8 Insider Threat

Managing security used to be an easy task. The expectation was that you installed a firewall to protect the corporate assets from attackers. Every now and then, you had to poke a hole into that firewall to enable some application to talk to the Internet or vice versa. If things were taken really seriously, you also had to install a network-based intrusion detection system (NIDS) to monitor the traffic entering the corporate network. It was a lot of fun—maybe I am just a masochist—to look at the NIDS logs to find how many times you were scanned in the past 24 hours. It got a bit frustrating every now and then when you did not really find any important events that would indicate someone actually attacking you. However, you did your job and everyone seemed to be happy.

Unfortunately, those times are over. If you are responsible for security at your company, now you not only have to make sure that there are no attacks from the outside, but you also must deal with the threat from within. It is no longer just the evil hacker on the other side of your firewall who you have to protect your users from. It could be your coworker, your teammate who is the one you have to turn in after the next investigation. The malicious **insider** is the new threat that we need to protect from, in addition to all the threats coming from the outside.

In this chapter, I take a close look at the insider threat problem. We take a look at what the insider threat really is. I answer questions such as, "Why should you have an interest in insider threat?" and "Why does visualization play an important role in dealing with malicious insiders and mitigating the insider threat?" To make the insider threat problem slightly more tangible, I walk through three sample insider attacks and show how they can be detected with visualization of audit trails and electronic log files. After you have a better understanding of the types of malicious insider activities, I present a generic

framework that can be used to mitigate the insider threat problem by visually identifying possible malicious insiders in an early stage. The detection process that I present is by no means a complete insider threat solution, nor do I claim it will necessarily detect all of them. The approach, early-stage research, needs to be extended and refined to become more efficient and reliable. However, the process has been applied to real data and has yielded successful results! The interesting property of the detection process is that it is extensible by design. Changes in the threat landscape or new findings and knowledge can be incorporated easily.

INSIDER THREAT VISUALIZATION

Why should visualization be used to detect malicious insiders? To analyze insider threat, we have to consider huge amounts of data—much more than for other security analysis tasks. This problem differs from others where we can just monitor violations. Malicious insiders often have legitimate access to machines and data. These activities won't be logged by security-monitoring solutions. They are not exceptions or breaches. Log levels might have to be turned up to detect malicious insiders. In addition, we are not dealing with network layer problems. Firewall log files will be of limited use to identify malicious insiders. Fraud, especially, is almost exclusively visible in application logs. In some cases, however, we need the network level logs, too. Again, here is another factor that increases the amount of data at hand. So far, you could still make the argument that deterministic or statistical methods could be used to do the analysis. This is where the core strength of visualization comes in. Often, the questions, the things to look for, are not known in advance. They are uncovered during the analysis process. Visualization is a great tool for provoking those questions and helping find answers to them.

The dynamic nature of fraud makes it a particularly challenging detection problem for static algorithms. Fraudsters quickly adapt to fixed threshold-based detection systems. With a visual approach, we are looking for any unusual patterns, making it an effective strategy for identifying new classes of fraud.

Visualization is a key technique to help identify and analyze insiders.

WHAT IS A MALICIOUS INSIDER?

To understand insider threat, we should start by defining what a malicious insider is:

> Current or former employee or contractors who intentionally exceeded or misused an authorized level of access to networks, systems, or data in a manner that targeted

> a specific individual or affected the security of the organization's data, systems or daily business operations.[1]

Malicious insiders are people with legitimate access to information. They are not trying to circumvent any access control mechanisms to gain access to interesting information. They already have access. This is a really important point. It means that traditional security devices do not have the capabilities to find these types of attacks. The NIDSs, as we know them, are monitoring network traffic for known patterns of malicious behavior. They are completely useless in scenarios where there is no attack and no violation of communication policies. The same is true for firewalls, host-based IDSs, antivirus, and so on. All these devices are good at detecting malicious traffic or behavior.

The market has realized that this is a shortcoming and created a new market segment for "insider threat tools." But these tools are no panacea. They might claim to have the capabilities to detect malicious insiders, but that's unfortunately an illusion. How are you going to find a malicious insider who is selling company secrets to the competition if all he does is take printouts of confidential financial statements he uses for his daily job with him when he leaves work? No tool or device will prevent him from doing so. Even if there were some kind of a device to detect documents when they are leaving the company premises, who says that the employee is not authorized to take the documents home to work on them? I think you get the point. It is illusive to believe that any device or tool can detect all insider crimes. However, what does exist are tools that can *help* detect malicious insiders. We will see in this chapter that these tools are not necessarily specific devices, but often log files of systems and application behavior that will help convict the guilty.

THREE TYPES OF INSIDER CRIMES

The insider threat space can be subdivided into three categories, as shown in Figure 8-1. The boundaries between the three categories are smooth. The categories overlap in many ways. For example, an information theft case can also be a sabotage and a fraud case at the same time.

[1] "Insider Threats in the SDLC: Lessons Learned From Actual Incidents of Fraud, Theft of Sensitive Information, and IT Sabotage," by Dawn M. Cappelli, Randall F. Trzeciak, and Andrew P. Moore (www.cert.org/archive/pdf/sepg500.pdf).

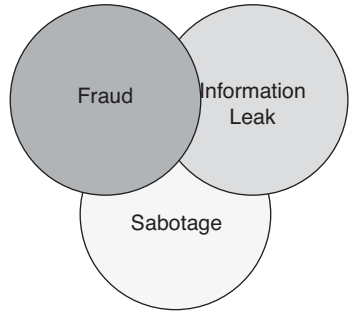

Figure 8-1 Insider threat can be subdivided into three categories: information theft, fraud, and sabotage.

The following are short descriptions for each of the insider threat categories:

- **Information theft** is concerned with the stealing of confidential or proprietary information. This includes things such as financial statements, intellectual property, design plans, source code, trade secrets, and so on.
- **Fraud** deals with the misuse of access privileges or the intentional excess of access levels to unjustly obtain property or services through deception or trickery. Examples range from selling confidential information (social security numbers, credit card numbers) to modification of critical data for pay (driver's license records, criminal record, welfare status) to stealing money (through financial applications, tampering with order processing system, and so on).
- **Sabotage** has to do with any kind of action to harm individuals, organizations, organizational data, systems, or business operations. Examples include the deletion of information, bringing down systems, website defacements, logical bombs, and so on.

Let's take a separate look at each of the three types of insider threat individually to understand better what is involved in each of these three types of insider threats.

INFORMATION THEFT

Information theft is incredibly hard to detect. Information is intangible. Copying information is quick and simple. We all carry huge data stores around with us: cell phones, USB sticks, iPods, and laptops. They can all be used to store and transport thousands of documents. It is almost impossible to detect someone sneaking out a confidential document loaded on a cell phone. It is not feasible to inspect all phones of people as they leave the office. And often, information is not removed on a physical device. It is put into an email message sent to a recipient outside of the company.

Protecting information to prevent it from being leaked turns out to be a challenging task. Insiders have legitimate access to confidential information. They need this information to do their daily jobs. Stealing confidential information does not involve an act of access violation or an attack. Customer records, for example, can be accessed by support personnel, the finance department, sales, professional services, and so on. Source code for computer applications needs to be available to all developers. They often have access to the entire product or, even worse, all the products that are hosted on the same source code control system.

An entire market space has formed around protecting information. There is still some confusion as to what to call this space. Some call it **extrusion detection,** alluding to the fact that there is not an intruder, but something is extruding. I personally think this is a horrible name, but that's my opinion. The space is also called **information-leak prevention** (ILP) or **data-leak prevention** (DLP). These names resemble much better and more descriptively what these information protection tools are doing.

Data-Leak Prevention Tools

What exactly are DLP tools doing? I usually point out two features. The first one is nothing really new. A DLP watches the network traffic, much like an IDS (except that the DLPs focus more on the application layer than IDSs do). It is configured with signatures that are looking for confidential or sensitive information leaving the network (for example, social security numbers or credit card numbers). NIDS could do this, too. The difference is that the DLPs are specifically built to do this type of analysis. A DLP would, for example, monitor network traffic for credit card numbers. If traffic is identified, the DLP can display the violating transaction. Transactions can be emails, IRC chats, Web connections, and so forth. DLP interfaces are optimized toward displaying this type of information. NIDS are not meant to display this type of information in the context of a transaction. They are good at flagging the violation, but not at showing the original transaction. NIDS are also not optimized toward application layer protocol analysis, such as email dissection.

Although the DLPs have the upper hand when it comes to information display, they could really learn from their NIDS colleagues. The products I have seen require configurations to be done in text files, don't support full regular expressions, do not have a good way to update the signatures, and so forth.

The second feature that DLPs offer is document watermarking or document tracking. This is where things are getting interesting. Your financial statements, for example, are documents that are not written for general consumption. With a DLP, you can register this document with a centralized document registration server. It then monitors the network traffic and finds instances of the document floating around. This is only

the beginning. The DLP can also detect whether someone copied the Word document into an Excel spreadsheet. Or, if someone takes a paragraph of the original document and pastes it into an email, the DLP can still detect the document.

Logging In Data-Leak Prevention Tools

Most DLP products are horrible at logging! One of the products does not even have predefined logging built in. The user has to define, on a per-rule basis, what the log messages look like. Bad idea! Don't let the user muck with logging. If security device vendors cannot get logging right, how are users supposed to? On top of that, not defining a logging format makes integration and interoperability a nightmare. I am digressing, but I had to bring this up.

The second drawback of DLPs is one that might sound familiar to a lot of you: **false positives.** These products have exactly the same problems that NIDS have. It is extremely difficult to configure the DLPs precisely enough to detect all the cases of information leaks while not flagging benign instances. Assume you want to watch for social security numbers floating around on the network. You would configure a rule to monitor all the traffic for strings of the form `\d{3}-\d{2}-\d{4}`. This corresponds to three numbers followed by a dash, then two numbers, another dash, and then followed by four numbers. Depending on where this sensor is placed, there might be legitimate traffic containing social security numbers. You now have to go ahead and teach the DLP all the instances of traffic where it is okay to see social security numbers. Try to be exact. You need to configure every client that should have access to this data as an exception to the rule. If an entire subnet of machines is allowed to access the data with social security numbers, but a couple of machines should be prohibited, you need to define these exclusions! Be careful about **false negatives** while setting up your policies. In various cases, policy violations can be missed. For example, encrypted traffic or unsupported versions of documents make it impossible for the DLP to analyze the documents. It is really exactly the same problem as IDS signature tuning. The fact that DLP vendors have not tried to learn from the NIDSs is not helping.

Information-Leak Examples

Take a look at Table 8-1. It shows a few examples of information theft and information leaks. It is worth noting that none of the attacks require technically sophisticated means or knowledge. Anyone can commit information theft.

Table 8-1 Examples of How Information Can Be Leaked

Activity	Example
Printing documents	When information is on a sheet of paper, it is simple to remove it from the workplace and share it with other people.
Copying information to disks	Similar to printing documents, information can be removed from the workplace without anyone noticing. Has anyone ever checked your cell phone for data before you left a facility?
Sending information via email	A common scheme is to send documents via email. A lot of times, private Web mail accounts are used to receive the information on the other side.

INFORMATION LEAK: EXAMPLE

In February 2007, a fairly large information-leak case made the news.[2] Scientist Gary Min faces up to 10 years in prison for stealing more than 16,000 documents and more than 22,000 scientific abstracts from his employer, DuPont. The intellectual property he was about to leak to a DuPont competitor, Victrex, was assessed to be worth $400 million. This is an extraordinary event of information leakage. However, similar cases occur all the time when employees are leaving their jobs. Shortly before actually leaving the company, they start collecting documents, emails, source code, and so on. In a lot of cases, the employees feel that they are entitled to this information. However, for the company this could mean significant losses, as the preceding case shows.

What can be done to detect these cases? Is it even possible to prevent them? Information leaks always have to do with people accessing information. Most often, the people have legitimate access to this data, which makes it a bit harder to detect mischief. Log entries will not flag suspicious activity. In addition, hardly anyone is auditing all document access. This is the precondition for this type of analysis. Access to critical documents needs to be audited. When this in place, visualization can be used to analyze the access logs. Figure 8-2 shows an example of a document access log. If your applications do not write an access log that contains this type of information, you could use file auditing (see Chapter 2, "Data Sources") to audit document access.

[2] www.informationweek.com/news/showArticle.jhtml?articleID=197006474

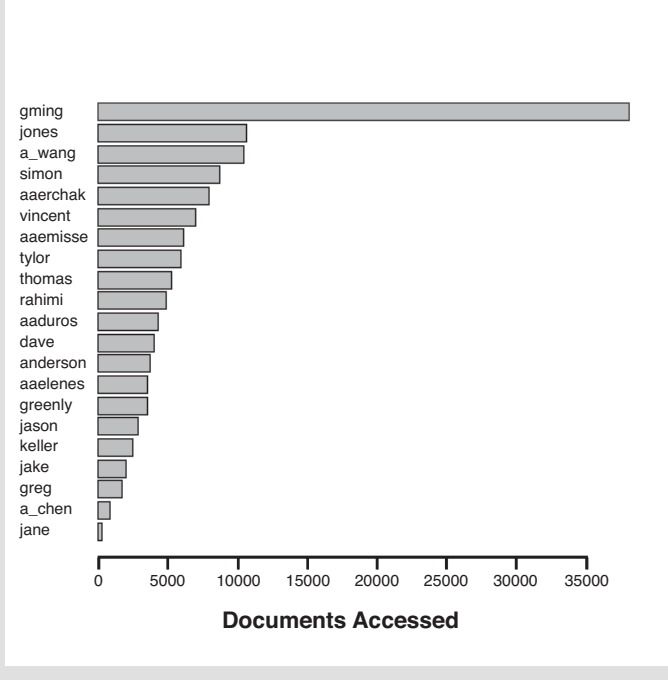

Figure 8-2 This bar chart shows how many different documents each user accessed.

If DuPont had been monitoring a document access graph like the one in Figure 8-2, they would have seen that Gary was accessing a lot of documents. The logical next step would have been to walk over to his desk to figure out why he was looking at all these documents. Granted, this is a focused way of monitoring activities. You would have to be specifically monitoring document access. This approach does not necessarily scale. The number of things to monitor is likely going to explode. Therefore, we need a more generic way of monitoring user activity. Figure 8-3 shows a slightly more generic way of monitoring user activity in a stacked bar chart. It is now possible to encode not just document access, but any type of activity in this chart. In this example, it is very visible that `gmin` showed the most activity. This is a fairly good indicator that you need to have a closer look at what he is up to.

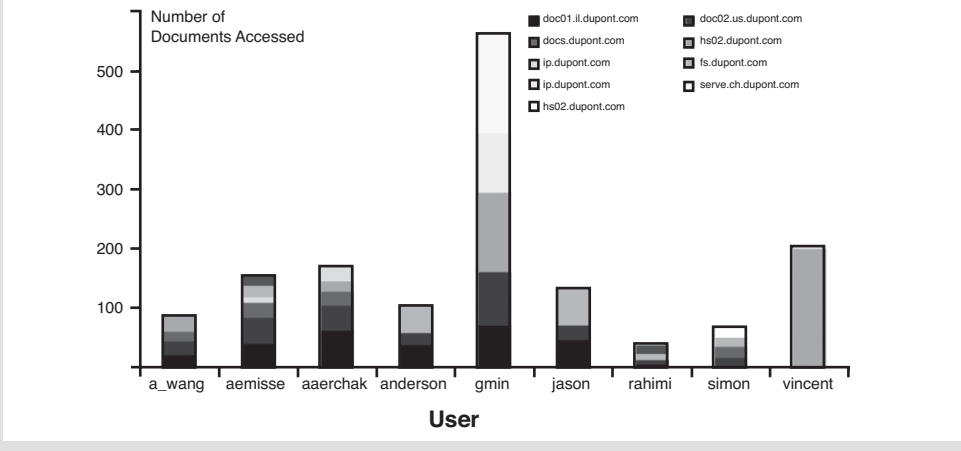

Figure 8-3 This stacked bar chart shows a way of monitoring data access on servers. Each user is represented as an individual bar, split by the different servers that were accessed. Each access constitutes a unit on the y-axis.

Yet another way of monitoring users would be to use a link graph, where each machine's amount of access (to, say, certain servers) for a user is encoded as an edge in a graph. The activity of just `gmin` is shown in Figure 8-4.

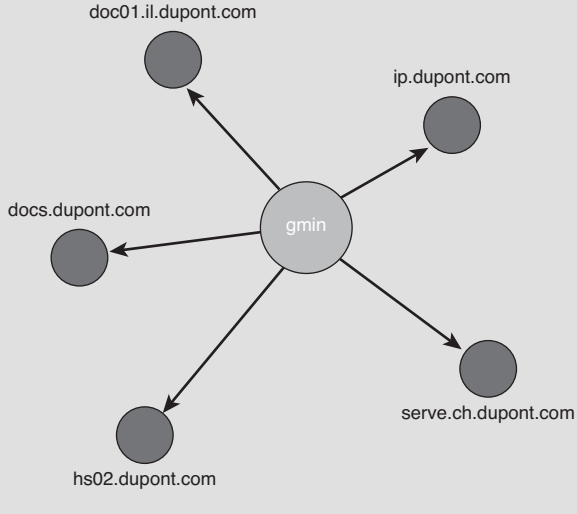

Figure 8-4 A link graph showing a single user's access of servers.

The size of the circles can be used to encode the amount of activity seen from that user. This graph can be extended in various ways. For example, the user nodes could be colored based on a separate property. Using the role of the user as the color shows how users behave relative to their role.

FRAUD

A class of insider threat that causes a lot of headaches in the corporate world and some significant loss is **fraud.** The reason for the activity and interest in fraud is, of course, that the potential monetary gain is compelling. This attracts well-organized and sophisticated groups that turn out to be involved in a lot of fraud cases. These groups are experienced and know exactly how to conceal their crimes. Frequently, these groups do not commit fraud themselves, but persuade people inside of companies to execute the crime in return for payment. There are many different types of fraud. Table 8-2 discusses some types that could likely be identified through electronic logs or electronic audit trails. Because of the difficulty of detecting these fraud cases in log files or audit trails, the table does not list tax fraud, real-estate fraud, insurance fraud, bribery, and so on.

Table 8-2 Different Fraud Categories and Examples

Fraud Category	Description	Examples
Occupational fraud	"The use of one's occupation for personal enrichment through the deliberate misuse or misapplication of the employing organization's resources or assets" Source: *Fraud Examiner's Manual*	False overtime Payroll and sick-time abuses
Financial statement fraud	Deliberate misrepresentation of the financial condition of an enterprise accomplished through the intentional misstatement or omission of amounts or disclosures in the financial statements to deceive financial statement users	Improper revenue recognition Overstatement of assets
Misappropriation of assets	Cash receipt schemes, fraudulent disbursements of cash, and theft of inventory	No receipt issued Sales during nonbusiness hours Any type of stealing of checks issued to the company

Fraud Category	Description	Examples
Financial institution fraud	Fraud related to banking institutions	Any false accounting entries Unauthorized withdrawals Unauthorized unrecorded cash payments Account sharing
Computer and Internet fraud	Tampering with computer programs, data files, operations, and so forth resulting in losses sustained by the organization whose computer system was manipulated	Phishing Auction and retail schemes

Each category in the table has its own detection mechanism. Most of the fraud cases can be identified by system irregularities—either through audits and checks in the applications or through monitoring of system-level parameters, such as changes of files, accessing of critical files, or accessing of database tables. Often it also helps to monitor access paths to uncover new paths or detect unauthorized access.

The Fraud Triangle

To determine how to detect fraud, it helps to understand what it takes for someone to commit this crime. For a person to commit fraud, a concept known as the **fraud triangle** has to be satisfied. Three factors must be present, as shown in Figure 8-5.

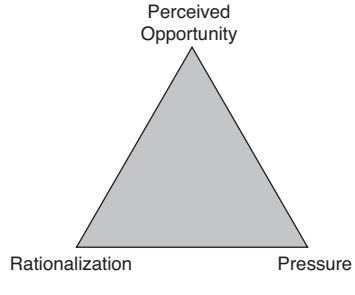

Figure 8-5 The fraud triangle, identifying the three preconditions for someone to commit fraud.

The first factor is **perceived opportunity.** The fraudster needs to have a way to override the antifraud controls or at least be confident enough that he has the capabilities to do so. An opportunity has to be present. A simple example is that the fraudster is left

alone in a room where the object of desire is situated. If there is no electronic surveillance, an opportunity presents itself. The second factor is **pressure.** This can be either from inside of the organization or from the outside. Something has to drive the fraudster. Some of the pressure can also stem from financial needs that the fraudster has. The third factor is **rationalization.** It is interesting to know that rationalization has to be present already before the crime takes place. The fraudster does not view himself as a criminal. He therefore has to justify his misdeeds before he ever commits them.

How does this materialize in a person's behavior, and how is this going to help us with detecting fraud? We need to look for opportunity. After-hours activities, for example, are important to watch. We can additionally monitor for pressure situations. For example, you could monitor a salesperson who did not make his or her quota, someone who was put on a performance improvement plan (PIP), someone who has not received a raise, and so forth. All these indicators are important when we are monitoring for fraud cases.

Fraud-Detection Solutions

The market has identified fraud as a problem, and a number of companies provide solutions to help address it. Interestingly enough, no universal antifraud solution is available that covers all types of fraud. Each type of problem has its own solution. And even among these solutions, there does not seem to be a common theme. They do not rely on one specific way to detect fraud, such as reviewing audit logs. Some of the solutions help verify that provided information about an individual or a company is accurate. Others monitor the user of a service closely and make sure that the probability of impersonation is low. Yet others statistically analyze data to find irregularities or anomalies, such as irregularly high revenues per employee.

This means that we have to come up with different analysis methods on a case-by-case basis.

FRAUD DETECTION: EXAMPLE

A while ago, I received a phone call from my credit card company, and they told me that they had blocked my card because of concurrent transactions happening in different geographic locations in the United States. It was physically not possible for me to be in both those places at the same time. This is a fairly common heuristic that credit card companies use to detect fraudulent transactions. A similar situation is associated with user accounts. If the same user account logs in from two different subnets in a short period of time, there is probably something wrong.

Doing such an analysis can be done in various ways. The following is a sample application log entry that shows access to my Web portal:

```
217.139.17.162 - ram [21/Nov/2006:02:26:00 +0100]
"GET / HTTP/1.0" 403 2898 "-" "-"
```

For each of the IP addresses in the log file, we have to determine its geographic location, which we can do based on a method similar to the one introduced in Chapter 4, "From Data to Graphs." The difference is the library I am going to use here. We need to retrieve not the country name, but the actual longitude and latitude of the IP address location:

```
cat access_log | perl -naF/.-/ -M'Geo::IP::PurePerl'
- e '$geo=Geo::IP::PurePerl->new("GeoLiteCity.dat", GEOIP_STANDARD);
my (undef,undef,undef,undef,undef,undef,$long,$lat)=
$geo->get_city_record($F[0]);
print "$F[0],$long,$lat\n";'
```

With the list of IP addresses, longitudes, and latitudes, we can then generate an XML file that can be read by the Google Maps API. The output format we need is the following:

```
<markers>
  <line color="#008800" width="2">
    <point lat="$lat" lng="$lng"/>
    <point lat="47.368527" lng="8.538503"/>
  </line>
  <line color="="#008800" width="2">
    <point lat="$lat2" lng="$lng2"/>
    <point lat="47.368527" lng="8.538503"/>
  </line>
  ...
</markers>
```

Each record in our log needs to generate an entry like this. I hard-coded the location of the second point. It is the location of my Web server that stays fixed. We can switch up the color of the individual polygon lines to match what we need them.

I leave it up to the reader to build a little script that translates the CSV output from the first command into an XML format like the preceding one. The next step

is then to find whether a single IP address is using more than one user name. This is fairly simple:

```
less access_log | perl -pe 's/\.\d+ /\. /' |
awk '{print $1, $3}' | sort | uniq |
awk '{print $2}' | sort | uniq -c |
grep -v "\s*1 "
```

The Perl command summarizes the data by class-C network. The crazy sequence of sorts first looks at unique IP address: user pairs. Then the IP address is left off, and the command checks whether there was an instance where two IP addresses from different class-C networks used the same user. The `grep` command filters for only violations. If this sequence of commands shows any output, there was a concurrent access of the same user from different subnets. You can then use this information to color the edges in the graph from before. Sample output looks like the map in Figure 8-6. The red edges indicate access with the same user name but from different subnets. Be careful when flagging malicious activities with this approach. If you are analyzing data that ranges over a longer period of time, you might have users who traveled and now access the systems from a different location, resulting in an IP address from a different network.

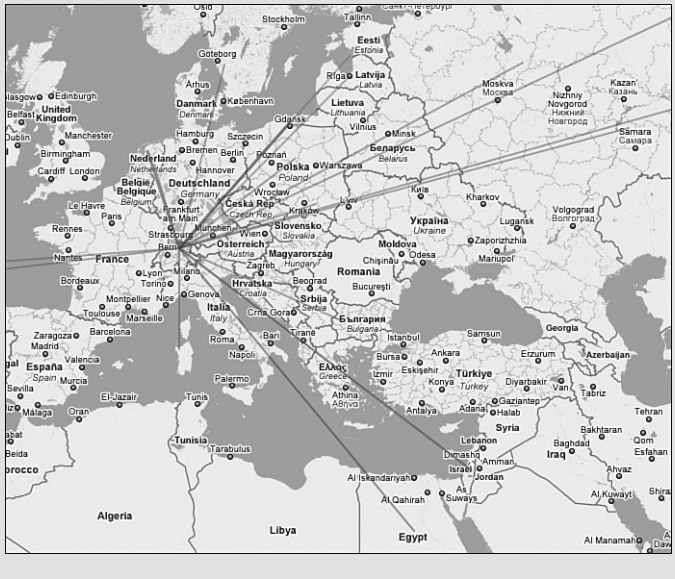

Figure 8-6 Map that shows access of a website that is located in Zurich, Switzerland. Red lines indicate concurrent access of the same user name but from different subnets, which could indicate stolen user IDs. (This figure appears in the full-color insert located in the middle of the book.)

To generate the map in Figure 8-6 through the Google Maps APIs, you need to use the JavaScript sample code from http://econym.googlepages.com/example_map7.htm. Save the source file on your machine and change the filename from `example4.xml` to the name you gave to the XML file generated earlier. Then open the HTML file on your machine and you should see the map you just defined.

SABOTAGE

Sabotage is not just a costly crime for the victim, but also one that is hard to fix. Operations might be impacted for extended periods of time and the effects will be felt for a long time. The following list is an attempt to classify different types of sabotage:

- **Information destruction,** such as deletion of files, databases, programs, backup tapes, or planting a logic bomb
- **Denial of service,** such as modifying passwords and user accounts, crashing systems, and cutting cables
- **Theft,** such as stealing computers, backups, and so forth
- **Harming organizations or individuals,** such as defacing websites, posting customer credit card data, modifying system logs to frame innocent people, and modifying a system or data to present embarrassing information

Interestingly enough, *fewer than half* (according to various studies) of sabotage cases involve sophisticated technical means. This would suggest that sabotage should be fairly easy to detect, at least in terms it being a relatively frequent occurrence.

Unfortunately there are no sabotage detection tools. The types of crimes are too different for one tool to address them all. The only category of sabotage that has created a software market early on is network-based denial-of-service (DoS) attacks. There is a fair collection of tools that help with detecting DoS attacks. They even mitigate them through different means, such as black hole routing or setting up firewall rules to block the offending traffic. There are challenges associated with these mitigation approaches. Using a firewall rule to block DoS traffic that clogs your Internet connection, for example, is useless. You have to filter before the traffic comes close to your corporate firewall. The only way to mitigate the problem is to collaborate with your ISP to block the attack closer to the attack origin.

Other types of DoS can be detected by monitoring tools. The simplest form of such a monitoring is ping. You ping machines to see whether they are available. A more sophisticated approach would be to establish connections to all the available services and make

sure they are functional. There are tools, some in the open source space, that implement this functionality (see, for example, www.nagios.org). This helps guarantee that services are available and functioning correctly.

Other sabotage cases can be detected with file-integrity monitoring tools. These tools can be used to monitor critical files on machines. Each time a critical file gets modified, the tool can report the change. This is useful for detecting changes in system configuration files or cron jobs on critical machines. This can be an addition of new cron jobs or a change to an existing one. An open source solution for file integrity monitoring is the advanced intrusion detection environment (AIDE).[3]

The problem with these monitoring tools is that system administrators have access to the tool's configurations. It is fairly easy for them to alter the configuration to conceal their sabotage activities. A solution to this is to centrally collect configuration changes and make sure separation of duties is implemented with regard to alerting data. There should be no way for a system administrator to change a configuration change record. An additional problem in a lot of sabotage cases is the absence of electronic trails. Even where there are trails, no tools exist to mine the logs for sabotage instances.

SABOTAGE: EXAMPLE

One way for a system administrator to execute a sabotage attack is to plant a cron job that will execute a malicious command at some point in the future. The following is a simple script that can be used to check how many cron jobs are scheduled on a list of machines. The list of machines can be stored in a file called `hosts.list`:

```
for host in `cat hosts.list`; do
  count=`ssh root@$host 'find /etc \( -name crontab
    -o -name 'cron.\*' \) -exec grep -P "^\d" {}\; | wc -l' `
  echo `date +%D`,$host,$count >> cron.watch
done
```

The script goes through the list of hosts and uses SSH to execute a find command on each of them. The find command tries to find all cron jobs located in the `/etc` directory. For each of the files, it extracts the number of actual cron jobs

[3] http://sourceforge.net/projects/aide

scheduled. The result is then written to a file called `cron.watch`. Running this script will generate a file that looks something like this:

```
04/17/08,vidar,4
04/17/08,thor,4
04/18/08,vidar,4
04/18/08,thor,6
```

I have two machines, vidar and thor. You can see that on April 18 two additional cron jobs were added to the machine. This might be okay, but it might not be. Figure 8-7 shows the deltas of the number of cron jobs found for each day. You can see that on January 8 two new jobs were added to all the machines under surveillance. Given this kind of finding, you would now go ahead and investigate this phenomenon.

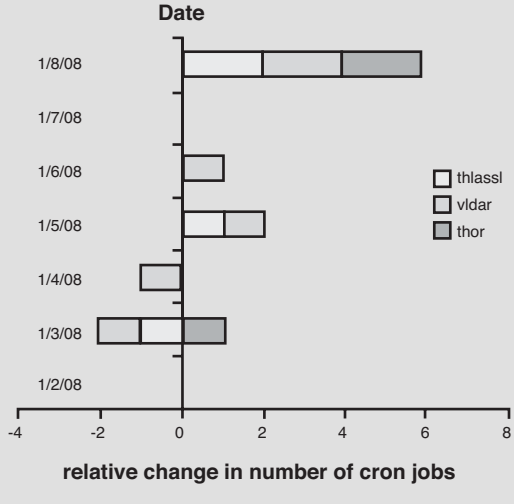

Figure 8-7 For each day, this chart shows the number of added or deleted cron jobs per machine.

The graph in Figure 8-7 was generated in Microsoft Excel. You need to build an additional column into your data series, which is the difference between the current number of cron jobs and the number from the previous day. Then use a stacked bar chart, where each series is one of the hosts.

WHO ARE THE MALICIOUS INSIDERS?

To better understand insider crimes, we should take a look at who is committing them. Some of the facts we want to understand are the following:

- Who are potential malicious insiders in our organization?
- What are these people's motivations?

By exploring the answers to these questions, we will gain some insight into how to detect insider crimes. The insight will, it is hoped, help us intervene and decrease the probability of an employee turning into a malicious insider altogether.

Different studies have been conducted to answer the questions just posed and other similar questions.[4] These studies take past cases of insider abuse and analyze them to answer these kinds of questions.

Across all three types of insider crimes, no single profile of malicious insiders characterizes them all. We have to look at the three crimes individually.

INFORMATION THEFT

Studies have found that a majority of the subjects who stole information were either former employees of that company or people who were on their way out. Generally, they were leaving of their own accord and not because they were fired. This is going to be an important fact for our approach to detecting malicious insiders. We might want to have a close look at employees who have turned in their resignation. But more on that later. There are three types of information thieves:

- People who are financially motivated
- Employees who are about to start a new job, even within their own company
- Employees who are looking for revenge

Employees who leave to start a new job often feel entitled to the information they are taking. They do not realize that they are committing a crime. Revenge materializes itself in various ways. For example, someone could leak embarrassing information about the company itself or certain individuals working there.

It is also important to know that most of the thieves used their access privileges to obtain the data, in a lot of cases in conjunction with compromised accounts. The vast

[4] See "Common Sense Guide to Prevention and Detection of Insider Threats" (www.us-cert.gov/reading_room/prevent_detect_insiderthreat0504.pdf).

majority stole information from within the workplace, which is another great indicator of what we have to look out for to detect these malicious insiders. All these data points will be of great help, as soon we are going to develop a methodology for detecting information thieves.

FRAUDSTER

Moving on to fraud, what is the persona of someone involved in fraud? The majority of fraudsters are current employees. They use their own accounts and access privileges to commit the fraud. No specific group of employees is more prone to commit fraud than others, which suggests that we need to monitor all employees to detect fraudulent activities. Interestingly enough, only a very small percentage of criminals exhibit financial needs, although fraud in general is financially motivated. A lot of fraudsters had either system administration privileges or at least privileged access to the systems they attacked. Only a minority used technically sophisticated methods, such as exploiting vulnerabilities of the systems. Same as in the information theft category, most fraudsters committed the crime from within the company workplace.

SABOTEUR

What about saboteurs? They are generally former employees. They generally act out of revenge for some negative events. Some of them are looking for revenge because they were fired. Others might not have gotten their bonus paid out, and others are upset about getting a new boss. This means that we might want to look for negative events and the people affected by them. In most cases, the saboteurs used to have, or still do have, privileged or even system administration access to the systems attacked. However, at the time of their crime, they generally do not have authorized access to the systems, which means that they are using compromised accounts or backdoor accounts established at an earlier time. Some saboteurs use technically sophisticated means to execute the attack. This includes things such as scripts to trigger logic bombs, for example. In a fair number of cases, saboteurs take steps to prepare for the attack. This is good for us to know. We should be looking out for anomalous types of activities that could indicate preparatory actions to set up for sabotage. Some of the precursors that should be investigated are employees who suddenly start working irregular hours or start hanging out in chat rooms during work. These types of activities indicate that the employee is not committed to the job.

All these factors give us a great starting point for thinking about how to detect insider threat.

A DETECTION FRAMEWORK FOR MALICIOUS INSIDERS

After looking at three different examples of how malicious insiders can be detected and studying the malicious insider personae, we now understand the insider threat space well enough to define a proactive approach to detecting malicious insiders.

The problem we already encountered a little earlier is that no devices identify a malicious insider when she or he enters the building or becomes active on the network. Similarly, no warning devices can warn us about malicious insider activities. The traces of malicious insiders are hidden all over throughout different applications and log files.

To help us identify malicious insiders, we need to know that they do not just decide one day that they will commit a crime and then execute it. There is a reconnaissance or preparation phase where the malicious insider slowly prepares his or her crime. We should go back to the insider personae discussed earlier. One key fact we learned from the different malicious insider types is that we have to not only monitor our current employees, but also need to have an eye on former employees. You might think that's an easy task. Former employees should not have access to any resources after they leave the company. I hope you are right. If your company is really good about its termination processes, all the user accounts will be disabled and all access will be revoked. However, reality shows that this is rarely the case. You are probably well off monitoring former employee accounts! Keep in mind that you need to take care not just of accounts directly associated with that person, but also shared accounts that the employee had access to.

It would be ideal if you not only monitored former employees but also employees who are on their way out, employees who show signs of resigning soon, and employees who show "suspicious" behavior. These groups of employees are especially prone to committing a crime.

In the following subsections, I first discuss a simple process to detect malicious insiders. The process discussion starts with the topic of **precursors.** Precursors are activities that we should be monitoring to detect malicious insiders before they become active. You will see that precursors are most useful for detecting saboteurs. Following the precursor discussion, we look at **watch lists,** and following that, we look at how to improve the detection process.

Visualization is going to play a role in almost each step of the detection process. Before we can start the discussion of the detection process, we need to look at the concept of precursors, which are the building blocks for the entire process.

PRECURSORS

To detect malicious insiders, we have to monitor user behavior and activities. The activities can be collected from applications, network traces, physical security devices such as

badge readers or IP cameras, and so on. Some activities can be used to immediately flag someone as a malicious insider (for example, a regular employee who installs a key logger on a colleague's machine). This means that a subset of all the activities that users execute can be considered precursors:

> A **precursor** is an activity that, when observed, flags the associated user as a potential malicious insider. Each precursor can be assigned a score, which reflects the extent to which the precursor classifies someone as a malicious insider. Precursors are a way to differentiate "good" users from the ones that are up to no good and might potentially turn into the malicious insider.

Precursors are based on user activity. Qualifying an activity as a precursor can be done through one of three possible processes:

1. *Signature matching:* A predefined keyword, a specific action, and so on is defined, and activities are matched against the signatures.
2. *Policy monitoring:* The activity is compared against a set policy. A violation constitutes a precursor. Examples include people accessing certain data, using specific applications, surfing forbidden websites, and so forth.
3. *Anomaly detection:* First normal behavior is identified to create a profile or a baseline. Once established, the profile is used to assess behavior and determine whether it is "abnormal" or falls inside the boundaries of being "normal."

Table 8-3 lists a short sample of precursors. These precursors help flag users or machines that should be looked at a bit more closely. You can find a more comprehensive list of precursors at the end of this chapter.

Table 8-3 Sample Precursors and Their Detection Method

Precursor	Detection Method
Accessing job websites such as Monster.com	Signature matching
Salesperson accessing patent filings	Policy monitoring
Printing files with *resumé* in the filename	Signature matching
Sending emails to a large group of recipients outside of the company	Anomaly detection

Many precursors can be used to flag potential malicious insiders. The problem is that a lot of the tools that can identify some of this behavior have a high rate of false positives and therefore need to be complemented with some other way of confirming the precursor activity. Note that not all the precursors manifest themselves in electronic trails or

log records. That is unfortunate.[5] A lot of precursors are hard to record. Things such as the family situation, personal problems of employees, and so forth do not leave an electronic trail that we could consume. I don't discuss any of these precursors, although experience shows that these are useful ones. A way to still capture these precursors is to use a simple application where the situation of each employee can be registered. One last point about monitoring people's activities: A number of countries have laws that prohibit the recording of user activity without a prior notice to the user or a legal case made. As an employee, I wish more countries had privacy laws covering these issues.

How do we compile a list of precursors? One way is to go through all your log files. Take each of them and identify activity that suggests suspicious behavior. Try to elicit input from as many people as possible. Application owners are, most of the time, the experts in understanding their log files. They generally have a fair understanding of what precursor you should be looking for to detect suspicious activity. You will have to get fairly creative. Think out of the box. It is not always the case that the log entry screams at you, asking for you to look at it. Things such as printing resumés can be on the list of precursors. This does not mean that someone printing a resumé is automatically a malicious insider. What you identify are precursors for users in general. In conjunction with other behavior, the fact that someone printed a resumé might help flag him or her as a malicious insider. It is likely that someone who prints a resumé is on his or her way out of the company. Along with printing the resumé, the person might be starting to collect proprietary information to bring along. Of course, this does not have to be the case. It might be that a manager was printing a resumé before interviewing someone. Or the person is leaving the company on good terms. However, some precursors are much more indicative of suspicious behavior. A salesperson should never access a company's patent filings. And an engineer should not access the financial records.

Assigning Scores to Precursors

Each precursor needs a score indicating the "badness" or level of suspiciousness of the activity. The higher the score, the more likely it is that the machine or user is malicious and needs to be looked into. A low score is associated with behavior that is common and could be seen any time. In most cases, a low score indicates normal behavior. Table 8-4 shows the same list of precursors from before, but now adds a score to each of the entries.

[5] Fortunate from a privacy perspective! Unfortunate, if we put our log analysis hat on.

Table 8-4 Precursor List with a "Badness" Scores

Precursor	Score
Accessing job websites such as Monster.com	1
Salesperson accessing patent filings	10
Printing files with *resumé* in the filename	3
Sending emails to 50 or more recipients outside of the company	5

The scores in Table 8-4 are determined by answering three main questions:

1. Is the precursor activity "normal" for *any* user? Do certain users trigger this precursor as part of their regular workday? For example, "customer data access" is a completely normal activity for certain people in an organization, but not for others.
2. If someone who was not allowed to execute a certain action were to trigger this precursor, what would be the impact? How bad would that be? What could happen, for example, if a salesperson has access to the source code server?
3. What is the confidence level associated with the precursor? Is the way of flagging activity with this precursor absolutely foolproof or is there a certain false-positive rate associated? If a precursor relies on NIDS events, for example, what's the rate of false positives for that signature?

Table 8-5 shows a sample decision matrix for how to assign the scores to the precursors based on the preceding three questions. The valid range of scores is set to be 0 to 10, where a 10 constitutes a full compromise. A different range of scores could have been used, too (for example, 0 to 100). However, think about what the difference between a score of 98 and 97. Unless you are defining a sophisticated scoring schema, you won't be able to discern a difference between the two.

Table 8-5 Decision Matrix for Assigning Scores to Precursors

Impact If Executed Outside of Role?	Normal for Some Roles?	False Positives?	Score
Full compromise	No	No	10
Full compromise	No	Yes	9
Direct link to potential compromise	No	No	8

continues

Table 8-5 Decision Matrix for Assigning Scores to Precursors *(continued)*

Impact If Executed Outside of Role?	Normal for Some Roles?	False Positives?	Score
Direct link to potential compromise	No	Yes	7
Direct link to potential compromise	Yes	No	6
Direct link to potential compromise	Yes	Yes	5
Could be setup for a compromise	Yes	No	4
Could be setup for a compromise	Yes (or no)	Yes	3
Minimal or none	Yes	No	2
Minimal or none	Yes (or no)	Yes	1

An impact of **full compromise** means that this precursor identifies a malicious insider act and not just a preparation activity. The compromise could happen in any of the three insider threat categories. It could be that an act of sabotage, fraud, or information leak is going to happen or already happened. A **direct link to a potential compromise** means that the user behind this activity is well on his way to setting up for the crime. It has not quite happened yet, but the stage is set. If you can tolerate a higher rate of false positives, you can assign higher scores to the individual precursors. The table illustrates only an example mapping.

INSIDER-DETECTION PROCESS

The ultimate goal of an insider-detection process is to detect or identify malicious insiders before they commit a crime. The previous discussion has shown that that is too ambitious. However, we can still try to use heuristics and the knowledge of previously published insider crime cases to define an insider-detection process. One important requirement for the detection process is extensibility. If we gather new intelligence, gain new knowledge, we need to be able to easily incorporate it into the detection process.

Without further ado, the four steps of the **insider-detection process** are as follows:

1. Apply precursors.
2. Visualize the insider candidate list.
3. Apply the threshold filter.
4. Tune the precursor list.

The output of this process is a visual representation of users or machines that are potential malicious insiders. These entities have raised suspicion by triggering precursors.

Visualization is not only used to visualize the final outcome but also in some of the earlier steps of the process:

- Some precursors rely on visualization to flag anomalous behavior.
- Visualization is used to identify anomalous behavior relative to a user's role in the company.
- Visualization is used to quickly communicate and represent the output of the detection process.

The following sections discuss each of the insider-detection process steps.

1. Apply Precursors

The goal of the first step of the insider-detection process is to compile a list of actors (i.e., users or machines) that were flagged through precursors. This **candidate list** will then be used in the next step to generate a visual representation of all the actors and their precursor activities. There are two ways to compile the list of actors. For each precursor, follow these steps:

1. A product deployed in the environment can explicitly detect the precursor behavior. For example, a NIDS can detect peer-to-peer connections that violate the information sharing policy. Violators can be placed on the list.
2. If no device exists that could detect some of the precursors, or you don't have such a product deployed, you need log files that contain enough information to detect the precursor-related activity. Detecting people printing resumés, for example, is generally not an action that is flagged by any security device. However, it is fairly simple to parse the printer logs to extract users printing such documents.

Figure 8-8 illustrates the first step in the insider-detection process. Either a precursor is found in a log file or a specialized monitoring device flags it. The specialized device could be monitoring log files itself. However, often these devices operate on other data sources, such as raw packet captures or they directly monitor systems and applications through APIs. As soon as a precursor is identified, the user or the machine—the actor—will be put on the insider candidate list. This list will be the working list that we are going to use to find malicious insiders.

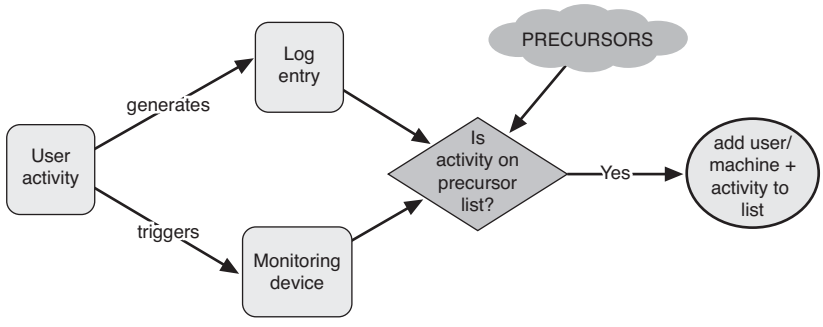

Figure 8-8 Precursors can be detected by either a specialized product or by applying filters to raw log files.

While analyzing log files to identify activity on the precursor list, we will run into the problem that the log file does not contain enough information. For example, a database access log shows, per database table, the users who accessed it. One of the precursors tries to identify engineers poking around in financial tables. There is not enough information in the database log file to detect this activity. The log file is not going to contain information about the type of database table and the role of the user. We need additional meta information about the database tables and the roles of the users. For each table, we need to know whether it is a financial table or not. In addition, we need a list of all the users from the engineering department.

Make sure this type of information is available for each of the precursors. In fact, before starting to apply precursors to log files, get familiar with the logs and determine whether additional information is needed to evaluate the precursors.

A lot of times, determining whether a precursor is present in a log file is as simple as applying a regular expression to a log file. If it matches, the precursor is present. However, sometimes, this is not enough. In some cases, a correlation engine can help here. These systems apply advanced techniques to log entries and can express more complex relationships between not just simple events but among multiple ones. An example of an open source correlation engine is the simple event correlator (SEC).[6] A representative commercial grade system is ArcSight.[7]

After this first step of identifying all the instances where a precursor was triggered, we need to filter the list of users and machines that were flagged and sort out the "good" candidates from the "bad" ones. To do so, we are going to use a visual approach.

[6] www.estpak.ee/~risto/sec

[7] www.arcsight.com

2. Visualize the Insider Candidate List

Visualizing the insider candidate list is the main step in the insider-detection process. The goal of this step is to produce a graph that helps identify a set of likely malicious insiders and weed out unlikely ones. Instead of using visualization to process and analyze the list of insider candidates generated in the preceding step, we could use a deterministic approach, especially since we have a score associated with all the attributes. Although this would be a valid solution, visualization offers some unique capabilities that we want to exploit:

- Visualization is nondeterministic. A script does exactly what it is programmed to do. Visualization provides the viewer the possibility of making his or her own interpretations and lets him or her find and discover new patterns.
- Machine-based detection methods are largely static, whereas the human visual system is dynamic and can easily adapt to the ever-changing techniques used by malicious insiders.
- Visualization brings up questions you have not thought about, instead of just answering existing questions.

Let's step back for a second to think about what exactly we want to accomplish. This is a list of three goals we are seeking to accomplish:

- Get a feeling for groups of users. Which users exhibit approximately the same behavior? Do any users not fall into any groups? Are there outliers in otherwise consistent clusters? Why?
- Adding information about a user's role helps determine whether a precursor is indeed a warning sign. For example, printing a resumé is part of an HR person's regular job and not generally a sign of a malicious insider activity.
- Find users with high scores (i.e., users who triggered a significant number of precursors).

Identifying Groups of Users with Similar Behavior

The first goal we can accomplish by visualizing the users and their precursor behavior. Figure 8-9 shows an example. The graph draws users with similar agendas in proximity. This is an effect of using link graphs, which try to find an optimal placement of the nodes. This helps grouping users of similar roles together.

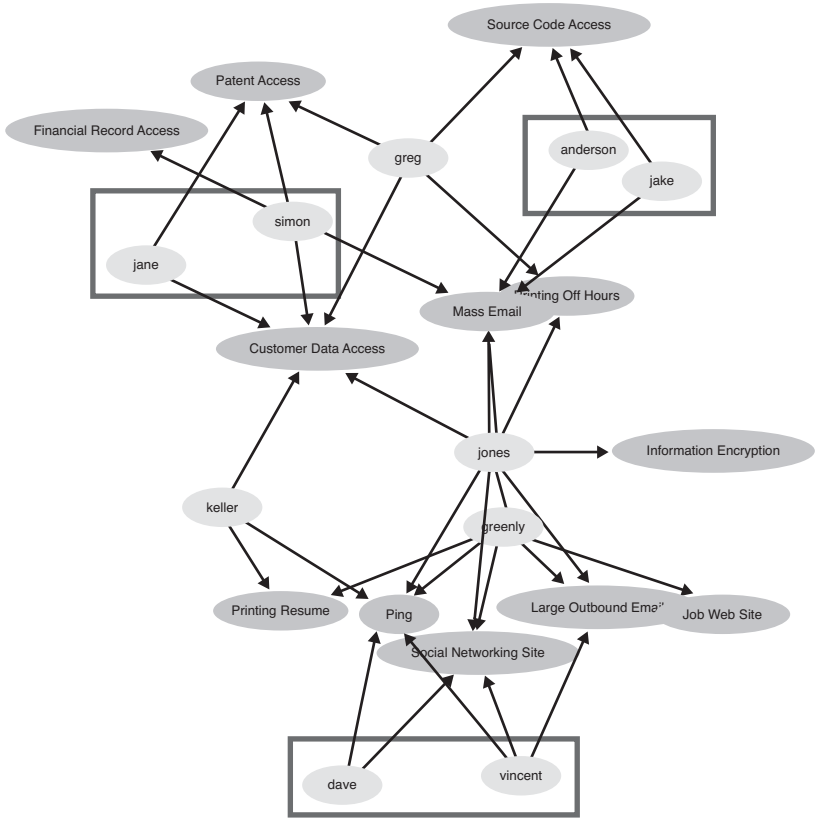

Figure 8-9 This graph shows how users can be grouped into roles by analyzing their precursor behavior. Users of the same role exhibit similar precursor behavior.

I used rectangles in Figure 8-9 to annotate users who exhibit similar behavior. It does not necessarily mean that these users have a similar job role, but their observed activities suggests so. Note that the grouping was done loosely. The users do not exhibit exactly the same behavior. Nevertheless, it seems that the users' roles are related. For example, both Jane and Simon are accessing patent information, as well as customer data. Hopefully, both of these users are working in legal and have a real need to do so. Greg, on the other hand, seems to be accessing patent data and customer data. In addition, he is printing off-hours and is trying to access source code. This is not a good combination of activi-

ties. If the attempted accessing of source code were not in the mix, this user could also belong to the same group as Simon and Jane, but not with the attempted accessing of source code. The graph helps quickly identify such instances.

Also note that not all the user nodes in Figure 8-9 are annotated with a rectangle. Jones and Greenly most likely do not belong to the same role, even though they are drawn very closely to each other. Jones is exhibiting much more activity than Greenly, and Greenly distinguishes himself by the additional customer data access. These are subtleties that need to be taken into account when grouping the users together. However, the visual representation greatly helps finding users that might be related.

This type of clustering or grouping helps us accomplish the first goal; we have a way to identify groups of users.

Augmenting the Analysis with User Roles

If, in addition to the precursor logs, we have an actual list of roles for each user, we can augment our graph with that information. This will ease our analysis of user groups and provide an additional data point. Instead of guessing what role various users belong to by comparing them, their roles are explicitly encoded in the graph. Figure 8-10 shows the same graph as before, but this time the roles of the users are encoded as the color of the user's nodes.

Coloring the nodes based on the role of the users simplifies our precursor analysis immensely. We can see that two of the groups that we found earlier, the ones covering the users on the top of the graph, indeed coincide with real user roles (engineering and legal). However, we also see that Greg is an engineer, yet he exhibits behavior that is unique to legal workers. This was not obvious in the previous graph.

The bottom of the graph shows a different picture. We identified the two users, Vincent and Dave, to be of the same group (see Figure 8-9). This is, however, not the case, as Figure 8-10 shows. The single finding in Figure 8-10 that gives the most cause for concern is related to the marketing people. They are all over the map in terms of triggering a number of precursors. Looking at which exact precursors they trigger, however, mitigates this concern; they all seem fairly benign. None of them really poses a significant threat.

So far, we are only identifying groups of users, without making any statement about how bad those activities really were. The next step is going to address this objective by starting to use the precursor scores.

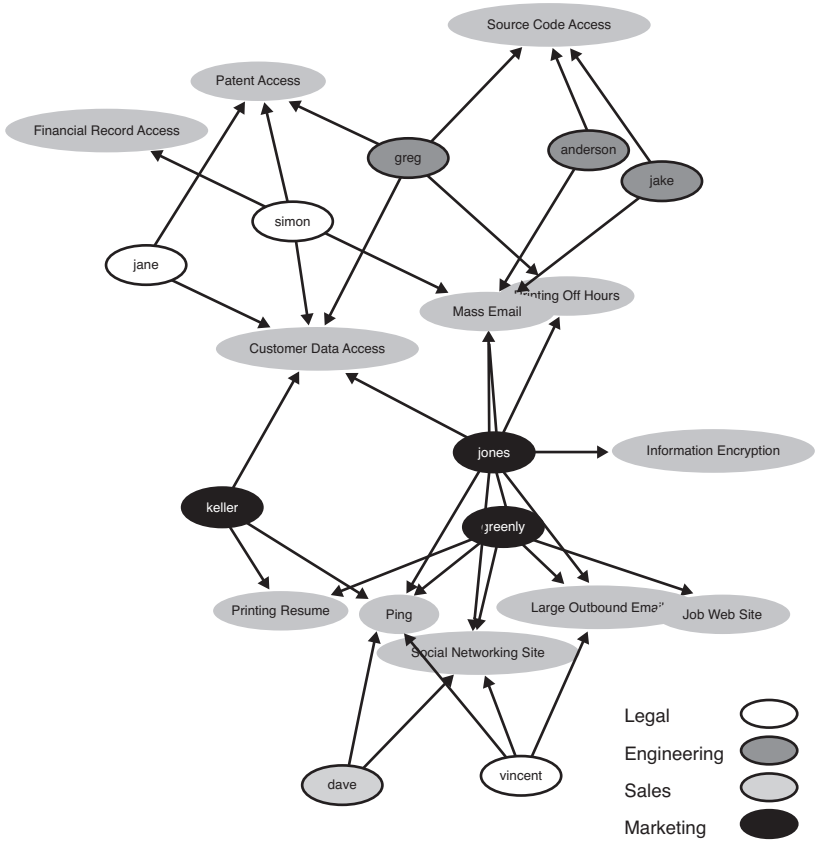

Figure 8-10 Users and their precursor activity. User nodes are colored by using the user's role. This way the identification of outliers becomes much easier.

Identifying Users with High Scores

It is now time to make use of the precursor scores to identify users who triggered many of the precursors. We can add an additional property to the graph to encode the precursor scores. Figure 8-11 uses a new set of data than the previous graphs have used. In the new graph, the size of the user nodes (the rectangles) encodes the cumulative score of a user. Note that the size of the circles is kept the same for all nodes and does not encode any information.

In Figure 8-11, malicious precursors are colored black. These are all the ones with a score of 5 or higher. Precursors that could just as well be benign behavior are gray.

Malicious precursors should never be seen. However, the problem of false positives prevents us from taking immediate action if a user triggers them. A user who triggers multiple malicious precursors increases the certainty of something being wrong.

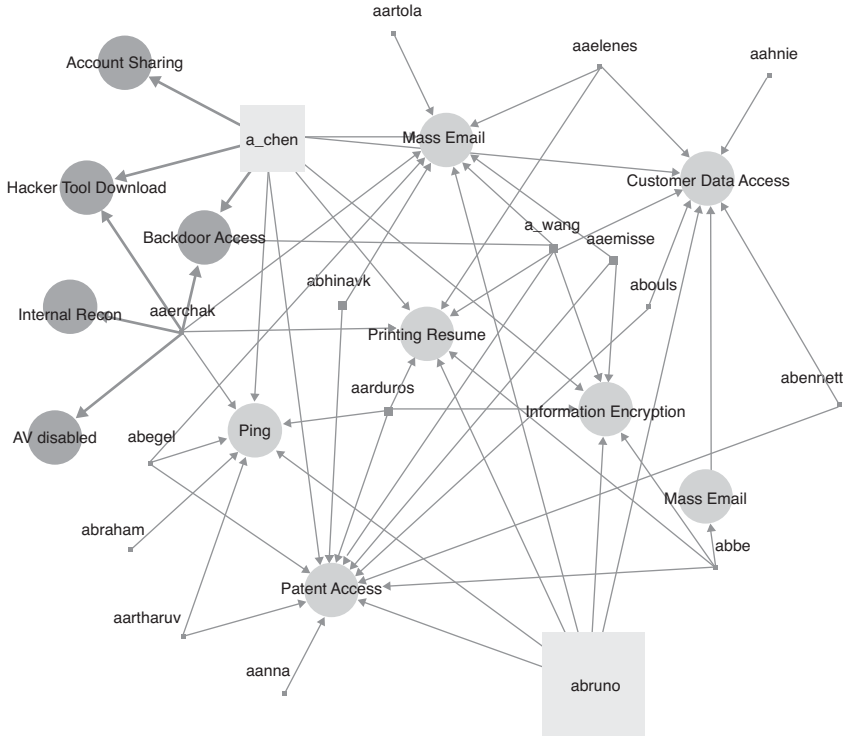

Figure 8-11 Users and their precursor activity. Malicious precursors are colored in black, and precursors that could as well indicate benign activities are represented in gray. The size of the rectangles encodes the score a user accumulated.

In Figure 8-11, we notice two users right away: abruno and a_chen. They seem to be triggering a lot of precursors. It looks like abruno is the worst person in the mix and deserves some closer attention. Is that right? Take a look at the precursors abruno triggered. They are all benign ones (gray circles). Not a single one of them is a malicious one (black circles). On the other hand, a_chen triggered some of the malicious precursors, and so did aaerchak, who shows up as a small node compared to the others—the reason for this being that the user only triggered its precursors once, whereas other users have triggered some of the precursors many times. This is why aaerchak accumulated a

relatively small score. Having a closer look at the individual precursors, it seems that aaerchak should be investigated closer.

This finding is fairly interesting. It seems that someone who triggers a few malicious precursors can almost fly below the radar. Maybe a link graph is not the best way to visualize this type of data?

What happens if we take the same information and try another type of visualization? In Figure 8-12, you can see how the same information looks in a treemap. The users are on the top of the hierarchy, followed by the precursor they triggered. Color encodes the score of the precursor. The darker the squares, the higher the precursor score. The size of the boxes encodes the number of times a certain precursor was observed.

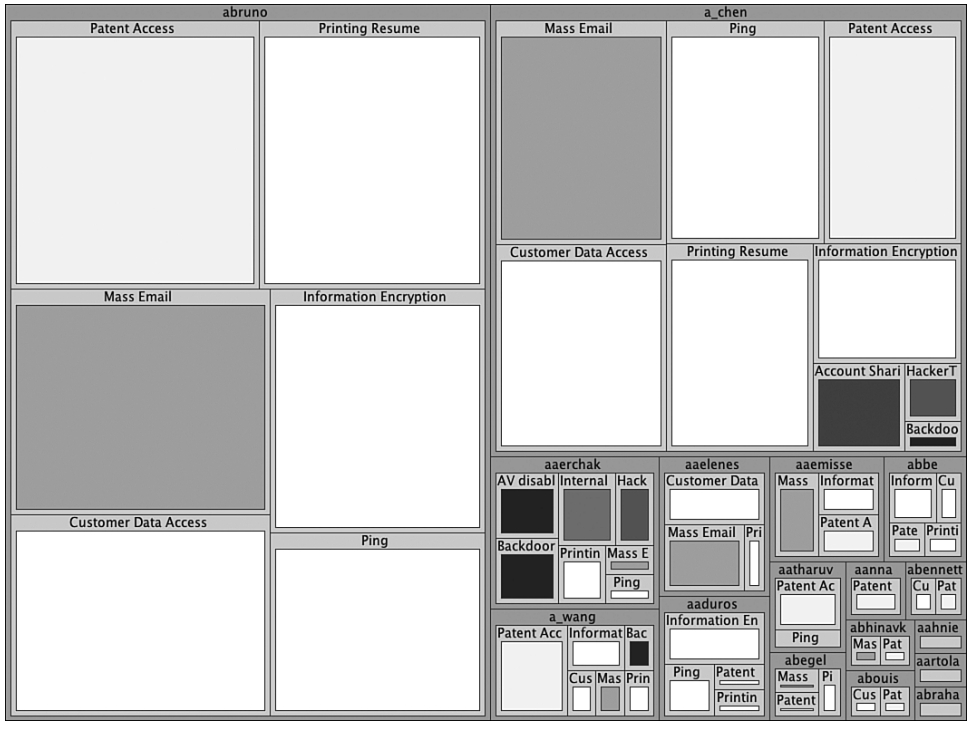

Figure 8-12 Precursor activity encoded in a treemap. The hierarchy is User > Precursor. The color represents the score that a certain activity represents, and the size of the boxes is the amount of times a certain behavior showed up.

We are interested in big dark boxes. With this treemap, we can quickly identify the same group of potentially malicious users who we already found with the link graph. Abruno shows a lot of low-score behavior, whereas aaerchak is noteworthy for triggering significant and potentially malicious precursors. Compared with the link graph (Figure 8-11), the treemap makes it easier to spot the users who triggered high-score precursors.

This concludes Step 2, the visualization of the insider candidate list. We will later see how some of the shortcomings of our visualization methods can be addressed through an improved detection process. The next step in the insider-detection process is going to help us manage larger insider candidate lists.

3. Apply the Threshold Filter

The graphs that visualize insider candidate lists can grow large quickly. We have to find some filtering techniques to narrow down the list of users to focus on.

Filtering the candidate graph can be done in various ways. The most obvious is to focus on the size of the user nodes. We can define a threshold to remove small nodes. Small nodes represent users who did not accumulate a high score. To determine a good threshold, various approaches can be applied. One is to choose a trial-and-error method, whereby we slowly increase the threshold to filter out more and more nodes until the graph represents a small and workable set of users. *Small* in this case is relative. It can be any size that you are comfortable working with. A simple way to see what happens if such a threshold filter is applied is to look at a bar chart that shows the score plotted for each user. Figure 8-13 shows a bar chart for the example we looked at before.

Based on the distribution of users and their scores, you can now make a decision where to set the threshold. Be cautious, though. The bar chart in Figure 8-13 shows a long tail. This would suggest we set the threshold somewhere above 50. This would then leave us with only the two users with the highest scores. However, as we have seen in the previous graphs, the next user down (aaerchak) is worth having in the picture, too!

Another way of setting a threshold is to sort the values and always pick the top *n* users, where *n* can be any number. This is can be dangerous, too. If the "*n* plus 1" user is one who behaves in an anomalous fashion or even maliciously, you just missed him. However, it seems that the scoring process should be able to take care of that and not the filtering process. We will investigate what we can do to improve the scoring process to circumvent these problems.

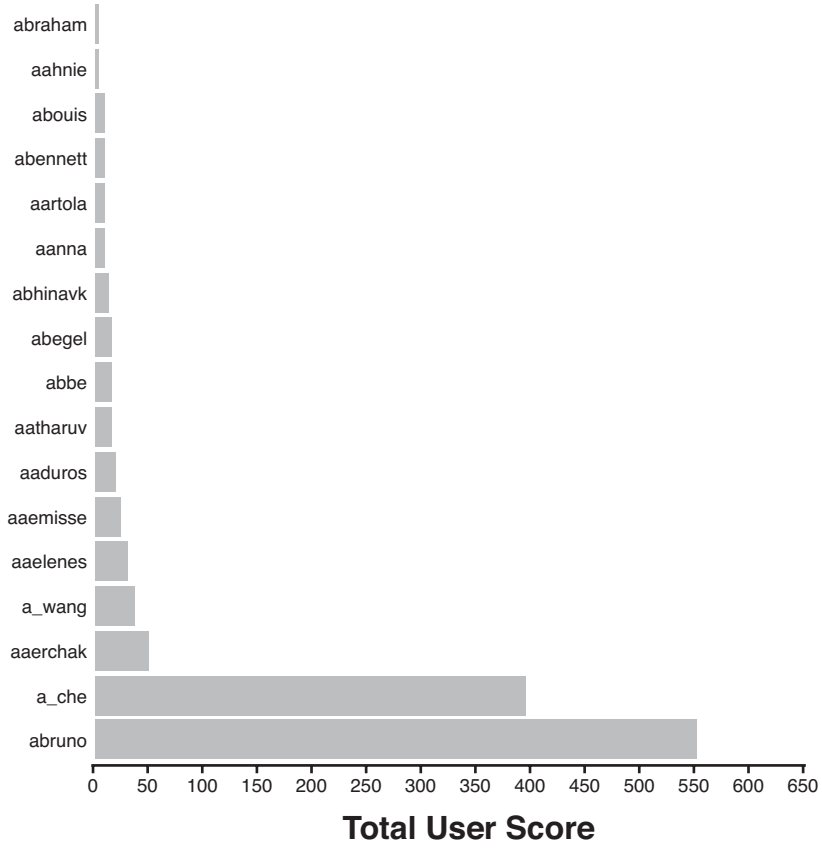

Figure 8-13 Bar chart showing the accumulated score per user.

As soon as we determine how to filter our list of insider candidates, we can use the new list to generate a graph for only the remaining users. To continue on the example from before, if we filter the graph, we end up with Figure 8-14. The threshold was set to include only the top three users, which was the outcome of the previous discussions.

This step almost concludes the insider-detection process. The only step we have left is tuning the precursor list, which is discussed in the next section.

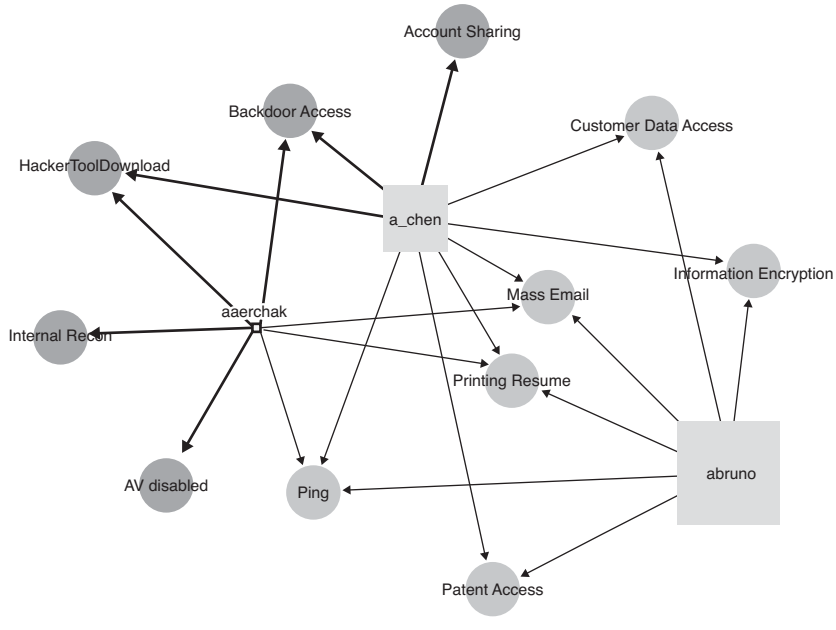

Figure 8-14 A link graph of malicious insider activity, filtered to show only users who accumulated a high score.

4. Tune the Precursor List

The last step in the insider-detection process involves tuning the precursors and their scores. The goal of this tuning process is to arrive at a score for each precursor that helps to clearly identify a malicious insider, but at the same time it should not flag benign actors.

In the previous steps, we recognized that four different situations can apply to a precursor:

1. A precursor turns out to flag benign activity all the time. *Remove it.*
2. A precursor is missing. Some activity is not caught. *Define a new precursor.*
3. A user triggers a precursor although the activity is absolutely benign for this user. *Define an exception for this user.*
4. A score for a precursor is either too high or too low. *Adjust the score.*

After you have made these adjustments, rerun the detection process and verify the result. You will realize that this is an iterative process. Adaptation and tuning is necessary to get better and better results.

When going through tuning of precursors, make sure you are also looking at the graph without the threshold filter already applied. The reason for doing so is that you will miss the people flying below the radar, the ones with a low score. Precursors could be hidden below the threshold.

The process of changing precursor scores starts with finding precursors that seem to either escalate users too quickly or with realizing that some activity is more important than the score reflects. Adjust the score for the precursor and pay attention to why the change seems necessary. Possibly, it is not just this one precursor that needs adjustment, but an entire class of precursors should be adjusted.

A more sophisticated approach to tuning precursor scores is the application of an **expert system.** The idea is to use a system that can tune the scores based on user feedback. It learns from the user (the expert) who interacts with the system. The expert makes a judgment about a user's score based on all the information present: the precursor, the user's role, and so on. The system learns from the feedback and adjusts the scores of the precursors. Many algorithms and approaches are documented in the literature. For more information about this, refer to the work of Maloof and Stephens.[8]

SUMMARY OF INSIDER-DETECTION PROCESS

We looked at a lot of things in this section. It is time to stop and summarize what we have covered. Figure 8-15 graphically summarizes the insider-detection process. We started out with defining a precursor list. As mentioned earlier, a fairly large list of precursors can be found at the end of this chapter. After defining the precursors, we discussed how to assign a score to them. Once this was done, we put the precursors in action. We have seen that devices flag behavior based on the precursors or the precursors can be used to analyze behavior via log files. The application of the precursors resulted in a list of insider candidates that we then visualized. Using the candidate graph and some meta information about the users, we conducted a role-based analysis of the user activities. To reduce the size of the graph and eliminate the "not so bad" users from the analysis, we discussed how to apply thresholds. The last step before closing the process circle was the analysis of the output, which directly led to a discussion about tuning the precursor list and scores.

[8] "ELICIT: A System for Detecting Insiders Who Violate Need-to-Know," by Marcus Maloof and Gregory Stephens, RAID 2007.

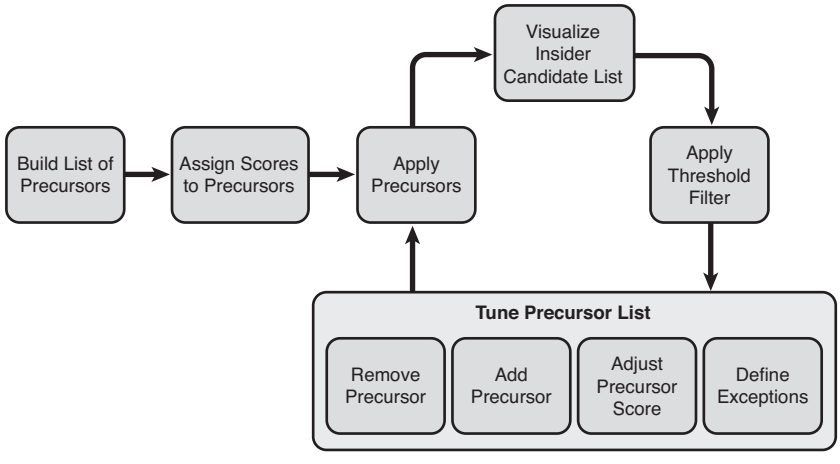

Figure 8-15 Flow diagram of the insider-detection process.

Tuning the precursors will be a work in progress, and precursors will have to be added and removed constantly over time. New precursors will be necessary for many reasons: new detection technologies, new threats, new users, new types of users, and so on. When a single round of the insider-detection process has been completed, the process starts over again.

Before we take a look at how this process could be improved and address some of its shortcomings, let's take a look at an example insider case.

Insider-Detection Process at Work

This section considers the insider-detection process again, but this time with a more practical, hands-on approach. Each step of the process outlines how the data has to be prepared and how the graphs can be generated.

To start the insider-detection process, we define a set of precursors. You can find a sample set of precursors at the end of this chapter. The list is a good starting point for our sample use-case. The precursors in the list are already scored based on the methodology introduced earlier. We therefore do not have to go through the scoring process.

The detection process starts with applying the precursors to the log files at hand. The following is an example of how to do this. Assume we want to find users who printed resumés. On a Windows print server, you can use the printer logs to check which files have been printed. A sample log entry looks like this:

```
Event Type: Information
Event Source: Print
Event Category: None
Event ID: 10
Date: 8/5/2002
Time: 8:32:39 AM
User: Domainname\User1
Computer: Server
Description:
Document 208, Microsoft Word - resume.doc owned by ram was printed on HP DeskJet 550C
Printer/User1/Session 12 via port
\\sa-01\Sales\TS012. Size in bytes: 128212; pages printed: 3
```

You can collect these log entries from Windows print servers in the application log. To access the logs on the Windows machine, use the `wmic` command (see Chapter 2). The simplest way to find users who printed resumes is to `grep` for the word *resumé* in multiple languages:

```
wmic ntevent where "LogFile='Application' and EventCode='10'" GET format:csv |
grep –i "(resume|lebenslauf|cv)"
```

Extract the user from the event (for example, by using the `awk` command) and store records that look as follows in a `precursor.list` file. It contains the user, the precursor name, the role of the user, and the precursor score:

```
user,precursor,role,precursor score
ram,printing resume,engineering,1
```

The roles can be retrieved from either a directory service, such as Active Directory, or from a static mapping between users and their groups, depending on what data you have available. You get the precursor scores from the precursor list, which you can find at the end of the chapter.

Repeat the preceding step for each of the precursors you are monitoring. Do not forget to automate his process, for example, via a `cron` job. This will enable you to monitor the precursors on a regular basis.

When this step is completed and we have a `precursor.list` file, we are ready to generate a first graph. I am using AfterGlow (see Chapter 9, "Data Visualization Tools") to create the candidate link graph. To decorate the nodes correctly, create a property file like the following one and save it as `ithreat.properties`:

```
 1 # Variables
 2 variable=@violation=("Backdoor Access", "HackerTool Download", "AV
   disabled", "Account Sharing", "Internal Recon", "Password Sharing",
   "Classification Breach", "Unauthorized Web Site", "Locked Account",
   "Hacker Site");
 3 # shape
 4 shape.source=box
 5 # size
 6 maxnodesize=1.5
 7 sum.target=0                # do not accumulate target node size
 8 sum.source=1                # source node sizes are cumulative
 9 size.source=$fields[3]      # the fourth column indicates the size
10 size=0.5
11 # color
12 color.target="royalblue3" if (grep(/^\Q$fields[1]\E$/,@violation));
13 color.target="skyblue1"
14 color.source="#b2e2e2" if ($fields[2] eq "Contractor")
15 color.source="#66c2a4" if ($fields[2] eq "Privileged User")
16 color.source="#b30000" if ($fields[2] eq "Terminated")
17 color.source="#6dc200" if ($fields[2] eq "Legal")
18 1 color.source="#edf8fb"
```

The configuration file first defines a variable (`violation`) that contains the names of the precursors that denote a significant violation. This variable is then used to determine the color of the target nodes in line 15. If the second column (`$fields[1]`), the precursor, shows up in the list of violations, the node is drawn darker than otherwise. On line 7, the shape of the source nodes (the users) is defined to be a box. Lines 9 to 13 define the size of the nodes. First the maximum node size is set to 1.5. Then we instruct AfterGlow to accumulate the size of the source nodes. For each time that the source node (i.e., the user) shows up in the log file, we add the score to determine the size of the node. In line 12, we define that the fourth column (`$fields[3]`), the precursor score, is used to define the source node's size. In the last few lines, we define the color of the nodes. Color is determined by the third column in the data, the user's roles. Each role has its own color. Add your own roles here, if you are using different ones.

After defining the properties file, run AfterGlow with the following command:

```
cat precursor.list | afterglow -t -c ithreat.properties |
neato -Tgif -o candidate.gif
```

The result of running this command on a sample data set is shown in Figure 8-16. You can see that there are some large green boxes. These are users in the legal department who triggered a number of precursors. It seems that we can more or less visually group the different user groups in this graph. Around the Source Code Access precursor, there are a number of engineers and contractors. The legal people have the Patent Access in common and so forth. It would now be useful if we could find outliers or users who behave atypically for their role. In this particular graph, it is hard to find such outliers. Not a single user seems to pop out or draw attention. Users are too entangled and are sharing a lot of the precursors among the roles. We had to serially scan the graph to find possible violations of separation of duties or users who acted atypically for their roles. Your graph might not suffer from this, and you might see clear clusters of users and outliers thereof.

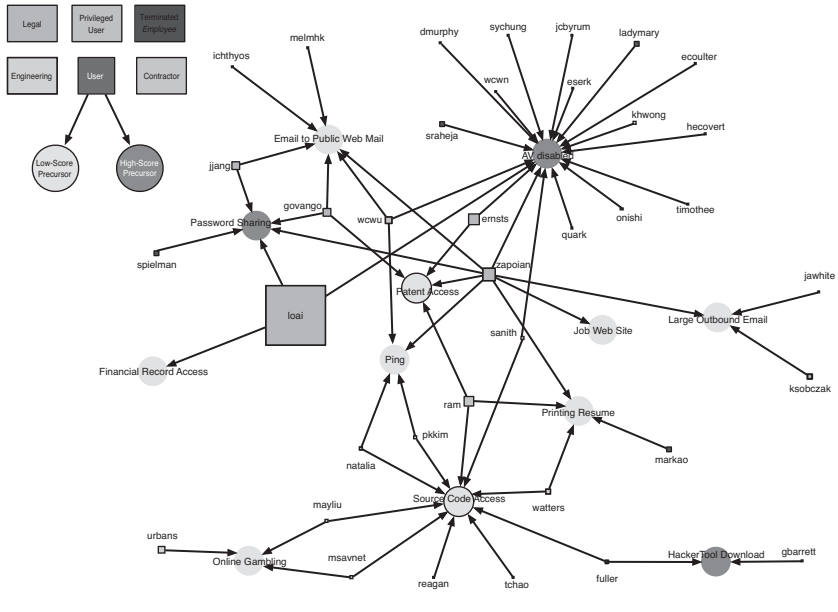

Figure 8-16 Link graph showing users and the precursors they triggered. Color is used to encode the role of the users on the source nodes and the severity of the precursors on the target nodes. The total score a user accumulated determines the size of the user nodes. (This figure appears in the full-color insert located in the middle of the book.)

One of the interesting things in Figure 8-16 is the number of people who disabled their antivirus software (AV disabled). This does not have to be malicious. In fact, it is probably a sign that the AV software gets in the way of a number of employees' regular

work and they disabled it. However, from a security standpoint, this is a warning sign, and someone needs to follow up on it. Another interesting finding is that there seem to be a handful of users who are on the Terminated Employees list, and they show up around various precursors. These are a couple of initial findings that call for action.
To gather some more information about the actors, we can use the treemap visualization shown in Figure 8-17. To generate a treemap with the Treemap 4.1 tool, we have to generate a TM3 file (see Chapter 9) first. The first step is to manually enter the header lines:

```
COUNT          User     Precursor  Role      Score
INTEGER        STRING   STRING     STRING    INTEGER
```

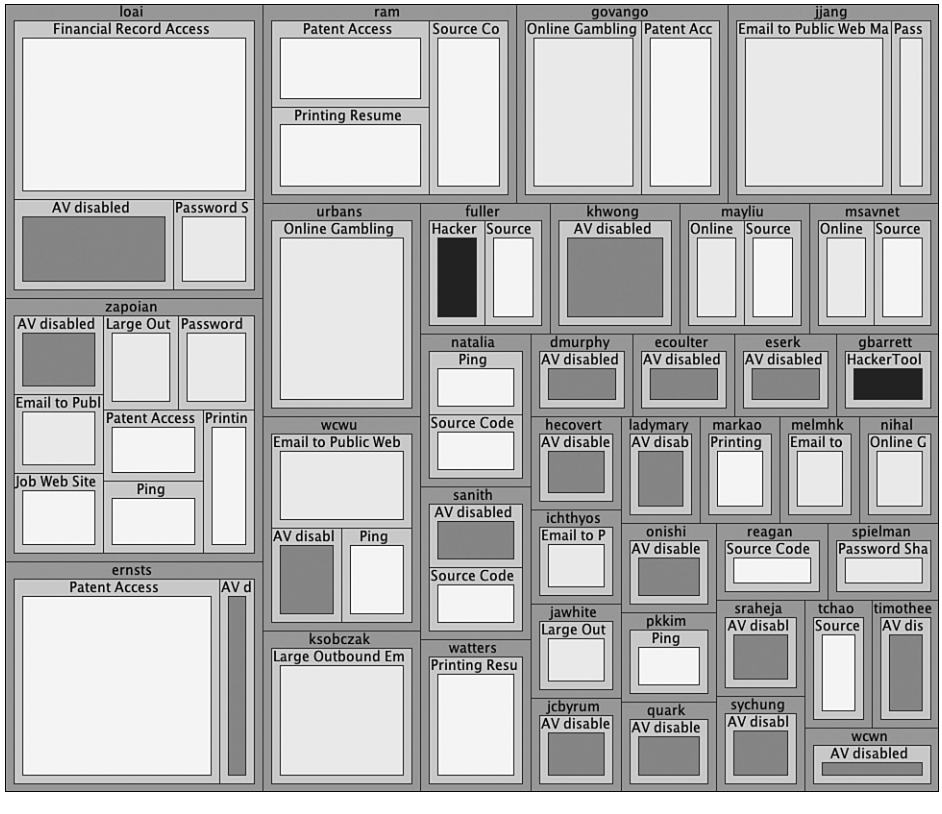

Figure 8-17 This treemap visualizes the candidate list. It uses a configuration hierarchy of User > Precursor. The size of the boxes is determined by the number of precursors run up, and the color encodes the precursor score. (This figure appears in the full-color insert located in the middle of the book.)

Make sure tabs separate the tokens. Then use the following command to translate the CSV file, `precursor.list`, into the TM3 format:

```
cat precursor.list | sed -e 's/,/	/g' | sort | uniq -c |
perl -pe 's/^\s*//, s/ /	/' >> precursor.tm3
```

Note the two tabs in the command. The output then looks like this:

```
COUNT	User	Precursor	Role	Score
INTEGER	STRING	STRING	STRING	INTEGER
1	dmurphy	AV disabled	Terminated	4
1	ecoulter	AV disabled	Contractor	4
```

Load the file with Treemap 4.1 and configure it to use a hierarchy of User > Precursor. Assign the total score to be used for the size and the precursor score to be the color of the boxes.

Figure 8-17 has the interesting property that two nodes pop out immediately based on their dark color. These users trigger only a small number of precursors that do not seem too bad. The fact that someone downloaded a hacker tool is interesting to know, but by itself is probably not a huge concern.

Instead of looking just for dark boxes, we can look for users who triggered a lot of precursors. They show up at the top left of the map. It seems that some users triggered interesting precursors and combinations thereof. User ram is interesting. He did not just access patents and source code but also printed resumés. If this is an engineer, he should not be accessing patents. If he is a lawyer, he should not be accessing source code. The combination of these three precursors could be a sign of an employee who is leaving the company. He is printing his resumé and at the same time is browsing for company confidential information to take with him: patents and source code. A closer investigation should be initiated.

In line with the discussion at the end of the introduction to the insider-detection process, we realize that there are some deficiencies to the process as it is. We need to improve a few things, which we address in the following section.

IMPROVED INSIDER-DETECTION PROCESS

We have identified a few weaknesses in the insider-detection process as it stands. Namely, if a user triggers the same precursor multiple times, he gets escalated very quickly, even if the precursor was quite benign. The process also does not incorporate any external

information, such as a list of terminated employees. Lastly, we would like to treat users differently based on their role. For example, an administrator should be allowed to run a vulnerability scanner, whereas a salesperson has no business of doing so. To address these issues, we are going to explore the concept of watch lists. Following the discussion of watch lists, we examine how we can add the watch lists to our insider-detection process. This leads us to trying to prevent users' scores from shooting through the roof because of a single precursor that is triggered multiple times. The solution to this problem is grouping precursors into buckets.

WATCH LISTS

The insider-detection process used a fairly simple way of scoring activities and flagging potential malicious insiders. A malicious insider was identified by adding scores based on the precursors an actor triggered until a certain threshold was exceeded. We have seen that there are some problems associated with the scoring scheme. Following is a list of features we should include in the detection process to make it more powerful and address some of the problems identified earlier:

- External and dynamic intelligence should feed into the detection process (for example, watch lists defined by HR). These lists can be used to flag groups of employees and monitor them more closely than others (for example, terminated employees or employees on a performance improvement plan).
- Certain user roles should be monitored all the time (for example, administrators or privileged user accounts). These users have a lot of power over the system, and it is comparatively easy for them to circumvent detection systems. We also need to know who these users are to define exceptions for a lot of the precursors; they trigger a lot of them throughout their workday (e.g., scanning the internal network, using password crackers).

Watch lists, which are nothing other than lists of actors (users or machines), can be used to address these objectives. They incorporate external data into the insider-detection process and can be used to give special treatment to subsets of users. The detection process can be extended to utilize watch lists in a couple of ways:

- Precursors can be restricted to monitor users on specific watch lists only.
- Precursor scores can be adjusted based on membership in specific watch lists.

Table 8-6 is an overview of a sample set of watch lists. These lists are fairly generic ones and should be extended with ones specific for your environment.

Table 8-6 Watch Lists and the Activities/Precursors That Are Relevant and That Should Be Monitored for the Users on These Lists

Watch List	Monitored Activity / Precursors
Terminated employees	All activity executed by accounts that are associated with people who left the company are a warning sign. In reality, there will be automated scripts or similar activity that is going to show up. However, all of this activity needs to be eliminated.
Privileged users / administrators	Users with administrator accounts or privileged user accounts for any system should be monitored closely, especially changes to systems, such as adding new users, giving more access rights to users, changing critical settings on systems, and so forth. All of these can be used to prepare for attacks or install backdoor access. This type of activity should have a change request ticket associated with it to justify the action. In addition, even regular activity should be monitored for administrator or root accounts. Generic activity should not be executed with privileged accounts!
New hires	Don't trust new hires from the first day of employment. They need to prove themselves and earn trust. Monitor their activities and be more stringent about what they are allowed to do. Practically, this means that the precursor score for these users is going to be slightly higher than for employees who have been with the company for a while. The first couple of weeks of employment can also be used to build up a usage profile or "normal" behavior pattern for a user.
Contractors	Contractors are generally given access to the corporate resources, such as the network. Under certain circumstances they are even given privileged access to systems and applications. It is therefore crucial to monitor these users extra carefully. Verify the hours of operation and make sure the accounts are terminated when their work concludes.
Finance users	"Finance users" is just one example of a set of watch lists that can be instantiated to monitor access to specific resources. There is a finite set of users who should be accessing financial system or other specific resources. The better these groups can be defined and monitored, the smaller the chance for abuse.

The detection process needs to be extended to utilize the watch lists. We can use them to define exceptions for precursors and restrict certain precursors to actors on specific lists only. The second way of using them is to use the watch lists to adjust the precursor scores. Table 8-7 shows a way to adjust the scores for precursors depending on what watch list a user belongs to.

Table 8-7 Adjustment Table Showing, for a Set of Precursors, How Their Score Is Adjusted Based on the Actor Belonging to a Watch List

Watch List	Precursor	Score Adjustment	Final Score
Terminated employee	Any activity	+6	<10
Privileged users	Access information outside of need to know	+2	5
	Use of anonymizing proxy	+1	4
	Deleting audit logs	+1	5
	Creation of backdoor account	+n	10
	Disabling of anti virus or security software	+1	5
	Changes to critical files	+2	5
	User account sharing	+1	8
	Direct data table access	+2	8
New hires	Any Precursor	+2	N
Contractors	Not complying with configuration guidelines	+2	5
	Web traffic bypassing proxy	+3	7
Owner of critical data	Unauthorized encryption of information	+2	5
	Failure to create backup	+1	5
	Unauthorized information transfer	+4	7
	User account sharing	+1	8
	Anomalous email activity	+2	4
	Storage device attachment	+3	7
	Traffic to suspicious countries	+3	7
IT security staff	Not complying with configuration guidelines	+2	5
	Physical anomalies	+1	10
Everyone except IT security staff	Download of password crackers	+2	5
	Downloading of hacker tools	+1	8
	Promiscuous network interface	+1	5

Table 8-7 uses the list of precursors that is printed at the end of this chapter. Not all of them are incorporated in the table because a lot of precursor scores are not impacted even though the actor belongs to a watch list. To understand the adjustment table, let's look at the first row. If activity is detected from a terminated employee, the score of that precursor is increased by 6 points. However, the score should be kept below 10. This adjustment has the effect of increasing very low scores. Very low scores are not assigned an automatic 10. Only precursors with an original score between 4 and 9 will be bumped up to a 10.

Figure 8-18 visually summarizes how scores are adjusted for an actor based on belonging to one of the watch lists. Per the watch list, the treemap shows which precursors are getting adjusted. The darker the color of a box, the bigger the adjustment of the precursor score.

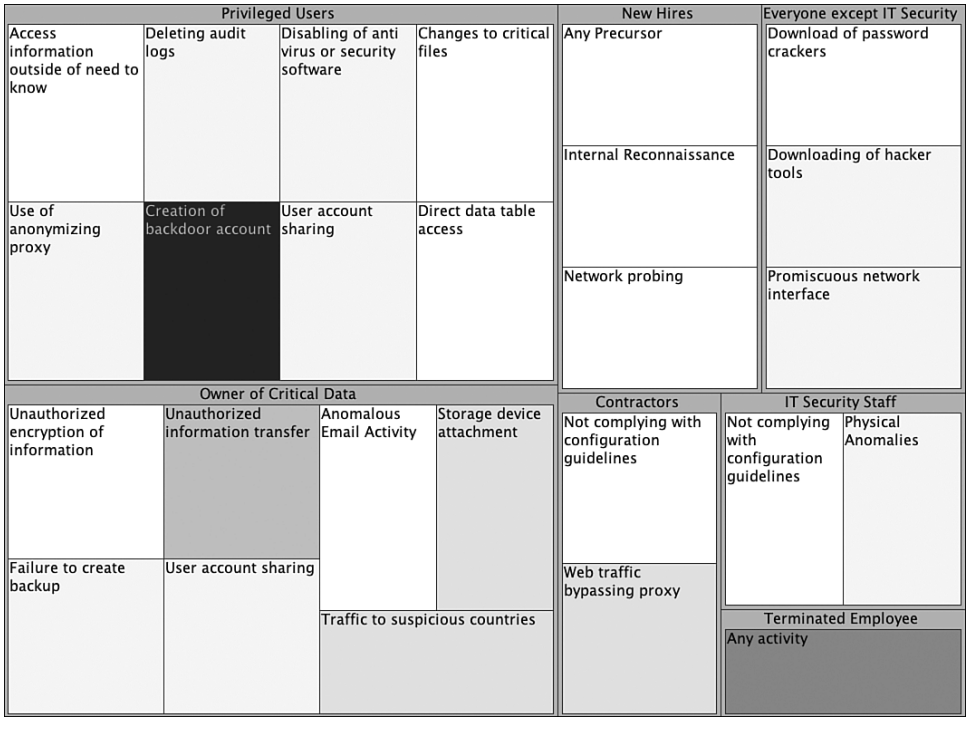

Figure 8-18 Treemap showing how the watch lists increases the scores for specific precursors. Color encodes the amount of the change; the darker the color, the bigger the adjustment.

Watch lists need to be kept up-to-date. Ideally, populating them is part of a regular IT process. For example, the new hire list should be populated as soon as IT creates the user accounts for a new employee. The terminated employee list should get an update from HR as soon as an employee leaves the company.

ADDING WATCH LISTS TO THE INSIDER-DETECTION PROCESS

The insider-detection process introduced in this chapter has some deficiencies, including the following:

- If a user triggers the same precursor multiple times, he gets escalated very quickly, even if the precursor was quite benign. Printing many documents off-hours, for example, quickly flags an employee as a malicious insider.
- Depending on which watch lists an actor belongs to, we would like to use a different set of precursors. Changes to files, for example, should be monitored mainly for privileged users and not for the entire staff.
- Depending on the watch list, different scores should be used for the same precursor. A new hire conducting an internal reconnaissance should be rated higher than if an administrator did the same.

To address some of these deficiencies, we can use the watch lists introduced in the preceding section. The first step is to adjust the precursor scores according to Table 8-7. In addition, we can use the watch lists to color the user nodes. The result of incorporating these new data inputs into the insider-detection process is shown in Figure 8-19.
If we look at the graph in Figure 8-19, we can see a few things:

- Color is used to highlight users who are on a watch list.
- The size of the user nodes has changed according to Table 8-7.

The net effect for Figure 8-19 is that a new user suddenly draws attention to himself: abouis. This user is on the terminated employee list and therefore his score was bumped up significantly. You might notice that aaerchak's node has grown, too. This means that this user deserves more attention than if he were a regular one. This is a privileged user, and he should therefore be monitored closely due to his extended access and capabilities.

Coloring the users according to the watch list they belong to has yet another benefit. It is suddenly possible to visually group user nodes and compare activities with the watch list attribution in mind. One thing to watch for, for example, is privileged users triggering precursors clearly outside of their domain, such as a privileged user accessing patents.

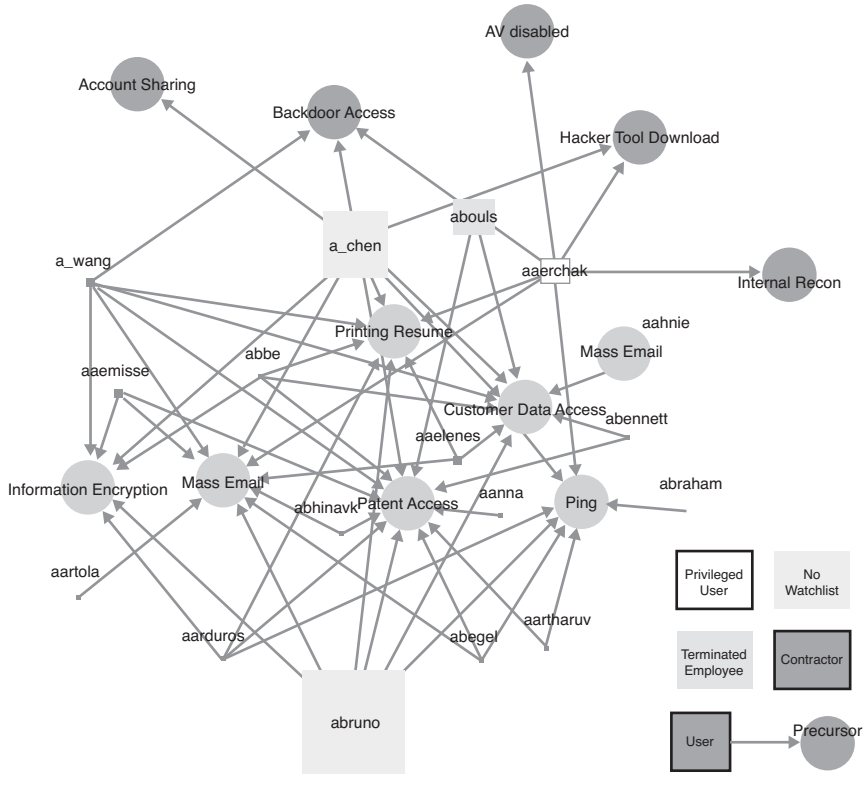

Figure 8-19 The insider candidate list shown as a link graph. The precursor nodes are colored based on how "bad" a precursor is deemed. The actor nodes are colored according to the watch list that the actor belongs to. (This figure appears in the full-color insert located in the middle of the book.)

GROUPING PRECURSORS INTO BUCKETS

The approach of using watch lists to adjust users' scores and also color nodes in the candidate graph is promising. However, we still have not addressed one of the most pressing needs we identified throughout the insider-detection process. We need a way to improve scoring. The first problem we should address is that a user who executes the same activity over and over again will keep triggering the same precursor and therefore drive his or her score higher and higher.

To address this problem, we are going to group the precursors into five buckets. Each bucket can only accumulate a maximum score of 20. This way, the maximum score that

a user can ever reach is 100. This will prevent a single precursor, or even a set of precursors, from compounding and driving the score for a user through the roof. The five buckets are the following, based on the original score of the precursors:

- Precursors representing minimal or no impact—original precursor scores of 1 and 2
- Precursors showing signs of a setup for a malicious act—original precursor scores of 3 and 4
- Precursors indicating malicious behavior, activity that is clearly not normal but under certain circumstances or for certain user roles the activity is benign (e.g., scanning internal machines is a precursor that is bad, but a system administrator scans internal machines every now and then)—original precursor scores of 5 and 6
- Malicious behavior (behavior that should *never* be seen)—original precursor scores of 7 and 8
- An insider crime—original precursor scores of 9 and 10

This way of grouping the precursors into five buckets has a nice side effect. We can define a tiered system to classify users. A user with a score between 81 and 100 is regarded as malicious with a very high likelihood of committing a crime. Given the new scoring schema, the user needs to trigger at least one precursor in group 5, which is a real malicious insider act. In the range of 61 to 80, we can be fairly certain that a user is malicious. He needs to trigger a lot of precursors and not just in the benign tier, but in some of the others, too. Figure 8-20 shows a classification of users, based on the score they accumulated. From 0 to 20, there is not much to worry about. From 21 to 60, users are on a bad track, and there seems to be something going on. From 61 to 80, an actor has executed some malicious activities and seems to be worth checking out. Actors with a score above 80 deserve some attention. There is something going on.

Figure 8-20 A tiered system to classify users based on their accumulated score.

Note that a user could trigger a precursor with a score of 9 or 10 once, which by our definition of the scores constitutes an insider crime. However, because of the rating, the overall score of the user will not trigger an alarm. Only if this user also triggers some other precursors would he be noted. Depending on the precursors, this has to be

adjusted, and these precursors have to be given a higher individual score to escalate the user more quickly. For some precursors, on the other hand, this fact can be used as a way to reduce false positives and make sure that a user was indeed malicious.

CANDIDATE GRAPH BASED ON PRECURSOR BUCKETS

The candidate link graph is not going to change much with this new way of capping the size of nodes. The only difference is that the size of the nodes is not going to increase infinitely. In cases where a maximum node size is specified, the smaller nodes will show up slightly larger due to the new scale that is applied. A way of reporting the scores per user is to use a stacked bar chart that encodes per user the scores for each of the buckets. Figure 8-21 shows an example of such a bar chart. This chart is best used for a small number of users. As soon as the list gets bigger, users need to be filtered.

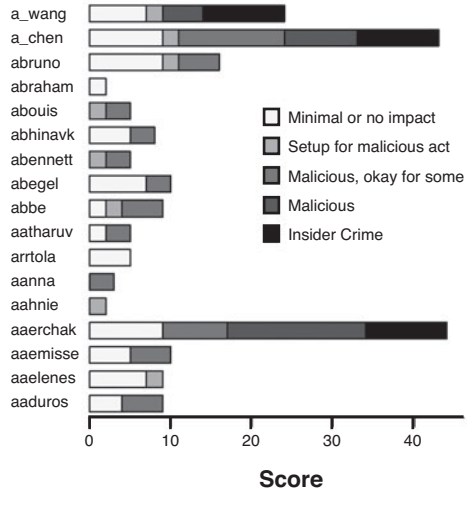

Figure 8-21 Stacked bar chart, encoding the score per precursor group for each user.

A more scalable solution is to use a treemap representation. An example is shown in Figure 8-22. The configuration hierarchy of the treemap is Watch List > User > Precursor Bucket > Precursor. What we should focus on are large, dark boxes to identify malicious insiders or users that are misbehaving. The darker the box, the higher the score of the precursor represented by the box. The size of the box represents the total score accumulated for that precursor. In other words, if the precursor has a score of 5 and a

user triggered it 6 times, the box would have to be drawn with a size of 30. However, because we allow only a maximum score of 20, the box is capped at the size of 20! Dark patches in the treemap represent users who triggered a lot of significant precursors.

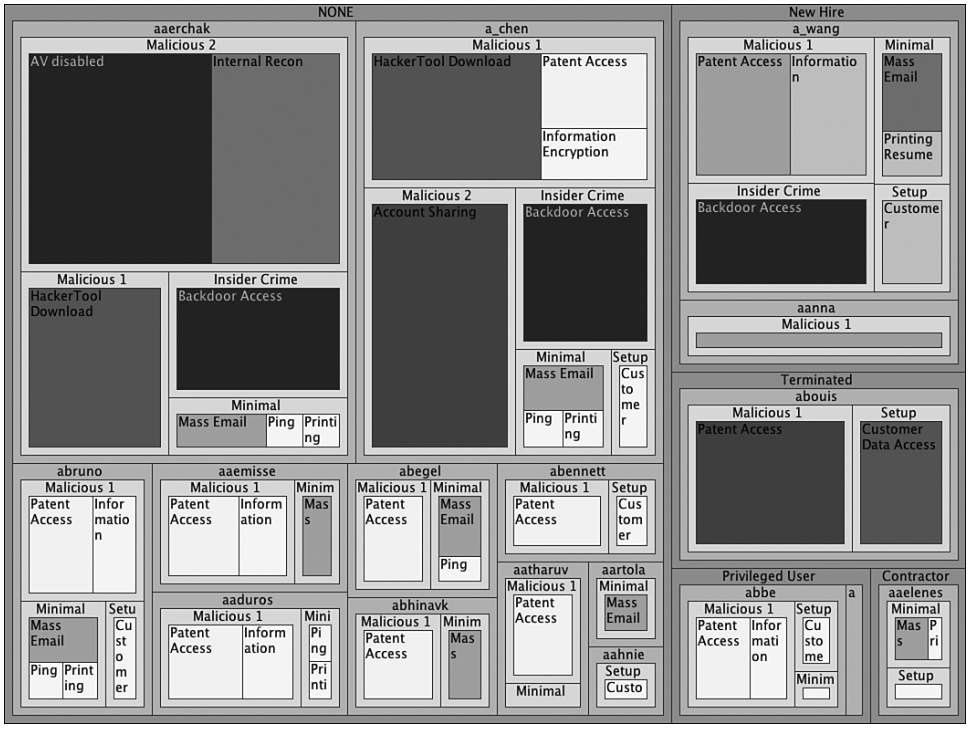

Figure 8-22 Treemap showing the accumulated score per user. The configuration hierarchy for the treemap is Watch List > User > Bucket > Precursor. The size of the boxes is based on the score a user accumulated. The color is based on the score of the individual precursor. The darker the boxes, the higher the score.

In Figure 8-22, we can quickly identify that aaerchak is one of the users we need to investigate closer. We can also see from the treemap that this user is not on a watch list. The other user that deserves attention is a_chen. He also triggered a lot of high-score precursors.

Another way to look at the treemap, instead of identifying big red patches, is to start with the watch lists. We can see some activity from users on the Terminated Employee list. The activity of those users is represented with dark patches. This is yet another set of users that deserve our attention.

This treemap should be used as a working graph. Have it generated either in real time or at least on a regular basis to review activity in your environment. By going through the insider-detection process, and specifically the tuning step, you will see that this is going to become a great tool for understanding your environment.

IMPROVED INSIDER-DETECTION PROCESS SUMMARY

With the previously discussed extensions to the insider-detection process, we can address the shortcomings of the initial version of the process. We started using two new concepts: watch lists and grouping of precursors into buckets. The watch lists added the capability to:

- Incorporate external input from, for example, HR. An example is a watch list for terminated employees.
- Limit the scope of precursors to just certain actors that are on specific watch lists.
- Adjust precursor scores based on what watch list an actor belongs to.
- Color actor nodes in the candidate graph.

In addition, we introduced buckets of precursors. Each precursor lives in one of five buckets. The maximum score a user can accumulate in a bucket is then limited. This has the effect that individual precursors can escalate an actor only to a specific limit, unlike before when a single precursor could run an actor ad infinitum.

The basic insider-detection process from before did not change. The only change is in how scores are accumulated and how the candidate graph is drawn. Let's continue on with our example from earlier and see how the extensions to the process change the visualizations.

EXTENDED INSIDER-DETECTION PROCESS AT WORK

Earlier in this chapter, we started to look at an insider case by applying the insider-detection process. At that point, we did not know about any of the more advanced techniques that we just discussed. Let's revisit that insider example and augment it with the new techniques.

Previously, in the regular insider-detection process, we had the following data elements to generate our graphs:

- *User:* The user triggering the precursors
- *User role:* The role of the user to analyze user behavior relative to their roles

- *Precursor:* The precursor triggered by the user
- *Precursor score:* The score of the precursor

The extended insider-detection process adds the following pieces of information to the analysis:

- *Watch list:* The watch list a user belongs to
- *Precursor bucket*: The bucket that the precursor falls into
- *Total user score*: The accumulative score of the user across all buckets

To generate the candidate graphs, we need to prepare our data. The central problem is the calculation of a user's total score. The calculation has to respect two properties:

- There is a maximum score that can be accumulated in each precursor bucket.
- Watch lists influence the score of precursors. They can either limit the applicability of precursors, or change the score depending on what watch list a user belongs to.

The product of our data preparation needs to be a file that contains the user, the precursor the user triggered, and the total score a user accumulated:

```
govango,Password Sharing,8
govango,Email to Public Web Mail,8
govango,Email to Public Web Mail,8
govango,Patent Access,8
natalia,Ping,1
```

Note that we need to include the total score a user accumulated in this file and not the individual score of a precursor. The reason for this is that we need to guarantee that a user does not accumulate more than a certain maximum score per precursor bucket. When visualizing the candidate list, AfterGlow would blindly add all the scores for the individual users, even beyond the maximum that is allowed per precursor bucket. This is precisely what we did before. This is why we have to deal with the calculation of the total score per user outside of AfterGlow.

How do we calculate the total score for a user? Unfortunately, no single command line could do this. It is slightly more involved than that. Let's see what a possible approach would look like.

The first step is to adjust the precursor score based on the watch lists, and then in a second step we cap the scores based on the precursor buckets. To adjust the scores per watch list, I use a set of `awk` scripts that look similar to this one:

```
awk -F, -v OFS=' '
  '/govango|peter|john/ {$4=$4+2}  # first watch list, adjust by 2
  /annen|baumann|stillhard/ {$4=$4+4} # second watch list, adjust by 4
  {print}' activity.csv > new_activity.csv
```

Each line of users represents a watch list, for which the precursor scores are adjusted. In this example, we're adding two points for the first watch list and adding four points for the second.

In addition to using the information about the users and their watch lists to adjust the scores, we store it in a file (`watchlist.csv`). The file has to list the user along with the watch list the user belongs to:

```
dmurphy,Terminated
ecoulter,Contractor
fuller,Contractor
```

After all the adjustments have been done and the watch list file has been defined, we then have to calculate the total score for a user. You need to write a little script or manually go through the data to guarantee that a user does not accumulate more than a certain maximum score per precursor bucket. I wrote a little tool to do this. You can find the tool (`capper.pl`) on this book's CD. Here is how you run it:

```
./capper.pl user_activity.csv precursor.csv > user_activity_new.csv
```

The script takes two inputs: a file with all the user activity (user, precursor) and a file with precursors and their scores. The precursor file also lists the bucket for each precursor and not just the precursor score. The `capper.pl` script does nothing other than add the score for each user and make sure that, per bucket, the user does not exceed its maximum. The output then looks as shown earlier.

Insider Candidate Link Graph

After all the data has been prepared, we are ready to define the properties for the insider candidate graph. We need to build another property file for AfterGlow to format the link graph correctly:

```
1 # Variables
2 variable=@violation=("Backdoor Access", "HackerTool Download", "AV
  disabled", "Account Sharing", "Internal Recon", "Password Sharing",
```

```
   "Classification Breach", "Unauthorized Web Site", "Locked Account",
   "Hacker Site")
 3 variable=open(FILE,"<watchlist.csv"); while(<FILE>)
    {chomp; split(/,/); $wlist{$_[0]}=$_[1]}
 4 # shape
 5 shape.source=box
 6 # size
 7 maxnodesize=1.5
 8 sum.target=0                # do not accumulate target node size
 9 sum.source=0                # source node sizes are cumulative
10 size.source=$fields[2]      # the third column indicates the size
11 size=0.5
12 # color
13 color.target="royalblue3" if (grep(/^\Q$fields[1]\E$/,@violation))
14 color.target="skyblue1"
15 color.source="#b2e2e2" if ($wlist{$fields[0]} eq "Contractor")
16 color.source="#66c2a4" if ($wlist{$fields[2]} eq "Privileged User")
17 color.source="#b30000" if ($wlist{$fields[2]} eq "Terminated")
18 color.source="#6dc200" if ($wlist{$fields[2]} eq "Legal")
19 color.source="#edf8fb"
```

The property file is similar to the one we used previously. The only big difference is the way we are using the watch list. In lines 6 and 7, we are reading the watch list from `watchlist.csv` into an internal variable. It is then used in lines 19 through 22 to check whether the user is on a specific list. Also note that we are not accumulating the score for the source nodes (line 13). We already did the calculations in our preprocessing.

Figure 8-23 shows the result of visualizing this information.

The graph in Figure 8-23 does not look much different from the one we generated with the simple insider-detection process. Notice that the user nodes are colored based on the watch list they belong to and not based on their roles (as previously). Also note that the size of all the nodes has changed. The user ioal is not the single biggest node anymore. Other users are now drawing attention to themselves, as well. Analyze the graph more closely by using the colors to see whether any of the users on a watch list are behaving strangely. Look for large, colored nodes.

This is a great starting point for tuning the precursors. In some cases, you might find that you need to define exceptions for specific groups of users. For example, you might realize that privileged users are triggering specific precursors (such as using ping) all the time. For these cases, consider defining an exception to not show those instances anymore.

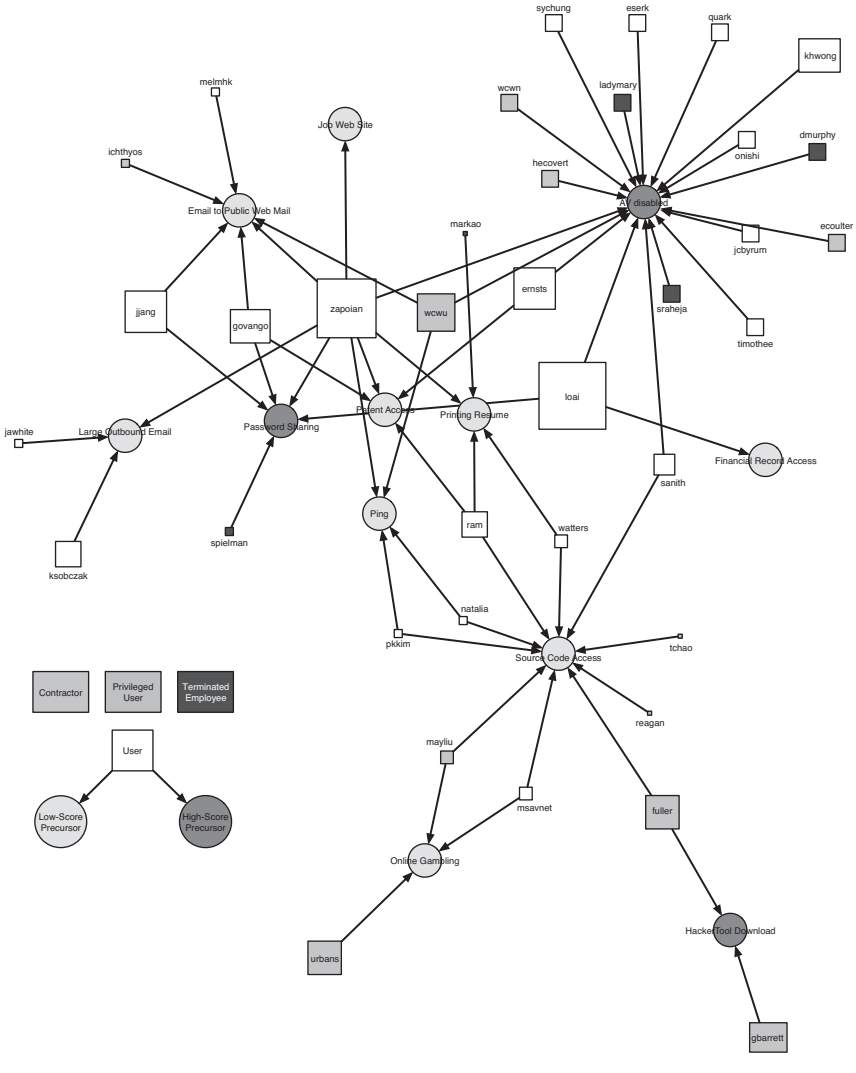

Figure 8-23 The insider candidate list shown as a link graph. The precursor nodes are colored based on how "bad" a precursor is deemed. The actor nodes are colored according to the watch list that the actor belongs to. (This figure appears in the full-color insert located in the middle of the book.)

Insider Candidate Treemap

After going through the link graph, we can move on to the treemap visualization. The input for the treemap differs slightly. We cannot use multiple files to generate the

treemap. Let's start out by defining a CSV file that contains all the necessary information and then convert it to a TM3 file that can be read by the treemap visualization tool. As input, we need a file that looks like this:

```
User,Precursor,Precursor_Score,Total_Score,Watchlist,Precursor_Bucket
jjang,Email to Public Web Mail,2,2,NONE,Minimal
jjang,Password Sharing,2,2,NONE,Minimal
khwong,AV disabled,4,4,NONE,Setup
ksobczak,Large Outbound Email,2,2,Contractor,Minimal
```

The calculation of the total score in this case is a bit trickier. Treemaps can display visual properties of leaf nodes only. What does that mean? In our example, we are using a hierarchy of Watch List > User > Bucket > Precursor. This means that the precursors are the leaf nodes, and therefore we can use properties of the precursors to be graphed as the box properties. However, we would like to make sure that the total size of the user boxes is set to a specific value, the total score a user accumulated. To do so, we have to go through some tricky calculations that adjust the total score in the input and assign it to the precursor's size. Here is the formula to do so for each precursor:

New precursor score = (Total user score / Sum of precursor scores for this user) * Precursor score

This will distribute all the precursor scores evenly, not paying attention to a maximum score per bucket. The sum of precursor scores for each user is now going to add up to the total score of the user. You can run the `capper.pl` script with the `–a` option, to do this calculation for you:

```
./capper.pl –a -u user_activity.csv -p precursor.csv –w watchlist.csv >
user_activity_new.csv
```

The following columns are needed in the preceding files:

- `user_activity.csv` `user, precursor`
- `precursor.csv` `precursor, bucket, precursor_score`
- `watchlist.csv` `user, watchlist`

Example output from the preceding command then looks like this:

```
ladymary,AV disabled,4,4,Setup,Terminated
spielman,Password Sharing,7,7,Malicious,Terminated
onishi,AV disabled,4,4,Setup,
```

When this all is done, we can convert the CSV file into a TM3 file (see the earlier discussion of converting a CSV file to TM3 in the section "Insider-Detection Process at Work") and then generate the treemap shown in Figure 8-24.

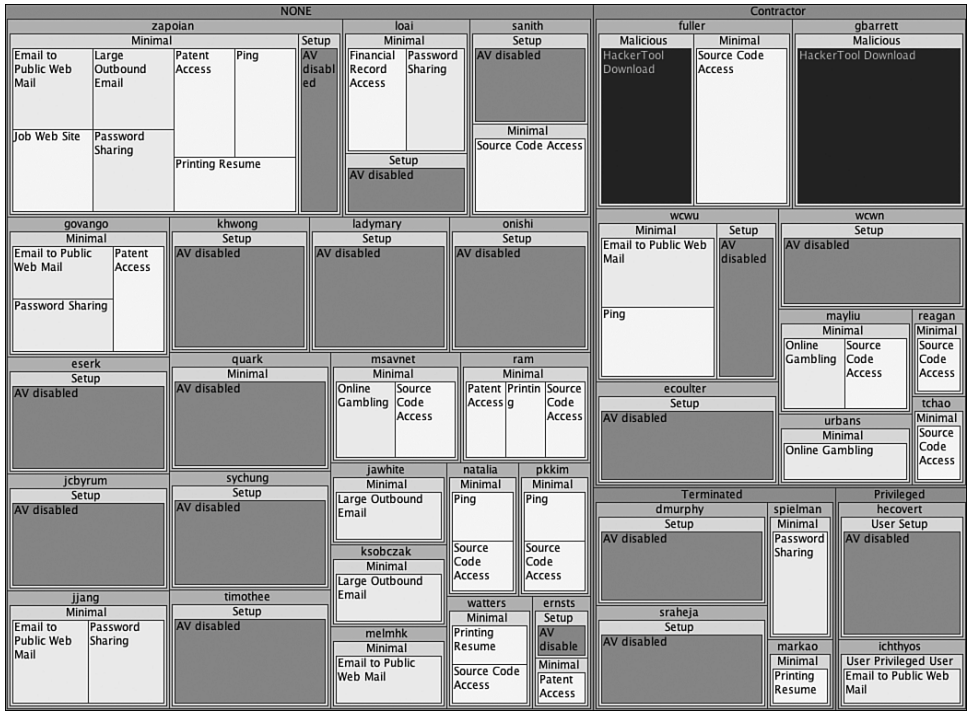

Figure 8-24 Treemap showing the accumulated score per user. The configuration hierarchy for the treemap is Watch List > User > Bucket > Precursor. The size of the nodes is based on the score a user accumulated. The color is based on the score of the individual precursor. The darker the boxes, the higher the score. (This figure appears in the full-color insert located in the middle of the book.)

We can see that Figure 8-24 now sizes the user boxes according to their total score. The precursor boxes are then proportionally distributed over that box. We can see some high-score precursors on the top right, indicated by the dark red boxes. We can also see that those precursors were triggered either alone or with one other precursor. Depending on the role of the users, this is not very significant. The fact that contractors triggered these precursors might be a bit alarming, unless these are security consultants. On the top left, we see that the user who accumulated the most points is zapoian. This user triggered quite a few precursors. However, all of them are in the low range of scores. It is probably still worth having a chat with this person.

The next step in the process is now to go ahead and tune the precursors. Based on the two graphs, we should go ahead and investigate more closely what all the activity signifies. Some of it will turn out to be absolutely benign, whereas other activities will probably be reason for concern.

CHALLENGES

One challenge remains with the insider-detection process. We worked with the assumption that all activity can be traced back to a user. However, a lot of precursors involve monitoring log files that do not contain information about the user executing the specific action. In the worst case, the log file does not contain any information about the subject. In some cases, there will be some sort of a subject represented in the log, but not the actual human responsible for the activity. A lot of log files will show an IP address, for example. There are three approaches to this problem:

1. All the activity can be traced back to machines (i.e., IP addresses), and this is all we care about. There is nothing else we need.
2. Each machine that generates traffic can be mapped one to one to a user, and we are in the possession of the mapping. We can use the mapping to track down the user responsible for the actions.
3. We use an additional source of information that links the activity from machines to users. Logs that contain this type of information are DHCP logs, authentication logs, identity and access management (IAM) logs, and so forth.

Another challenge in this context is an identity management problem. A user typically uses multiple user names across different systems. I use the user ram on my UNIX systems, but I am rmarty in the Windows domain of my employer. And even worse, sometimes I log in to systems as root rather than using `su` or `sudo` to switch from my user account to the root user.[9] Who is responsible for commands executed by the root user? This is definitely tricky and falls in the area of identify and access management (IAM). To get as close as possible to tracing back activities to users, collect as much information as you can and link things back to the real actor or user responsible for the activity. Also look at other data sources that might prove useful. For example, collect login information. If you know that a user logged in from a specific machine using the user identity ram and then the same machine opened a connection using another user rmarty, you

[9] Although I restrict this on most of my machines, sometimes I do log in as root on my desktop or laptop to do some maintenance work.

can be fairly certain that these user IDs belong to the same person. There are caveats, but this might be a viable source from which to start constructing user account maps.

This concludes our discussion of the insider-detection process. You should now have a starting point to address your information protection and insider abuse problems and challenges. Let's discuss some proactive approaches that may enable you to counter the insider threat problem.

PROACTIVE MITIGATION

The insider-detection process we developed throughout this chapter is a great step toward dealing with the malicious insider problem. However, unless we define exceptionally good precursors, the process is very much a reactive one. What we really need is a way to proactively mitigate the problem of insider crime. *Insider threats can be stopped, but it takes a complex set of policies, procedures, and technical means to do so.*

To prevent the problem of malicious insiders, we need to implement a comprehensive and effective security program. That is really what it comes down to. As you know, that is not an easy task, and there will always be gaps or areas that are not addressed by policies or technical safeguards. This means that all the work we did to identify users before they turn into malicious insiders is still relevant.

What this also means is that in addition to the insider-detection process we should try to mitigate some of the problems by implementing a "good" security program. A lot of processes that can help with mitigating the malicious insider problem do not directly relate to IT and computer systems but are human resource processes. Things such as awareness training or background checks are examples. The following list of IT security practices can be used to further minimize the window of opportunity for malicious insiders. All the following practices can be monitored via log files.

- Application of least privilege
- Access path monitoring
- Separation of duties
- Privileged access
- Rigorously following processes for configuration management
  - Deactivate access for users following termination
  - Strict password and account management policies and practices
  - Actively defend malicious code

- Use of layered defense against remote attacks
- Implementation of secure backup and recovery

One way to use this list is to define precursors and use those for detecting malicious behavior or preparation for insider crimes. Another, more effective way of using the list is to implement processes to enforce these properties. Separation of duties, for example, could be monitored via log files, but the disadvantage is that only cases of active violations can be detected. On the other hand, if you were to analyze the configuration of your systems to make sure that there are no conflicting roles associated with users, you could completely mitigate the problem. Take a look at Chapter 7, "Compliance," for a discussion about what such a verification could look like.

An overall solution should implement both prevention and detection measures. It is fairly likely that some cases will be missed by preventive measures. Perhaps those will be detected with the insider-detection process, however, and can be mitigated before any real harm is caused.

SAMPLE PRECURSORS

Precursors are an integral part of the insider-detection process. This last section of the chapter provides a collection of precursors that you can use to monitor actors in your own corporate environment. The precursors are organized in a set of tables. They list the precursors themselves, mention what electronic record or log file can be used to find the precursor, identify how the precursor is detected in the log, classify the precursor in one of the three categories of detecting the precursor (signature matching, policy monitoring, or anomaly detection), assign a score to it, and in some cases even show a sample log entry to illustrate what the precursor looks like.

Table 8-8 discusses precursors that apply to all three categories of insider crime: categories: sabotage, fraud, and information leaks. The following tables then discuss precursors specific to one of the insider crime categories.

Whereas some precursors apply to more than one of the insider crime categories, others are specific to a single category. Sabotage- and information-leak-specific precursors are listed in Table 8-9 and Table 8-10, respectively. And finally, Table 8-11 shows precursors for fraud.

Table 8-8 A Sample Set of Precursors That Apply to All Three Types of Insider Crime: Sabotage, Fraud, and Information Leaks

Precursor	Description	Data Source	Detection Method	Score	Detection Category
Use of organization's systems in violation of acceptable use policy	Activities such as playing games, using company equipment to work on personal projects, and so on.	Operating system	Log analysis.	1	Policy

Windows event log example:
`EventlogType=Security`
`DetectTime=2005-04-29 14:15:16`
`EventSource=Security`
`EventID=592`
`EventType=Audit_success`
`EventCategory=Detailed Tracking`
`User=COMPANY\yfan`
`ComputerName=NIGHTLESS`
`Description=A new process has been created`
`New Process ID=4856`
`Image File Name=\Program Files\Windows NT\Accessories\pinball.exe`
`Creator Process ID=3532`
`User Name=yfan`
`Domain=ARCSIGHT`
`Logon ID=(0x0,0x1BDF9)`					

Printing activity	Each of the following scenarios could indicate employees who plan to leave: • Printing resumé • Printing off-hours • Excessive printing	Printer logs	Log analysis.	1	Signature Anomaly
Windows printer log example (someone pasting information into a Notepad document results in the following log entry): `#Document 92, Untitled.txt - Notepad owned by ram was printed on HPIJ via port HPLaserJet4050Series.  Size in bytes: 12342; pages printed: 5`					

Precursor	Description	Data Source	Detection Method	Score	Detection Category
Job Web page access	Accessing job Web sites could be a sign that an employee is about to leave the company and is looking for new opportunities.	Proxy NetFlow Router Firewall	Log analysis with list of job websites.	1	Signature
OpenBSD pf firewall example: `Feb 18 13:39:27.667326 rule 71/0(match): pass in on xl0: 195.27.249.139.63280 > **66.218.84.150**.80: S 948010585:948010585(0) win 32768 <mss 1460,nop,wscale 0,nop,nop,timestamp 24077 0> (DF)`					
Access of Web sites prohibited by acceptable use policy	Accessing websites violating of the acceptable use policy could indicate employees who are not committed to their work anymore.	Web proxy	Log analysis.	2	Policy
Privoxy log example: `Apr 01 13:13:05 Privoxy(b55aaba0) Request: www.sex.ch/index.html`					
Account creation outside of normal business hours	Depending on the work habits of the system administrators, this could indicate a problem.	Operating systems Applications	Log analysis.	2	Signature
Download and use of password cracker	Regular users do not need these types of tools.	Web proxy Operating system (file change monitor)	Log analysis.	3	Signature
Privoxy log example: `Apr 04 19:45:29 Privoxy(b65ddba0) Request: www.google.com/search?q=**password+cracker**`					

continues

Table 8-8 A Sample Set of Precursors That Apply to All Three Types of Insider Crime: Sabotage, Fraud, and Information Leaks *(continued)*

Precursor	Description	Data Source	Detection Method	Score	Detection Category
Access of information outside of need to know	Users should not be accessing data unnecessary for their job.	Operating system Web server Applications	Log analysis: Define what is normal for each user or role.	3	Policy
Not complying with hardening or configuration guidelines	Not deploying patches or making necessary configuration changes could indicate deliberately left vulnerabilities that later could be exploited.	Configuration and Patch management Software distribution Vulnerability scanner	Log analysis: Look for patch or configuration management violations.	3	Policy
Use of anonymous proxy	Anonymizers can be used to conceal the destinations of connections and also encrypt their content. This helps evade detections mechanisms such as IDSs or DLPs.	Proxy NetFlow Router Firewall	Log analysis with list of public proxies.	3	Signature
Privoxy log example: `Apr 04 20:25:11 Privoxy(b7df9ba0) Request: www.anonymizer.com/cgi-bin/open.pl?url=http%3A%2F%2Fraffy.ch%2Fblog`					
Internal reconnaissance	Browsing applications and shares on servers could be a sign of someone doing reconnaissance. At a later stage, this knowledge could be used to execute an attack.	Applications Operating system NIDS	Log analysis.	3	Policy Anomaly

Precursor	Description	Data Source	Detection Method	Score	Detection Category
Deleting audit logs	Audit logs should be archived and only in rare cases should they be deleted. Deletion could be an attempt to hide traces.	Operating system (file change monitoring)	Log analysis.	4	Signature
Web traffic bypassing proxy	If a proxy setup is in place, all the Web traffic should be leaving via the proxy to guarantee accountability.	NetFlow Router Firewall	Log analysis: Look for direct Web connections.	4	Signature
Unauthorized use of coworker's machine left logged in	Utilizing someone else's equipment attributes activity to the wrong person and can be used to draw attention to an innocent third person.	Physical access control	Log analysis: User is either physically not in the building or in another building while his computer is being accessed.	5	Policy
Physical anomalies	Impossible concurrent physical access Logical access without physical access Access of off-limit facilities	Physical access control Operating system Applications	Log analysis: Correlate physical logs with logical access logs.	9	Policy
Creation of backdoor account	Unauthorized modems Creation of user accounts without a change request	Vulnerability scanner Operating system Applications	Log analysis.	9	Signature

Table 8-9 A Sample Set of Sabotage Precursors

Precursor	Description	Data Source	Detection Method	Score	Detection Category
Network probing	Regular users do not have any justification for scanning or probing machines on the internal network. This is reserved for security staff or auditors.	IDS NBAD	Checking for abnormally high traffic volumes or specific indicators of scans.	3	Policy Signature
Unauthorized encryption of information	If data is encrypted and the encryption key is lost, the data cannot be restored anymore. This could be an act of sabotage	NIDS	In some cases, a NIDS uses statistical analysis methods to detect encrypted traffic on the wire. It is another problem to then map the activity back to a policy and verify whether encryption was allowed.	3	Signature?
Changes to critical files	System configuration changes, or new cron jobs, both executed without an associated change ticket, are potential signs for mischief.	Operating system	Log analysis.	3	Signature
Disabling of anti-virus or other security software	Turning off security software will allow the user to execute commands that would otherwise be reported or blocked. This could be used to prepare for an attack.	Operating system Security tool log	Monitoring the shutdown of these tools or disabling of the service on the operating system level.	4	Signature

Precursor	Description	Data Source	Detection Method	Score	Detection Category
Failure to create backups as required	The next thing after omitted backups is usually a failure somewhere. This could indicate a setup for an act of sabotage.	Backup	Log analysis.	4	Signature
Download and installation of hacker tools or malicious code	Unless someone is a virus researcher, this is definitely not normal behavior. Rootkits and password sniffers are not part of the normal repository of applications for a user.	Proxy IDS	Log analysis Possibly monitoring sites known for virus code distribution.	7	Signature

Table 8-10 A Sample Set of Information-Leak Precursors

Precursor	Description	Data Source	Detection Method	Score	Detection Category
Anomalous email activity	Emailing documents to public Web mail accounts. Emails going to the competition.	Email Data-leak prevention Intrusion detection system	Log analysis with specific lists of recipients to watch. Analyzing cliques to find communication anomalies.	2	Policy
Unauthorized information transfer	Any piece of critical information in the wrong hands could lead to direct or indirect financial loss.	Proxy Application Email Instant messenger	Log analysis.	3	Policy
Checking someone else's email Microsoft Outlook example: `COMPANY/ram logged on as /o=COMPANY/ou=US/cn=Stefanie Boem Recipients/cn=ram on database "COMPANY\zurich(VMSG32)". For more information, click http://www.microsoft.com/contentredirect.asp`	Users should read their own email only.	Email	Log analysis.	4	Signature
Promiscuous mode interfaces	This is one example of security parameters to monitor. Promiscuous interfaces are a sign that someone is sniffing traffic. Except for networking or security people, this is not something an employee should be allowed to do.	Operating system	Log analysis.	4	Signature

Precursor	Description	Data Source	Detection Method	Score	Detection Category
Storage device attachment	The attachment of storage devices, such as CD-ROMs or USB storage, especially on server machines, could be an attempt to steal or remove data.	Operating system	Log analysis.	4	Signature
Traffic to suspicious countries	Some companies are more sensitive to where connections are going. If the company operates in Switzerland only, connections inside of Europe might be normal, but outside of that, they might raise suspicion.	Router Firewall NetFlow Proxy	Log analysis with either a white list or a black list to monitor the traffic endpoints.	4	Policy
Sharing passwords or user accounts	Each person should have a unique user account to guarantee auditability and accountability.	Operating system Application	Detecting access of one and the same account from two either physically separate places or different logical networks.	7	Policy
Laptop theft	No automated systems will detect a stolen laptop. However, the owner of the system should report it as soon as the theft is detected. That way, any future use can be flagged, and other necessary precautions can be initiated.	Human	Every company should have a theft-reporting process. This is a manual process.	8	N/A

Table 8-11 A Sample Set of Fraud Precursors

Precursor	Description	Data Source	Detection Method	Score	Detection Category
Changes to critical files	Changes to critical files can be used to mask behavior, change intrinsic properties of systems, and serve as a pathway into fraudulent activity.	Operating system Applications	Log analysis.	3	Signature
Direct DB access	Generally, databases are accessed via applications, and data is not directly changed in the database itself. Direct changes in the database can be used to circumvent application layer security controls.	Database Application	Monitor specific tables to ensure access via the dedicated application only.	6	Policy
Role-based access monitoring	Each user role has specific activities they execute. Monitor for violations. For instance, database administrators accessing data tables in the database might be a bad sign.	Database Application Operating system	Monitor logs to ensure user actions stay in their roles.	6	Policy

The precursors listed here are only examples. Many more precursors can be used, especially if you define precursors significant for your own environment. Use these precursors as a starting point and let them inspire you to extend the list.

Figure 8-25 summarizes the lists of precursors from this section in a graph. The graph will help you get a feel for the types of precursors the tables introduced and in what areas we might have to spend some more time to define new ones. You can see four groups of data points. The cluster on the left identifies all the generic precursors, the ones that apply to all three categories of insider crime. The next cluster represents the fraud precursors, followed by information-leak precursors, and then on the right side you can find the precursors that flag saboteurs. The graph shows that most of the precursors have a score associated with them that is in the range of about 3 or 4. This indicates that it is not trivial to find precursors with a high score. Furthermore, there are hardly any precursors colored in black, which are ones that use anomaly detection to find precursor activity. It seems that generic precursors, ones that apply to all three insider threat categories, are the easiest ones to find.

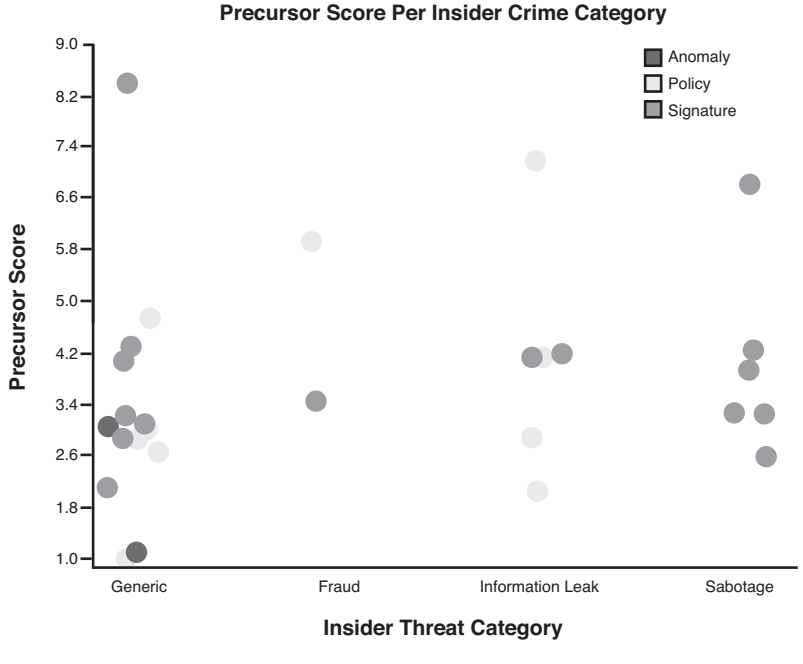

Figure 8-25 Summary of all precursors introduced in this section and their associated scores.

SUMMARY

Implementing a security program is not finished when the perimeter is secured. Although that is an important and necessary task to prevent external attackers from abusing our computing resources and stealing information, we also need to protect our systems and information from malicious insiders. The problem with detecting malicious insiders is that no security devices or programs can generate an alert when malicious insiders become active. It is more complicated than that. A lot of data has to be analyzed and monitored to detect malicious insider activities.

In this chapter, we explored an insider-detection process that can be implemented to find instances of insider crimes and people setting the stage for such activities. We have used visualization to analyze the vast amount of data and narrow it down to pertinent information. We have discussed precursors that can be used to trigger when malicious activity is detected. Using a scoring schema, we then rated the activities to focus on users who drew a lot of attention. Using visualization, we analyzed all the precursor activity. This insider candidate graph helped determine whether the activity was commonly observed for groups of users, which in turn helped tune the precursors by letting us define exceptions for specific user groups.

In a second round, we improved the detection process by grouping the precursors into buckets and only allowing a maximum score per bucket. This addressed one of the initial shortcomings of the detection process. Another improvement was the addition of watch lists to monitor specific groups of users more closely or score them differently.

Unfortunately, with today's technologies, it is not always possible to easily implement the insider-detection process. There is an information problem. Most important, few log files identify the human responsible for the activity. It is not always possible to do so for an application, but we need systems that can tie all the information together to trace activities back to humans. There is not much use in determining that 192.168.23.1 was the offender. It is much more effective if we can "nail" Joe for committing a crime. A discussion of this identity problem concluded this chapter.