ASTER: Active Smart Targets for Effective Response

Frank Adelstein, PhD
Odyssey Research Associates, Inc.
(607) 266-7104
fadelstein@oracorp.com

Abstract: In this paper, we describe a new approach to intrusion detection and correlation, in which we actively control or "mark" the information seen by each adversary that probes the site. When the adversary attacks, defenders detect the marked information and use it to correlate the attack and the probe. More complex correlations can be used to detect larger patterns, such as coordinated attacks. We have developed ASTER, a system that consists of (1) Active Smart Targets that disseminate and later recognize the marked information, and (2) a correlation engine to analyze the information. We describe the feasibility prototype we have implemented and discuss our future plans.

1 Introduction

Modern intrusion detection schemes log all available information and then sort through it to find attacks. This passive approach greatly reduces the signal-to-noise ratio of the logs. Until the fundamental way in which information is gathered changes, only very limited correlations can be made against coordinated attacks.

Unsophisticated attackers with limited resources can be detected and defeated with standard best-practice security measures, such as running firewalls and virus detectors, installing and maintaining current software updates, and auditing log data. In this paper, we focus on sophisticated attackers who engage in coordinated attacks. Sophisticated attackers present a difficult challenge, because their techniques often bypass standard detection mechanisms and penetrate or bypass firewalls. Worse, coordinated attacks may be launched from multiple machines, e.g., by using one for reconnaissance and another for the attack, and may target multiple machines, sometimes within the same department or organization. When the probing and attacking machine in a coordinated attack have different IP addresses, current techniques, which are based solely on IP addresses, cannot reliably recognize the link between the two machines.

In this paper, we introduce the concept of an *active* system [ART] that gives attackers information when they probe the target and detects the information when they attack the target. We can compare such a system to an electric eye that actively sends out a beam, which is reflected back to the eye's detector. The rest of the paper is organized as follows. Background material is presented in section 2, followed by related work in section 3. The ASTER system is presented in section 4, and two Active Smart Targets are described in Section 5. We discuss the implementation status of ASTER in section 6 and present a summary and future work in section 7.

2 Background

Current intrusion detection systems (IDS) take a passive approach to gathering data, relying solely on data provided by the system. This approach reduces the load on a system and provides for universal interoperability, in that the IDS uses standard, available data with little or no custom configuration of the OS upon which it runs, other than possibly enabling additional standard logging features. (Note that this does *not* imply any interoperability between *different* IDSs.) While relying on standard system logs simplifies IDS installation and configuration, it also limits what the IDS can detect.

A network IDS uses the information contained in IP packets. IP packets contain sender address (though this may be faked), recipient address, protocol identification (TCP, ICMP, etc.), and a few other fields like time-to-live (TTL). This information is intentionally limited in order to minimize packet headers. Unfortunately, the limited nature of the information restricts the overall power of an IDS.

Generally, an IDS logs large amounts of information to detect attacks and perform forensic analysis. An organization the size of a mid-sized university or company that wishes to save a couple of months of log data from its routers and firewalls can easily accumulate hundreds of gigabytes of data. Complicating this situation is the fact that these highly visible organizations are under an almost constant assault by probes and attacks. Worse, not every probe leads to an attack, so there is a low "signal-to-noise" ratio in the log files.

These organizations need a way to increase the effectiveness of their log files. In this paper, we present ASTER – Active Smart Targets for Effective Response, an extensible system that performs complex correlations, including linking probes and attacks.

Problem Statement: Rather than logging a huge amount of data that contains very little useful information, we need to gather and store information that helps us correlate probes and subsequent attacks.

3 Related Work

Much work has been done in intrusion detection, and many commercial products are available. However, few projects focus on manipulating information the adversary sees. Two common approaches are "honeypots"– systems designed to lure adversaries, or "fishbowls"–isolated, fake systems in which the adversary cannot cause any real damage or get any real information.

The Deception ToolKit [DTK] makes it easy to implement replacements for services. It produces fake responses and is intended to waste the time of an adversary. There is some ability to customize responses, but this feature tends to be used only to fake error messages rather than to gather information about the attacker.

The HoneyNet Project [HoneyNet] is a large-scale honeypot system designed to watch how attackers break into systems. The project is not intended for correlation or protection.

In [Cheswick], Cheswick describes how their honeypot system drew in a cracker

("Berferd"), and how they gathered information on him and logged all his actions while giving him minimal system access.

ManTrap [ManTrap] is a commercial honeypot intrusion detection system that enables administrators quickly and efficiently to set up fishbowls.

In general, honeypots and fishbowls lure in attackers and gather information. Unfortunately, the information is typically of limited value, since they run on isolated, stand-alone systems that are dedicated to luring attackers, rather than as part of production systems. None takes the approach we propose of feeding information to an attacker in order to correlate phases of an attack.

4 ASTER

ASTER is designed to increase the effectiveness of log files by actively manipulating information an attacker sees. ASTER's goal is to correlate the probe and the attack IP addresses, even in cases where two (or more) different machines are used. By linking the probe IP to the attack IP, we can more effectively search for truly hostile probes in conventional log files, while ignoring probes that do not lead to attacks. By properly mining audit log data, we can also gain insight into the bigger picture: the pattern connecting the phases of a coordinated attack.

The key concept of ASTER is active smart targets. An *active smart target* (AST) is a decoy target, meant to lure attackers and hand them marked cards by means of which coordinated phases of an attack can be recognized and correlated. A *marked card* is traceable information provided by the AST, a low-bandwidth, high-value data item that makes it possible to correlate coordinated attacks from multiple IP addresses.

In addition to the ASTs, ASTER has a correlation component, which mines the data gathered by the ASTs to gain additional information about the attack.

ASTER can be linked into other correlation engines as well as into response systems that can take actions such as change firewall rules on the fly to block further attacks. In addition, ASTER can be tied into a "fishbowl" environment that misdirects the attackers, wasting their time making them pursue harmless, but attractive, avenues of attack, while leading them away from valuable resources and gathering information on them.

In the initial version of the project we built a simple proof-of-concept system consisting of two prototype ASTs and a database interface for the marked cards and related information (e.g., the time the marked card was given and the IP address to which it was given). Next, we defined some more complex queries and correlations for our database API to support, such as queries over a time range.

In the next section, we describe the two ASTs that were implemented and the correlations that we can perform on the data they gather.

5 Active Smart Targets (ASTs)

In the following section, we describe the implementation of two active smart

targets. Each AST uses a different scenario. For each AST, we first describe the view presented to the attacker, followed by what is really happening, and then describe how the marked cards make this possible.

5.1 Scenario 1:

The story:

A cracker discovers that the password file (/etc/passwd) on a target machine is visible through the anonymous ftp account, since anonymous ftp allows access to a limited number of files without the use of a password. He downloads the file, runs a password-cracking program on it, and discovers that the "www" account has a trivial (i.e., single word) password on it, allowing him to log onto the system using the www account via the telnet program. Access to this unprivileged account gives him a toehold on the system from which he can attempt to elevate his level of access. From a second IP address, he telnets into the target machine and logs into the www account.

What ASTER does:

ASTER enables us to correlate the IP addresses of the machine that was used to steal the password file with the one that logs into the www account. ASTER does this by providing every probe with a different version of the password file, each one of which has a different password for the www account. The use of this unique password from a different IP address allows ASTER to correlate the probing IP address with the attacking IP address. By determining that a particular probe IP address is "malevolent" we can then check to see if that IP address has probed other ASTs. We can also search conventional log files for that address.

In addition, the www account is a *fishbowl*, a protected decoy system, in which the user can do no harm while all of his actions are observed and logged for later analysis.

How ASTER works:

On the AST machine, the FTP daemon is modified to behave analogously to CGIs on a web server. Attempting to download a file via FTP (the "get" command) invokes a script that checks if the name of the requested file is on a list of sensitive files. Currently, "/etc/paswd" is the only file listed. If the file is on the list, then a second script is invoked to generate a fake password file, otherwise the file is downloaded as usual.

The password file generator script uses a template to generate the password file. All entries except for the "www" account are static and the encrypted passwords are difficult to crack. The script fills in the entry for the encrypted password for the "www" account by encrypting a simple dictionary word "on the fly." It determines what dictionary word to use by consulting the main database. The script is passed the IP number of the machine attempting to download the file and checks if that IP address has requested the file before. If so, then it uses the dictionary word that was previously used. If not, then it uses the next available word from a list, stores that word in the database, and updates the index that indicates the next available word.

The configuration file for the telnet daemon (inetd.conf) was modified so that a fake login program runs instead of the normal one. This login program appears to allow users to log in, printing the usual prompts. However, any account other than the "www" account is denied access. In the initial version of the implementation, the www account is allowed to log in regardless of what password is used. Future versions may restrict access only to those passwords that have been disseminated. The user is put in a fake shell, which provides no access to system resources. In the initial version, only a minimal number of commands are implemented (this shell could be referred to as fsh for the "frustration shell"). When the attacker logs in to the www account, the password he used to log in is sent to the correlation program, which associates the attacker's IP address with the probe. The correlation program is described in Section 5.3.

The marked card:

The marked card in this scenario is the easy to guess password. It is disseminated in the fake password file, and is detected by the fake login daemon. The probe and attack represent two very different aspects and are likely to be from two different machines, but can be linked via the marked card.

5.2 Scenario 2:

The story:

A cracker discovers an unpublished web server running on a non-standard port. The main page of the server has a link to another machine that appears to be running an "interesting" application (e.g., accounting). The attacker then attempts to attack this secondary site, read "unpublished" files, and run vulnerable CGIs (e.g., finger).

What ASTER does:

ASTER associates the IP address of the machine used to locate the first web server with the IP address of the machine used to read the web page on the second web server. Scenario 2 is deliberately simple in order to allow us to define and test the correlation functions. However, we could increase the complexity of this scenario by encrypting the content of the first web page (which points to the second page) or requiring that the "attack" consist of exploiting a known web server bug, like a buffer overflow on a CGI, thus forcing the attacker to use more sophisticated techniques.

In any event, the machine name of the second web server is the marked card that enables us to trace the attack to the probe.

How it works:

The main page for the unpublished web server ("index.html") is actually a CGI script, although it looks like a static web page to the attacker (because of the URL name and its properties). The CGI script uses a template to generate the web page, and fills in the entry for the link to the second machine "on the fly." The URL is of the form:

```
http://machinename/filename
.html
```

where "machinename" is a fully qualified domain name (FQDN), like `abcdefg.astersubdomain.bigc`

ompany.com. In this case, the "abcdefg" is uniquely generated for each probe. Given the IP address of the machine attempting to retrieve the web page, the CGI fills in the entry for the machine name. If that IP address has requested the page before, it uses the name that was previously used. If not, then it uses the next available name from a list, stores that name in the database, and updates the index that indicates the next available name.

The key point is that *all* names point to the same IP address. In the initial implementation, we do this by using a small list of names, all with the same IP address. Future versions will involve either dynamic DNS host allocation or modification of the DNS. All requests go to the same web server, which is able to distinguish what name it is being called by using "virtual hosting" technology. Current web servers allow different web databases to be used depending on the name by which they are called. CGI scripts can detect this via the SERVERNAME environment variable, bypassing the need for configuring the web server to run multiple virtual servers. The server determines the name because it is specified in the HTTP request via the HOST: directive [RFC2068].

The second web server then creates an environment that appears to be appropriate for the intruder, for example, files in the /etc directory can refer to the appropriate machine name. While the web requests are being processed, the host name is passed to the correlation program.

The marked card in this scenario is the name of the second machine. It is disseminated by the CGI script and is detected by the second web server, as the browser specifies the host name through the Host: directive as per the HTTP1.1 specification [RFC2068].

5.3 The Correlation Program

The correlation program links the probe to the attack, and can also perform more complex correlations to detect larger patterns of similar behavior of the attacker, such as detecting if the attacker has probed other machines on our net. The correlation program runs separately from the ASTs. Ideally, it should run on a separate machine with severely restricted access rules, behind a firewall. It has access to the ASTER database and can perform correlations when an AST passes it a marked card or when a console user requests additional analyses.

The first type of correlation it performs is to link the probe and attacker IP addresses given a marked card. Whenever an AST gives out information, it records the IP address of the requestor and the marked information in its database. When an attacker hands us a marked card (e.g., a specific hostname or password), we look it up in the database and find the IP address from which the attacker first received the card.

Currently, the correlation program is limited to performing a simple link between the probe and attacker. However, the database library is able to perform far more complex correlations.

The marked card:

There are two primary calls to the database library, StoreEvent() and RetrieveEvent(). StoreEvent() is used to add an entry to the database. RetrieveEvent() gets matching events from the database. Events can match based on the IP address, the timestamp (date and time) or timestamp range, and the AST marked card data. Any field can be omitted as a search criterion—for example, it is possible to look just for events with matching IP addresses or events in the same timestamp range. Wild cards can be used in IP addresses, in order to check for a range of addresses (e.g., 192.175.*).

We plan to extend the correlation program to look for larger patterns. Given an attack, we can find the IP address that corresponds to the probe. Once we have the probe's IP address, we can find any probes that occurred around the same time, say within an hour. We can also find all IP addresses that come from the same Class C address space. Since the database interface currently supports these queries, it is a simple matter to add these capabilities to the correlation program. Extensions to the database interface include comparing the delay between the probe and the attack, as well as the overall scope of the attack (i.e., how many machines were targeted by that attack).

Our goal is to create a model of the attack. We could create a directed graph representing the types of links mentioned above. Then we could compare one graph to another to determine if the attacks have similar parameters (in time and space). Specifically, is there a pattern between machines that are probed and attack, and machines that are used to perform the probes and attacks? For example, are all machines in a subnet targeted, or only certain machines, perhaps ones with the same final number in their IP address? Are all vulnerable machines attacked, or are some probed and left alone? Do the attacks come from similar hosts or different ones? Do these probes and attacks share similar patterns in time? There are many questions to explore by mining the correlation data.

6 Implementation Status

We have demonstrated the feasibility of the ASTER approach. We are able to disseminate "marked cards," record that we have done so, and then later detect the marked cards and link them back to the probe. This project does not focus on response, so in our prototype the output of the system is simply a message in a log file. In a commercial system, ASTER would alert a response agent, providing it with all relevant information about the attack.

One of ASTER's strengths is that its behavior looks like normal system behavior and its marked cards appear to be legitimate system data. The ASTs cause no discernable delays. In the web site example, the page appears to be a normal, static web page, with no indication that the content is dynamic. In the ftp example, the password file appears to be a normal password file. An attacker could run a probe from two *different* IP addresses and compare the output. If he did this, the differences would be apparent. However, such double probing doubles the amount of work a reconnaissance system must do as well as its risk of detection.

Field	*Type*	*Null*	*Key*	*Default*
ASTid	int(11)	YES	MUL	NULL
IP	varchar(20)	YES		NULL
Timestamp	Timestamp(14)	YES	MUL	NULL
ASTdata	varchar(200)	YES		NULL

Table 1: Description of ASTdb table

We have not tried using shadow password files, but adding them to the existing system would be straightforward.

The storage overhead for the database can be computed as follows. For each marked card, the ASTdb table contains an ID field to identify which AST generated the marked card, the IP address from which the probe or other attack originated, a timestamp, and the marked card itself. Table 1 shows how the ASTs are defined in the mySQL database.

Each record uses a total of 245 bytes. Storing data for 100 distinct scans per day requires 24,500 bytes per day. To maintain 6 months of records requires roughly 4.4M of data to be stored. This is a very small amount of data for a database, and searches on a database of that size can be performed quickly.

Finally, we examine the detectability of the marked data we disseminate versus the level of effort required to write the code. However since the code is written only once and then reused repeatedly, this is factored into our analysis. The code was written in less than one month, and is able to mimic most of the system responses to a high degree of accuracy.[1] Given more time, we believe it should be fairly straightforward to produce highly realistic ASTs.

7 Summary and Future Work

We have created Active Smart Targets to disseminate marked data to potential attackers. We detect the marked data during subsequent attacks. By "tagging" the data, we increase the quality of the information we can obtain about an adversary. We have implemented two ASTs and the database framework, and have run some initial tests on the system with promising results. We can perform simple correlations to determine, given the IP address of an attacker, the IP address of the probe associated with that attack. We have also created a database library that allows rich queries to be performed in order to mine valuable information on the nature of complex attacks.

Future work will focus on several areas. First, we intend to add additional functionality to the correlation engine that will allow it to look at the bigger picture of the attack. It will also be able to provide a warning to systems in

[1] We note one "bug" in the current AST implementation of a decoy ftp daemon: downloading a non-existent file behaves in the same way as downloading a 0-length file. The system should generate a file-not-found error but currently does not.

sibling departments that do not run ASTs. By determining what IP addresses represent "hostile" or "aggressive" probes, system administrators can look through conventional log files (e.g., via "grep") to see if they have been probed by these hosts.

We intend to field a system at multiple universities to test it "in the wild." ASTER can provide key information as to what percentage of scans lead to attacks. This information is important to administrators and managers in order to determine the amount of resources that need to be dedicated to campus-wide network security. By field-testing it in a hostile environment, we hope to gain additional insight into ASTER's capabilities and understand where it can best serve security administrators.

Acknowledgement

We gratefully acknowledge support for this research by the Defense Advanced Research Projects Agency's Cyber Panel program, under DARPA contract DAAH01-01-C-R012. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either express or implied, of the Defense Advanced Research Projects Agency or the U.S. Government."

References

[ART] Leonard Popyack, Jr. and Stephen Taylor, "Active Response Technology," to appear in IEEE Spectrum, 2001.

[Cheswick] Bill Cheswick, "An Evening with Berferd In Which a Cracker is Lured, Endured, and Studied," AT&T Bell Laboratories.

[DTK] Deception Toolkit, Fred Cohen & Associates, http://all.net/dtk/dtk.html.

[HoneyNet] The HoneyNet Project, http://project.honeynet.org/

[ManTrap] http://www.recourse.com/download/white/ManTrap.pdf

[RFC2068] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1," DEC, MIT/LCS, RFC2068, January 1997, http://www.w3.org/Protocols/rfc2068/rfc2068.