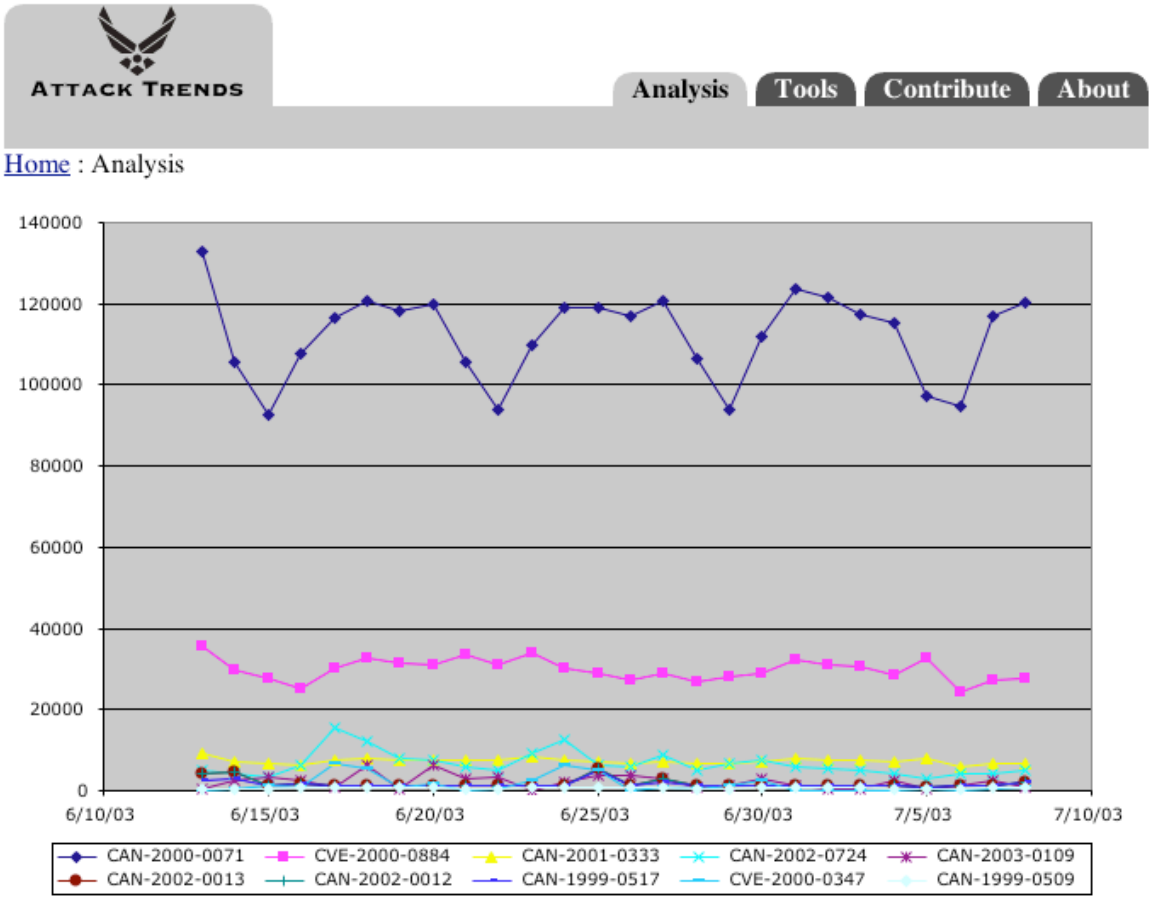


Figure 6: Analysis Page

Protecting My Network:

Scanner: Nessus

Where: http://www.nessus.org/

Synopsis: Nessus is an open-source vulnerability scanner. Nessus consists of a server, which actually performs the scanning, and a client that sends scanning results to the server and displays the result.

The Perl script below automatically downloads the latest list of vulnerabilities being actively exploited, determines which vulnerabities your Nessus server can scan for, and optionally scans a set of target hosts for those vulnerabilities.

Tool: cverc_v3.pl

Documentation: cverc_v3.txt

Figure 7: Tools Page

Step 1: Download a supported intrusion detection sensor. (Currently only Snort 2.x is supported) Install, work with, and turn the sensor until you are comfortable with it.

Step 2: Download the appropriate sanitization and summarization tool from our Tools section. Run the tool against the sensor log file, and make sure you are comfortable with the information that it is producing (e.g., make sure there are no local IP addresses in the output).

Step 3: **Get the support of your management.** Show them files you plan to upload, and show them it contains no information about specific IP addresses in your organization. This may sound paranoid, but read the article Bad Raps for Non-Hacks which showat that trying to do the obviously good thing can get people in trouble.

Figure 8: Contribute Page

5 Issues

While we succeeded in developing and deploying an end-to-end solution, from collecting and contributing sensor feeds to identifying specific vulnerabilities on specific systems in a system administrator's network that will likely be attacked in the near future, we still ran into a number of issues about which we are still concerned. In this section we examine two of the most important: quality data feeds and a consistent vulnerability identification scheme.

5.1 Garbage In, Garbage Out

The Air Force ASIM sensors used by the AFIWC/AFCERT organizations are also managed by those organizations. This ensures that they receive a relatively consistent data feed. Our TrendCenter model, on the other hand, is at the mercy of voluntary submission of feeds. Furthermore, because we have no reach-back to the original sensors, we have no way of verifying the content of the information.

For example, in Figure 6 there is a definite periodic cycle in the number of attacks per day on the first vulnerability, CAN-2000-0071. Each dip in the cycle occurs on the weekend. Do attackers take the weekend off? Are machines at businesses infected with automatic scanning tools and then turned off for the weekend? Or are many of the reports of attacks simply false alarms, flagging legitimate business activity that happens to fall off during the weekends? Because we do not have access to the original data that generated the attack reports we cannot answer these questions.

Another problem is that many sensors are out of date on their signatures. SANS GIAC group still takes Snort feeds from the University of Maryland Baltimore County (UMBC) to use in their training classes, but the rule set was so old that we could not identify the date of the rule set used. A survey of web pages listing Snort alert summaries via Silicon Defense's SnortSnarf program also showed very old rule sets. By accepting logs from such out of date sensors, newer attacks would be under represented while older attacks would be over represented.

In short, the quality of any analysis of data is at the mercy of the quality of the original data – "garbage in, garbage out" (GIGO). In the future we will need to determine how best to address this issue.

5.2 CVE Issues

TrendCenter is based on the premise that by observing attacks at other sites, we can predict which vulnerabilities will be attacked our site and prepare for those attacks. To achieve this capability TrendCenter needs to collect not what attacks are being observed but what vulnerabilities are being exploited. There is obviously a strong correlation between the two concepts, but the subtle difference is important. When an IDS sensor generates an attack report, we need some means to determine the vulnerability the attack exploits.

By using a common vulnerability identification scheme, we can pair up information from IDS sensors with information from vulnerability scanners, and this

capability is what allows us to identify specifically which vulnerabilities and on which systems will probably be attacked in the near future.

Many years ago a number of organizations formed the Common Vulnerabilities and Exposure (CVE) group [Cve 03a] to create a common scheme for identifying vulnerabilities, and the CVE group maintains an extensive list of computer security products that support the CVE identification scheme [Cve 03b]. Snort, the sensor we chose to initially work with, supports CVE by allowing the creator of a rule to identify the CVE identifier that the attack exploits.

Unfortunately, many of the Snort rules did not reference a CVE identifier. Most, however, did identify a second vulnerability identifier, SecurityFocus's Bugtraq ID. Thus, because we could not map a Snort attack record to a specific CVE vulnerability, we ignored some attack reports.

While talking to personnel involved with NIST's ICAT vulnerability database, which also extensively uses CVE identifiers, NIST complained that the CVE effort was very understaffed with effectively only a single person processing and updating the CVE information. This in turn led to delays from the time a vulnerability was initially published and when that vulnerability was given a CVE identifier.

We believe this delay in creating CVE identifiers for new published vulnerabilities may be the reason many Snort rules did include the CVE information – it simply did not exist when the Snort rule was added to the rule database. Many more Snort rules included a Bugtraq ID. Since many vulnerabilities are initially published on SecurityFocus's Bugtraq mailing list, we should not be surprised that a vulnerability is assigned a Bugtraq ID before a CVE identifier.

In retrospect, we should have initially standardized on Bugtraq IDs and not CVEs. In the long-term however, we will probably need to maintain mappings between multiple vulnerability identification schemes.

6 To Do

While we achieved a lot, especially given several changes in direction during the course of the project, we still have a list of "to do" items that we need to work on for the future. The following is a partial list.

- Support for more vulnerability schemes. As mentioned in the previous section, CVE proved inadequate as a means for mapping attack alert reports to the vulnerabilities that the attack exploited. At a minimum we need to support Bugtraq IDs, and we suspect several other schemes will need to be supported as well.
- Support more sensors. We built our initial capability around Snort because it is freely available, used by a wide number of organizations, highly configurable, and well documented. However, we obviously need to support some of the major commercial sensors such as ISS RealSecure.
- Detecting new attacks. Because our focus to date has been on identifying which known vulnerabilities a system administrator needed to patch, we did

not spend time developing techniques to detect new attacks. However, detecting the subtle spread of a hitherto unknown attack against an unknown vulnerability is an important problem, and we hope to address this in future implementations.

- Deeper analysis. To date we have only created the equivalent of Amazon.com's "top sellers" list, and this same list, or prediction of attacks a site will see, is given to every site. We hope in the future, once we are able to collect data from many more sites, we can develop techniques to provide a more personalized prediction capability.

7 Conclusions

This report summarizes our work performed to date for the Air Force SBIR TrendCenter project. We began in Section 1 with a summary of the problems we saw and our vision of a solution. In Section 2, using statistics, we examined in greater detail the threat environment faced by sites today and concluded that intrusion detection as originally envisioned cannot work in this environment. In Section 3 we examined the TrendCenter vision, both as originally conceived in early 2000 and the final vision in the summer of 2003. In Section 4 we give a tour of the web portal we created as part of this project. In Section 5 we examine some of the technical issues that we still face. Finally, in Section 6 we summarize some the items on our "to do" list.

8 References

[Ande 80] J. P. Anderson, "Computer Security Threat Monitoring and Surveillance," Tech. Rep., James P Anderson Co., Fort Washington, PA, Apr. 1980.

[CERT 03] http://www.cert.org/stats/cert_stats.html

[Coun 03] http://www.counterpane.com/

[Cve 03a] http://cve.mitre.org/

[Cve 03b] http://cve.mitre.org/compatible/

[Denn 87] D. E. Denning, An Intrusion- detection Model, *IEEE Trans. on Software Engineering*, SE-13, pp.222-232, February 1987.

[Dshi 03] http://www.dshield.org/

[Giac 03] http://www.giac.org/

[Icat 03] http://icat.nist.gov/

[Inci 03] http://isc.incidents.org/

[Rasc 03] Mark Rasch, "Bad Raps for Non-Hacks", SecurityFocus, June 16, 2003, http://www.securityfocus.com/columnists/167

[Sans 03] http://www.sans.org/

[Schw 02] John Schwartz, "Year After 9/11, Cyberspace Door Is Still Ajar," *New York Times*, 9 Sep 2002.

[Stol 88] C. Stoll. *Stalking the Wiley Hacker* . Communications of the ACM, 31(5), pp. 484-497, May 1988.

Appendix A: cverc.txt

```
NAME
     cverc - Generate a Nessus client configuration file from a
     list of CVEs

SYNOPSIS
     cverc [-v] [-s] [-o] [-w file] [-c file] [-h host] [-p
     port] [-u user] [-P pass] [-r -f result_file -t target -T
     type] config_file report_file

DESCRIPTION
     The cverc script takes a list of CVE numbers, obtained from
     AttackTrends by default, a creates a new configuration
     script for Nessus that will run the scans corresponding to
     the CVE numbers. This script requires a basic Nessus
     configuration file as a template. It creates an HTML report
     file which specifies which Nessus plugins it was able to
     enable based on the parameters passed to the script and the
     set of available plugins.

REQUIREMENTS
     The script requires Perl5 and the Getopt::Long module.

        If the -P option is not used to pass the password on the
        command line, then the stty program needs to be
        installed on the system for the password to be input
        interactively.

        The curl program is required to download the CVE list
        from AttackTrends. If curl is not installed, the CVE
        list will need to be downloaded manually and saved to a
        file. This filename will then need to be passed to the
        program via the -c option.

OPTIONS
     -v, --version
        Prints version and exits.

     -s, --no-safe
        Disables safe check mode in Nessus. This will allow
        dangerous plugins which may crash the scanned host to
        run. Use with caution. If this option is not set, unsafe
        plugins will be disabled even if they are in the CVE
        list.

     -o, --overwrite
        Overwrite the config_file with the new configuration
        file generated by this script.

     -w <file>, --outfile=<file>
```

```
      Output the new configuration file to <file>. This option
      must be set if the overwrite option is not set.

   -c <file>, --cve-file=<file>
      Use the specified file to generate the new
      configuration. <file> may be an existing file on the
      system or a URL. If this is not specified, the script
      will download the most recent CVE list from AttackTrends
      instead.

   -h <host>, --nessus-host=<host>
      Override the Nessus scanner host specified in the
      configuration file and use this host instead.

   -p <port>, --nessus-port=<port>
      Override the Nessus scanner port specified in the
      configuration file and use this port instead.

   -u <user>, --nessus-user=<user>
      Override the Nessus scanner user specified in the
      configuration file and use this user instead.

   -P <pass>, --nessus-password=<pass>
      Specify the password on the command line. If this option
      is not set, the password will be prompted for
      interactively.

   -r, --run-nessus
      Run Nessus with the newly generated configuration file.
      The following must be passed in order to use this
      option:

      -f result_file
        The name of the Nessus report file. All results will
        be written to this file.

      -t target
        The name of the targets file. This file contains the
        list of targets to scan.

      Optionally, the following may also be specified to
      change the Nessus report format:

      -T type
        This corresponds to Nessus' -T option. Please refer to
        the Nessus manual for valid types. If this option is
        not set, this script will default to "xml" as the
        output type.

EXAMPLES
   The following will take the existing Nessus client config
   file called $HOME/.nessusrc, overwrite it and place the
   information about plugins enabled in the file plugins.html.
```

```
     Since the Nessus password option was not used, it will
     prompt for the password of the user listed in
     $HOME/.nessusrc:

        user@blah:~$ cverc -o $HOME/.nessusrc plugins.html
        Enter Nessus client password for foo@bar:1241>

     Like above, but now passing the password on the command
     line:

        user@blah:~$ cverc -P blah -o $HOME/.nessusrc
        plugins.html

     The following will enable the unsafe plugins, take the
     existing client config file called myconfig, write a new
     configuration file called nosafe and put the plugins report
     in the file nosafe_plugins.html. Then run the Nessus client
     with the nosafe configuration file using the target file
     nosafe_targ, with the report type being HTML and the report
     file called nosafe.html:

        user@blah:~$ cverc -s -w nosafe -r -f nosafe.html -t
        nosafe_targ -T html myconfig nosafe_plugins.html

NOTES
     This requires the command line interface to have been
     compiled into the Nessus client, which is done by default
     on current Nessus compiles.

     If the script hangs during execution, check that the
     command line interface is properly responding by manually
     trying the following:

        nessus -c config_file -q host port user pass target
        result_file

SEE-ALSO
     nessus(1)

AUTHORS
     This script was created by Melissa Danforth for
     AttackTrends.
```

Appendix B: ats.txt

```
NAME
      ats - Attack Trends Sanitizer Perl script for Snort fast
      format alert files

SYNOPSIS
      ats [-debug] [-sd MMDD] [-ed MMDD] [-s <ip or part of ip>]
      <alertfile>

DESCRIPTION
      This script parses fast format alert file generated by
      snort. Takes in an <alertfile> as the argument and returns
      the parsed output in <alertfile>.parsed. The output file
      contains the following data:

         <# of unique alerts> <month/day> <src-ip> <sid> <dst-
         port>

         where # of uniq targets = <# of uniq targets>

NOTE
      If there are 'n' consecutive identical alerts, they are
      counted as 1. For example, if one attacker attacks one
      target with the same attack consecutively 500 times, it is
      counted as only one attack. On the other hand, if the
      attacker attacks 2 targets alternatively for 500 times each
      we would count them as 2 different attacks each launched
      500 times.

OPTIONS
      -debug
         When included prints out some helpful debug info.

      -s
         The alerts that have their source IP address belonging
         to the given IP class are suppressed and are not
         processed beyond the first parse. This option can be
         repeated to suppress multiple IP addresses.

      -sd
         Only those alerts generated on or after this date are
         processed beyond the first parse.

      -ed
         Only those alerts generated on or before this date are
         processed beyond the first parse.

DATA EXTRACTED
      Here is a list of the data extracted from the log file.

      $packet->{TIME} # Time of packet
```

```
      $packet->{SID} # The Snort alert ID that can be mapped into
      a CVE ID.

      $packet->{SRCIP} # Source IP Address

      $packet->{DSTIP} # Destination IP Address

      $packet->{DSTPORT} # Destination PORT Address

      The reference array has been used for extensibility;
      adapted from SnortLog by Craig Smith
      <smithc@cinstate.cc.oh.us>

AUTHOR
      Senthilkumar G. Cheetancheri
      cheetanc at cs dot ucdavis dot edu
```