A New Approach to Software Security

Computer Measurement Laboratory, Inc.
info@cmlab.biz

All modern software design methodologies have their origin in the very primitive standalone computer environments. These computer systems were not hooked up to the Internet nor even simple local networks. Typically one job ran on the computer at a time and jobs were batched and run sequentially. In this environment, the only threat to a program's integrity while it was executing was provided by the ineptitude of the computer operator. Life was simple enough that the execution of the program could be monitored as it ran through light displays on the operator's console. Control of the program was provided by the computer operator. Over time, control of program execution gradually shifted from the operator to the operating system. A human being simply could not respond in a timely fashion to the demands of a program running at electronic speeds.

From a computer security standpoint, the computer hardware, the operating system, user applications and the computer operator were all well contained within the confines of a room whose perimeter could be secured with existing security technology. In this limited environment, there was little need to consider invasive forces from outside. That only came when the complete secure environment was compromised by attaching it to a completely unregulated and potentially hostile information ether.

As an aside, it is interesting to note that evolution of the original computing environment was almost exactly duplicated by the evolution of the personal computer. Originally these boxes were self-contained computers. Folks used them for entertainment and, to a limited extent, for business. Then everything changed (except for the underlying security premise). People attached modems to their PC's and connected to the internet. The software that they were running was designed to run in a contained and completely encapsulated environment.

The Internet was not a major factor in the design of early Windows O/S's. It was certainly not even considered when the architectural framework for Unix was laid down. It still isn't a design consideration in the evolution of legacy code for managing a personnel system. We have a security problem today because the same naive premise about the operating environment of most our software is still in effect. That is, the software will operate in a closely confined hardware facility and if it is to be exposed to a more hostile world, a big hairy brother security system will protect it.

Security Cannot be Added On

As each new homeowner in the Latin countries to our south moves into his/her new house, the first attention given to the house is always in the area of security. An ironmonger is called and iron bars are installed on all of the street accessible windows. This defensive measure obliges any would-be burglar to carry a hacksaw to ply his trade. In response to this threat, the new homeowner will then construct a large wall around the perimeter of the property. This countermeasure now mandates that the potential burglar carry a ladder in addition to the hacksaw. In response, the homeowner will contract with a mason to install a row of broken glass bottles on top of the fence to deter the would-be burglar. The burglar must now add to his working stores, a burlap sack to cover the broken bottles in addition to the ladder and the hacksaw.   In response, the homeowner will then buy a trained attack dog to patrol the yard for burglars.   Now the burglar must increase his toolset with a pound or so of hamburger laced with strychnine (to kill the dog) in addition to the burlap sack, the ladder, and the hacksaw.

This cycle continues for many iterations until, the owner is impoverished, moves into a castle with a moat around it, leaves town or learns to live with the fact that burglary is a fact of life. It is a continuing cycle with no prospect of success. This is very much the saga of today's computer security field. The end game of the introduction of a new security technology is that a better breed of hacker is created. The only apparent way out of this game is to build a castle with a moat around it and only lower the drawbridge for known friends (and hope that they have left their Trojan horses at home). In the information security arena, the analog of the castle complex is the trusted O/S. Castles are big, hard to maintain, and damned inconvenient for the folks who have to live in one. So are trusted O/S's.

The alternative to massive security battlements is relatively simple. The security infrastructure may be incorporated in the design of the device to be protected.

South Florida is a region of great prosperity. This is due in part to the trade in recreational pharmaceuticals. Many dealers in recreational pharmaceuticals operate out of there.   Their approach to home security is very different from the South Americans'. Soon after acquiring an existing house, a drug dealer will have the house gutted and the grounds ripped up. As the house is being refurbished, security sensors that would stretch the budget of the CIA are embedded in every nook and cranny of the house. Similarly, as the place is being landscaped, sensors are built into the outside decor as well.

The long and short of the remodeling effort is that these new rehabilitated structures cannot be approached or compromised in any way. This is a good thing in that most dealers could not avail themselves of the usual criminal investigative infrastructure should their houses be compromised by a burglar.   Though there are no accurate statistics on the rate of successful burglaries against a house so protected, it is safe to assume that this rate is astonishingly low as measured against the typical Latin house.

The security bottom line is simple. Security cannot be added on to any system, hardware or software. It must be built in. The security controls and sensors must be an integral part of the software system they are to protect. If it is an operating system that must be secured, then the controls must be integrated into the operating system. If it is a payroll system that must be protected, then the controls must be embedded into the payroll code.   That is where the attack takes place. That is the best place to recognize the attack.

Forensics

Much of the computer security industry today seems to be modeled on the current start of the art in forensic work in police technology.   In this scenario there are many players. The first of these players is the criminal. This person conceives of an unhappy event, the crime, and then sets about work to execute the criminal plan. The second player in the game is the victim. This is the person how suffers the attentions of the criminal. After the crime there enters a psychologist who provides victim counseling. On the scene shortly after the crime are the police detectives who analyze the scene of the crime for clues and evidence that will ultimately lead back to the criminal. The evidence collected at the crime scene is collected and taking back to the police laboratory for extensive forensic analysis by a specially track staff.

Once the characteristics of the criminal have been established, the police are then alerted to go find a subset of criminals whose characteristics match those of the evidence collected at the scene of the crime. When the criminal(s) is (are) apprehended they are taken to jail where they are monitored by a team of professionals specifically trained to prevent the criminal returning to society. At this point another team takes over. There are lawyers for the defense and lawyers for the prosecution. There is a judge who oversees the proceedings also overseen by twelve hapless citizens drawn at random from their daily activities to participate in this drama.

At the end of this long play there is the final resting place of the criminal. This is a very expensive motel run by the government and presided over by a team of trained specialists whose sole role is to provide continuous supervision for the criminal.

This is an astonishing array of people. It began very simply with one criminal and one victim. Had the victim taken the appropriate defensive measures to thwart the crime, none of the other actors in this drama would have been necessary.

The current technology in the computer security industry today parallels the drama outlined above. The primary emphasis is on forensics. This emphasis leaves the victim well exposed. There must be a crime for the rest of the system to kick in. Our approach is entirely focused on the activity of the criminal ***before*** the crime has taken place. We have discovered that criminal activity can be identified at a very early stage. Potential intruders identify themselves and their intentions long before their criminal objectives are attained. Our methodology identifies the early onset of the criminal activity and ***rejects*** the intrusion.

We do not believe in information crime. We do not believe that there must be victims of this type of crime. If there is no crime, there will be no need for the forensic analysis of intrusion events. A very small investment in preventive measures will eliminate a vast market for post-intrusive forensic analysis. We don't want to know who committed the crime. We don't want the crime to occur in the first place.

Signature Based Systems

There are two major problems with the current state of the art in intrusion detection. First, response latencies are far too great. This inhibits the ability of the system to ameliorate the effect of the intrusion. Second, the overwhelming majority of the current IDS systems are signature based. They cannot recognize new assaults for which no signature data are available.

In order that we might understand the real problem with current IDS technology, a simple analogy is in order. We will imagine that there is a company, the Alpha Company, that provides security services to banks to help the banks eliminate the problem of bank robberies. As part Alpha's services, they will provide banks with a basic files of pictures of known bank robbers. To use this service, the customer banks of Alpha are instructed to post a security guard by the bank exit to monitor the people leaving the bank. As each person walks by the guard, his/her face will be compared to the file of bank robber pictures supplied by Alpha.

The problem with this system, so far, is two fold. First, the guard is making the comparison with the picture file as the customer (or robber) is leaving the bank. Thus, the guard will be able to tell us that our bank has just been robbed, which, of course, we already know. The damage has been done. The information provided by the Alpha system is certainly not timely. We would have liked to have prevented the robbery and not just identified the robber. Second, the majority of banks are held up by folks that have never robbed a bank before. Thus, we are unable to recognize the face (signature) from the picture file provided by Alpha because it is not there. Alpha's security system is not timely because of the response latency and it is not cogent because of the dependency on incomplete information.

As part of Alpha's security system they will provide their customer banks with updates to the picture files from time to time. The picture files will grow fairly rapidly over time as Alpha gets more historical data (pictures) of bank robbers. Ultimately the flow of legitimate bank customers past the guard will exceed the guard's ability to compare their faces with the picture file. To solve this problem, the bank will have to hire another security guard to help the first guard. They will split the file in two and each will examine his/her own file against the departing customers. As time progresses, the number of pictures in the two files will be too large for the two guards and a third security guard will have to be recruited. As time progresses, and the picture files grow without bounds, a significant amount of bank personnel will be devoted strictly to the security problem. The security overhead will have grown burdensome.

Measurement and Control Approach

The obvious goal is to create a security system that enables a security system or system administrator to control completely the behavior of a system, without having to personally micro-manage it. Controlling any system involves three steps: measurement, decision, and action. System administrators have the tools at their disposal for taking the actions necessary to manage their machines. They also have the experience and the skills to make decisions. The problem is a lack of meaningful measurement data on which to base decisions.

We have worked extensively in the instrumentation of the Linux kernel. We have developed software that automatically embeds sensors into software to gather behavioral data as a system executes. These sensors also tag the behavioral data with the cause of the behavior, be it an IP address or a process ID. With the sensors providing data, measurement of the activity of the Linux kernel becomes possible.

We have developed a real time monitoring system for the Linux kernel, and other software, that captures the sensor data in real time. This monitoring system provides for the creation of a model that represents stored behavioral data and then performs measurements against the model as new behavioral data is received. These measurements determine the relationship between the new data, and the stored model of behavior. The new data can then be compared with the known behavior of the system. This, in turn, permits policy decisions can be made and action taken to suppress unwanted or anomalous system activity.

The Bottom Line

Security problems are the result of a lack of control. Computer systems that are completely controlled by security or system administrators do not have security problems. Unfortunately, modern operating systems do not provide the necessary sensor data or measurement technology to make accurate control decisions. We have developed the sensor and measurement technology necessary to enable real time control over software systems. When the activity of software systems is monitored in real time, it is simply not possible to compromise the software.

Finally, security should not be an afterthought. The framework for the computer security system should be with the systems from its inception. This involves an integrated software/hardware approach to securing the software run time environment. Real time monitoring of this environment is the only realistic solution to the security problem.