Personal Firewalls and Intrusion Detection Systems

Joan Dyer, Ronald Perez, Reiner Sailer, Leendert van Doorn
{joandy, ronpz, sailer, leendert}@watson.ibm.com

IBM T. J. Watson Research Center

ABSTRACT

In this paper we explore how secure coprocessors can be used to secure client devices, especially mobile clients such as notebook computers. The goal is to protect data on the mobile client in case of theft, and to adapt the client's protection to the working environment --such as the Intranet or Internet-- without relying completely on the integrity of the client.

We show how physically secure coprocessors can introduce a security lifecycle into commercial off the shelf (COTS) clients by booting the client into a secure starting state, supervising the client's configuration and operation, and inspecting any network traffic that is sent or received by the client.

Keywords: Security, Secure Coprocessor, Applications.

INTRODUCTION

Physically secure coprocessors, such as IBM's 4758 (IBM 1997), nCipher's *nShield* (nCipher), and Baltimore's *SureWare Keyper* (Baltimore), have almost exclusively been used in server configurations, where the secure coprocessor primarily provides cryptographic acceleration and secure key storage. The use of secure coprocessors in client configurations has hardly been explored. This is surprising since physical protection offers many additional benefits for mobile clients. The need has been underscored by several recent high profile incidents in which notebooks containing highly sensitive information disappeared. A physically secure coprocessor would have offered much better protection in these cases.

Ideally we would like to secure mobile clients in the same way we do secure coprocessors, using a tamper responsive enclosure and a controlled execution environment, but the cost of such a system would be prohibitively high. In particular, we consider a new secure coprocessor implementation that we call Mycroft. It builds upon the features the IBM 4758 (IBM 1997), borrowing others from related projects (Neal *et al.* 2000), and adds a network Ethernet interface. This Mycroft device, in PCCARD/CARDBUS format, acts as an independent party within the mobile client.

In the following sections we discuss a number of applications of physically secure coprocessors and show how the security of the client is improved. In the ‘Applications’ section we describe those applications that, running on such a device, enhance the client's security. These include: secure bootstrap, host

monitoring, intrusion detection systems (IDS), firewalls, and controlled VPN. Conclusions and future work are discussed in the last section.

APPLICATIONS

The order in which we present the applications is motivated as follows: First what security mechanisms must be implemented securely? Second, which security mechanisms, as parts of distributed applications, profit from secure environments? Third, which applications can utilize secure environments to run in potentially hostile environments?

Figure 1 depicts the inter-relationships of the applications that we present in this section.

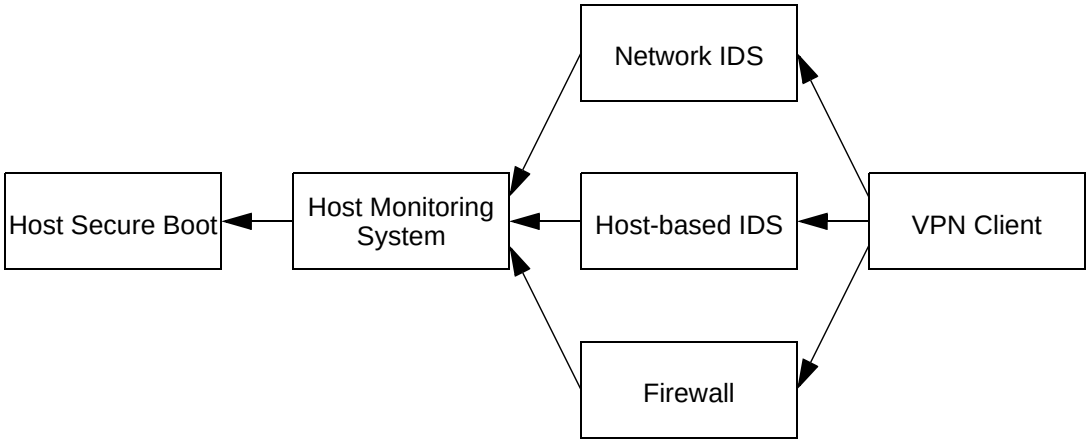

Figure 1: Application Inter-relationships

The host secure boot application ensures that the host is booted into a secure state. The host monitoring system supervises the operation of the host and depends on the initial secure state of the host to determine the kernel data structures and configuration files that must be evaluated. The IDS and the firewall depend on the host information provided by the host monitoring system, e.g., to examine host configuration files for those services that are passed through the firewall. Finally, the VPN client depends upon all the other components to ensure that the security status of the host conforms to the security policy of the VPN server throughout the connection phase.

Host Secure Boot

Host *secure boot* (Arbaugh *et al.* 1997) is a collective term for a combination of mechanisms that provide integrity guarantees during system startup. These provide initial system guarantees such that, if an operating system kernel is started, the specific kernel and machine configuration are known. The latter includes hardware configuration (disks, Ethernet boards, amount of memory, *etc*.) and software configuration (firmware versions, bootstrap loaders, *etc*.). The initial system guarantees prevent an attacker from modifying the bootstrap loader or firmware, or adding additional equipment without approval.

The host secure boot scheme consists of two distinct stages. The first stage is the configuration stage and the second is the verification stage. During the configuration stage we checkpoint the state of the system, that is: enumerate all the devices within the system, compute secure hash values over all the host firmware and its controllers, and compute secure hash values over the bootstrap loaders, kernel image, and other crucial files necessary for bootstrapping the system. This configuration is then signed by a security officer and stored in the secure coprocessor. Only configurations with a valid signature are accepted.

Rather than assuming the host is secure, we have the secure coprocessor verify that the host is in a secure configuration. We accomplish this by giving the insecure host a large number of puzzles to solve that involve some computation. These puzzles are different on every reboot and every puzzle has a chance associated with it that the host can guess the right answer even though it did not perform the associated computation. Since all puzzles are independent, these puzzles can be used to achieve a certain confidence level, say 99.99%, the host is executing with a valid configuration.

While the exact nature of the puzzle generation process is beyond the scope of this paper, the puzzles themselves need to be generated within a trusted environment that is different from the host. The puzzle answers should be verified within that same trusted environment. Obviously, the ideal place to generate the puzzles and verify them is within a secure coprocessor.

The secure boot process outlined above requires a safe place to store the secure state configuration for the host, and it requires the host to execute verification code. To trust the answers from the host we either ensure that a potential adversary cannot modify the verification code and its answers, or use the secure processor and software puzzles to achieve high confidence that the host is in the secure state configuration. We are currently exploring both approaches, but the former has preference because of its simplicity.

Intrusion Detection Systems

Computer systems commonly feature the ability to process programs of different users at the same time. The systems keep track of the users' identity and access rights to shield different users, their programs and data, from each other.

In practice, this does not work. Networked computer systems, offering services to remote users, are always as vulnerable as the authentication method that is used to determine the identity and access rights of remote and local users. Today, this is mainly user ID and password based authentication.

Additionally, computer systems have become more and more complex, and we do not know any general-purpose computer system that claims to run certified and formally verified (not to speak of verifiably correct) programs.

Exploiting weak authentication schemes or computer programs, users can gain administrator privileges on almost any commercial off the shelf computer system. Having such privileges, the system can be compromised. Information and programs of any other user can be modified and disclosed.

Hence, tools have been developed to detect known attacks that might compromise the integrity of a computer system. Combining the IDS with notification services, alarms are sent to local or remote users indicating the type of potential ongoing attack and proposals for reaction, as illustrated in figure 2.

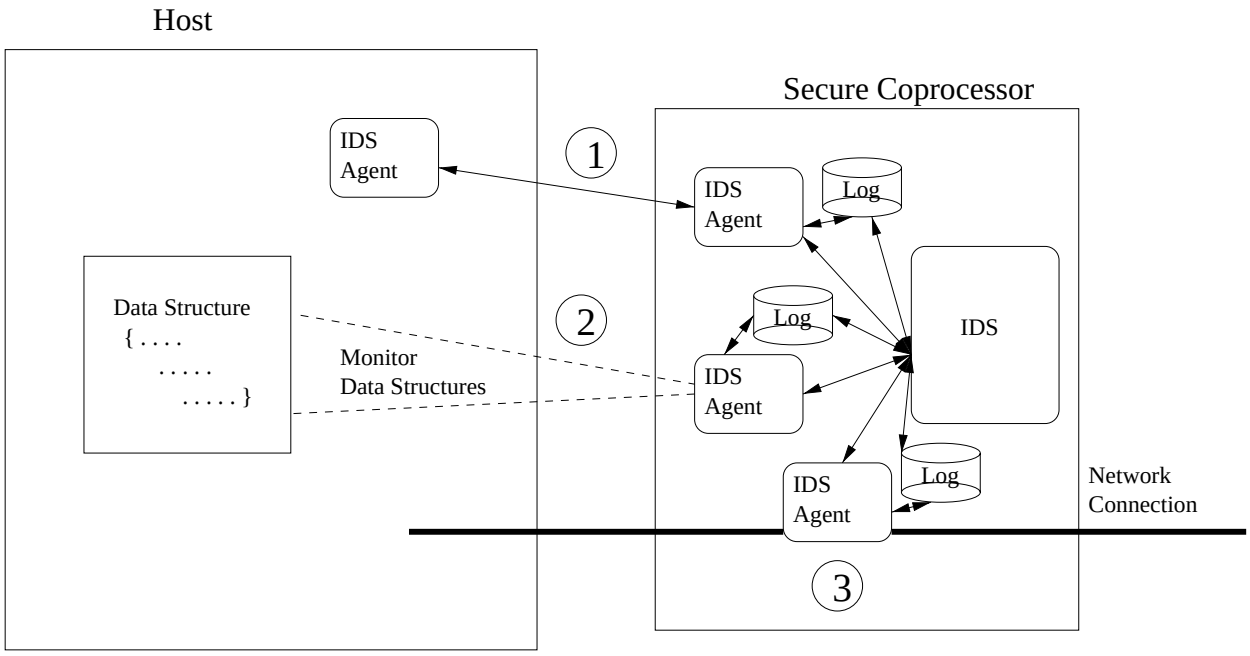

Figure 2: IDS with secure coprocessors

In this figure we illustrate (1) a host-based IDS with agents residing in the host, in the secure coprocessor, or both, (2) a simplified host monitoring system, and (3) a host-based network IDS, all with audit logs residing in the secure coprocessor (although these logs may be signed by the secure coprocessor and sent to the central control systems as well). The agents themselves could consist of pattern matching, anomaly detectors, and profile engines, or some of the analysis activity could be performed by the main IDS engine, which itself probably communicates with the central control systems.

There are different traditional approaches to implementing Intrusion Detection Systems. We will first discuss mechanisms that are based on host events and then mechanisms focusing on network events.

Host Monitoring System

Another application of a secure coprocessor is that of a host monitor, whereby an application on the secure coprocessor acts as an independent party that keeps a tap on programs executing on the host.

The monitoring application is closely associated with the host secure boot mechanism. Secure boot provides a strong integrity guarantee; it ensures that only an approved operating system is started on an approved hardware and firmware configuration. The ability to provide these strong guarantees vanishes as soon as the host processes external input, because the external input potentially allows code-injection attacks (for example) that inject foreign code into an initially secure configuration. While we are no longer able to provide strong integrity guarantees, we can take advantage of the secure coprocessor to ensure weaker integrity guarantees.

Weak integrity guarantees have the property that an adversary on the host may circumvent them. However, the level of sophistication required by the adversary depends on the level of difficulty of the monitoring program. Examples of monitoring programs include, a Tripwire (Kim & Spafford 1994) like process that, independent of the host, accesses the disk, computes checksums over the files stored on it, and then compares the result against a database of know values. Unlike the Tripwire product, where the

checksum database and verification process reside on the host and are therefore vulnerable to attacks, our database and verification process reside within the secure coprocessor, which is inaccessible to attackers.

Another function of a monitoring device is data structure invariant checking. Here the secure processor is given invariants of the data structures in host memory (for example kernel data structures that describe user access permissions). At regular intervals the monitoring program checks whether the invariants still hold. If an invariant fails it can either log this event or disable the functions of the secure coprocessor.

The host monitoring capabilities still require further investigation. The advantage is that they can provide additional but weaker integrity guarantees when the strong guarantees no longer hold. A serious problem with host monitoring applications, however, is that they have the potential to invade the privacy of the user.

Host-based Intrusion Detection Systems

Host-based intrusion detection systems use information originating on the host to detect misuse. Different information sources can be used, such as system and application event logs, account usage statistics, critical data access and modification, *etc.*

Host-based IDS are usually agent-based, consisting of executables that run on the host and communicate with some central control system. Raw data is collected and sent to the central control system for analysis. If the analysis indicates known or potential attacks, the central control system is the originator of the prescribed response or remedy. Raw data can also be analyzed locally, whereby the agents send status and alerts to the central control system as appropriate, thus reducing the load on the central control system and potentially affording nearer to real-time or in-time detection and response. Often, a combination of both approaches is taken.

The advantage of applying a secure coprocessor to host-based IDS is similar to that of service processors in many servers and mission critical systems today. That is, the ability to function independent of the host's system and application software and, to some extent, independent of various hardware components and subsystems of the host system. However, service processors are themselves not immune to physical or logical compromise. In most cases service processors are not designed with security in mind, and therefore cannot be trusted to the extent that secure coprocessors can (provided they have undergone various development process review and inspections, and more importantly, some sort of independent security evaluation). Most service processors also do not offer a general purpose computing environment, secure storage and management of data, or the ability to perform [hardware assisted] cryptographic operations in a timely fashion.

IDS agents, and the various policies they enforce, can be located and executed internal to the secure coprocessor where proper execution can be ensured. These agents would have access to data residing on the host --on storage devices (in file systems), or main memory-- for event log analysis, pattern/signature matching, and the like. In some instances, raw data such as the actual system and application event logs, behavior and usage statistics, *etc.*, may reside in the secure coprocessor as well, ensuring the integrity and confidentiality of such data (integrity checksums/hashes maintained in the secure coprocessor for data that is resident on the host may be used for this purpose as well), and also controlling access to this data. Because of it's ability to ensure data integrity, housing raw data on the secure coprocessor with its trusted time stamping abilities, even temporarily, would also facilitate further [off-line] analysis, as well as data forensics and support for any potential litigation --e.g., prosecution support or liability defense.

The secure coprocessor can also be used for surveillance of targeted data, programs, system resources, or accounts when misuse is already suspected. Surveillance is similar to the general data gathering and analysis described previously. However, surveillance is more finely tuned for specific attack or misuse scenarios. With the use of secure coprocessors, the host is oblivious to the fact that surveillance is being conducted, or that any IDS functionality is taking place beyond the bounds of what is normally performed (e.g. a sudden or unexpected change in an IDS audit or detection policy), avoiding the possibility of alerting potential perpetrators.

The secure coprocessor may also be used in conjunction with the host's system software, via redirection or collaboration, to assert access rights and privileges, or to monitor specific system call and system resource usage. The secure coprocessor is also an ideal platform for certain automated responses, such as responding to an anticipated attack (where a set of activities matches the signature of known preliminary attack pattern), thus stopping the attack or misuse before it begins (Proctor 2001).

Scalability over time (‘slow attacks’) and space (large numbers of hosts), where the central control systems cannot adequately address the amount of data needing to be analyzed in a timely fashion, is a problem for many current IDS solutions (Ptacek & Newsham 1998). In an environment where secure coprocessors are employed almost ubiquitously for IDS [and perhaps other] purposes, the collective IDS, or at least the analysis functionality, can be implemented in a highly distributed fashion in order to take advantage of the processing resources of the collective system of secure coprocessors, each with the ability to communicate securely and independently with each other.

Reliability of information sources (e.g., sensors and agents), as well as reliability of the analysis engines and response mechanisms, are also problems for many of today's IDS solutions (Ptacek & Newsham 1998, Bace 2000). With their ability to securely execute specified programs and ensure the integrity of collected data, secure coprocessors offer unique advantages to address many of these reliability problems.

An IDS (its agents) must execute in a secure environment. Otherwise, if the host is successfully compromised before or without the IDS noticing, the IDS is itself subject to compromise, rendering it useless. Given the role of host-based IDS, it will likely be [one of] the first targets of an attack.

The patterns/signatures and rules/policies that define what the host-based IDS looks for, reports, and responds to, should be confidential. Otherwise, attackers will likely be able to avoid specific patterns, time their attacks in order to avoid event timing windows, or react by launching subsequent or related attacks before the IDS can counter the original attack. As IDS and pattern detection are likely to never be comprehensive, shielding information from attackers is important.

Having physical control and protection of the IDS is also necessary to protect the IDS from misconfiguration by the host's administrator / operator. This is a critical issue as IDS functionality becomes more ubiquitous and the number of devices benefiting from IDS increases dramatically.

Using physical security provided by the secure coprocessor and IPSec or SSL to establish secure connections from the corporate security center to the IDS (e.g., in a telecommuter's laptop), enables secure remote management, update, and monitoring. Using certified secure coprocessors enables users to establish trust in the state of the secure coprocessor.

Similar to many of the applications envisioned here, host-based IDS implemented with secure coprocessors need sufficient general purpose and cryptographic processing power to accomplish the necessary tasks in a timely fashion. Since IDS are data intensive, sufficient system and local bus bandwidth, as well as network bandwidth and the capacity of onboard secure storage, will also be important factors in seamlessly implementing IDS with secure coprocessors.

As the numbers of computing systems increases, the average costs decrease (especially for mobile and pervasive devices where disconnected IDS abilities will be necessary), secure coprocessor footprint and the need for cost-effect but secure platforms to aid in IDS functionality will also be necessary.

Host-based Network Intrusion Detection System

The more traditional network-based, or promiscuous-mode, IDS consist of several to many sensors deployed throughout the network monitoring all/most network traffic to and from many systems on a segment. Host-based network IDS, in contrast, are usually more widely distributed among systems on the network, but need only monitor and analyze network traffic to and from the target host. Host-based network IDS also tend to communicate and collaborate with each other more so than the traditional network-based IDS. Also, with the narrower scope of a host-based network IDS (not having to monitor all traffic on a given network segment), the number of packets that can be inspected, patterns checked, and the analysis performed, can be finer-grained.

The host-based network IDS periodically checks the ``open'' services of the host to keep its view consistent with the host. The IDS can even shut down services on the host if they contradict the security policy or it can be combined with a firewall that forwards traffic to a local dummy proxy (firewall/IDS integration). Therefore, many attacks based on confusing the network IDS or installing new services won't work on such guarded systems.

As the host's interface to the network or particular network segment, the secure coprocessor-based network IDS has access to all network traffic originating from or destined for the host, even before the host has a chance to see the packet in the latter case.

The secure coprocessor, with its hardware accelerated cryptographic abilities, can also address one of the biggest problems for a host-based network IDS: encryption. Encrypted networks and secure tunnels into and out of the host (VPNs) encrypt the payloads of packets, making it impossible for the IDS to analyze traffic data for known patterns, viruses, and the like.

Operating in promiscuous mode, the secure coprocessor-based IDS can even function as a traditional but secure network-based IDS sensor, with the necessary computing power to perform much of the necessary analysis -even more so should such devices be widely deployed across the network.

Having the IDS directly at the host affords a better means to counter insertion/ deletion attacks (more effective, as host behavior can be predicted precisely). The secure coprocessor-based IDS could also handle packet de-fragmentation internally, countering overlap-attacks. And serving just one host, the IDS reacting to a detected attack can immediately isolate the host, shutting down questionable connections, and finally re-integrating the host into network.

Employing secure coprocessors for host-based network IDS offers many of the same advantages that using secure coprocessors in a purely host-based IDS solution does, including:

- Trusted and reliable execution of IDS agents, and secure storage of critical data --e.g., event log files, policy parameters, resource usage and pattern statistics, *etc.*
- The secure coprocessor can be independently updated and configured/managed from the host in a secure fashion.

- A highly distributed network IDS can take advantage of numbers and computing power of a system of secure coprocessor-based agents, communicating independently and securely with each other.

Secure run-time environments promote secure remote management and update of IDS software, patterns, *etc.* The secure coprocessor is a portable device that accompanies the user and can support multiple security policies, switching between a ``telecommuter user'' mode and an ``company user'' mode. Neither the organization nor the private user can install programs that are not certified by a common trusted party that --from a security perspective-- owns the secure coprocessor.

Thus, we tackle a problem regarding privacy requirements for remote/mobile users: the potential supervision of remote users by organizational security officers. This problem is partly solved by enabling a user to determine the policy of the active IDS. By establishing a trusted third party, users can make sure that the corporate IDS software, being signed by the third party, only retrieves and stores the information agreed upon between the organization and its member, the user. Disabling the IDS, even temporarily, makes no sense from a security perspective, as successful attacks launched when the IDS is inactive remain undetected even when the IDS is later re-activated.

Host-based network IDS have challenges similar to host-based IDS implemented with secure coprocessors, perhaps with more emphasis on network bandwidth and sufficient onboard resource to perform packet inspection and re-assembly as necessary.

Firewalls

Firewalls protect sub-networks or host by countering attacks that originate from the network. They intercept any traffic from and to a protected host, here the client. This traffic is examined and restricted to specific ports and protocols as stated in the security policy; in doing so it restricts client applications that are remotely accessible (Cheswick & Bellovin 1994, Zwicky *et al.* 2001). Insecure services (such as NETBIOS on Windows systems or Sendmail on Unix systems) must be blocked when connecting directly to the Internet and enabled when connected to the Intranet where corporate firewalls offer protection against attacks from the Internet. As most applications listen on well-known ports using well-known protocols, a firewall can protect clients independent of the client's operating system and environment by filtering incoming packets based on port number and protocol.

To protect client machines inside and outside the Intranet, we need to enable firewalls at the client. Pushing firewalls from the sub-network boundary to the client machine has been discussed earlier by Steve Bellovin in his much-cited paper about distributed firewalls (Bellovin 1999). Recently, he also admits that there might not be correct client systems in the near future (Bellovin 2001). Thus, besides trying to identify attacks exploiting respective vulnerabilities as described in 2.2, remote access in unprotected environments (Internet) to vulnerable applications should be disabled or minimized by firewalls. For instance, there is no point in keeping Windows NETBIOS services open when the user is at home and connected to the Internet; although they might be needed open when the user is inside the corporate Intranet to access shared resources. The same holds for remote management services.

Users running vulnerable applications are prone to become the victim of remote attackers. Once attackers successfully exploit such vulnerabilities, they can easily manipulate the user's run-time environment and gain long-term control over the user's computer - without the user being aware of this.

In effect, the remote user's client becomes the weak point in the company security architecture and defeats the security policy that is established and enforced by corporate firewalls -- a huge waste of money!

However, implementing firewalls on the client itself makes them as vulnerable to manipulation as any other host software. They are of no practical worth once an attacker gained access to the client (e.g. by executing a single --downloaded-- manipulated program on the client).

Thus, we propose to enhance mobile and home computers by a security device running a firewall that conforms to the corporate security policy. Incorporating firewalls into the secure coprocessor as illustrated above, these firewalls can be remotely managed and updated by the corporate security officer and are protected against user's misbehavior and against (potentially manipulated) applications running on the client. In order to optimize flexibility and security, we envision two generic operation modes for personal firewalls:

- The company mode firewall protects clients inside the company Intranet. It keeps more services open, as the client is protected against attackers by the corporate firewalls of the Intranet. Client services, such as NETBIOS on Windows OS and SNMP agents, remain accessible to share drives, access directory services, or remotely manage the client configuration.
- The telecommuter mode firewall protects clients that are connected through an ISP to the Internet. This mode is much more restrictive, as the clients are unprotected against attacks from the Internet. Any NETBIOS, mail, ftp, or telnet services of the client are most likely to be blocked.

To be effective, a company mode firewall must ensure that no other network interfaces are active through which attackers could bypass the firewall and build up a connection from the Internet into the corporate Intranet. Besides this, the environment must support remote configuration by the company security officer, remote management and regular inspection by a trusted party, protect company secrets such as management keys and passwords, filter rules, filter settings, security logging data, and should be certified to ensure the company that it works properly and the users that their privacy is not compromised by the device.

On our secure coprocessor we are going to extend the Linux filtering and firewall functions (Linux) to include checks of host integrity (interfacing the IDS) and checks of the configuration of the service being addressed by the inspected packet (interfacing the host monitoring system). The packet is then forwarded to proxies within the secure coprocessor, passed through to the client, or discarded depending on the policy settings.

As both user and corporation must trust the firewall, it must resist attacks in hostile environments (user, company, or environment, lost devices *etc.*). Secure coprocessors offer an ideal place to implement such personal firewalls independent of the client's operating system and management.

Although personal firewalls introduce a new granularity of protection, they are only part of the solution. They need to be combined both with IDS and with the host monitoring system to ensure effective and secure operation:

- The IDS ensures that those services that are open are not manipulated or used for break-ins via known vulnerabilities.
- The host monitoring system must ensure that those services that are open are configured appropriately (e.g., no read/write anonymous ftp directories, no overlapping ftp directories, ftpd run with changeroot and minimal privileges).

To conclude, personal firewalls protect mobile computers in a flexible way according to the situational need. Running these firewalls on multi-party trusted environments enables them to implement multi-lateral policies, namely the corporate security policy and the user's privacy policy. A general-purpose secure run-time environment also opens the possibility of running several proxy services on the secure coprocessor that implement finer-grained control of a single service (e.g., FTP, SMTP, HTTP application level command filters).

There remain some challenges. First, it is not easy to find the right policy, which is neither too restrictive nor opens vulnerabilities. Additionally, we need to discover and (to a certain degree) control any other potential network interfaces --e.g., internal modems, PCMCIA, parallel port, serial port-- that are controlled by the client and could be used to bypass the firewall.

Virtual Private Networks

As working from home becomes more and more convenient, such user-managed systems, directly exposed to the Internet, become part of the corporate Intranet by using VPN clients on the remote systems and connecting to VPN servers inside the VPN. The VPN connections are encrypted and are usually passed through the Intranet firewall to the VPN server without further inspection. The VPN server authenticates the user and implements the IPSec tunnel endpoint.

Thus, insecure and manipulated clients pose a substantial security risk to any organization that allows remote workers to access the company VPN or to process sensitive data. Figure 3 shows the basic scenario and the way attackers can circumvent the corporate firewall via a vulnerable client outside the Intranet.

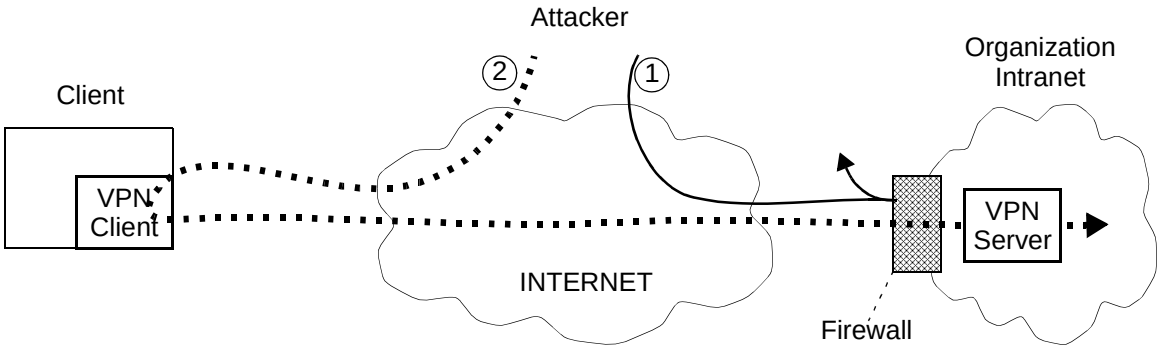

Figure 3: Circumventing the corporate firewall

The figure shows that instead of defeating the well configured and professionally managed Intranet firewall of the corporation (1), attackers gain access to ill-managed remote clients attached directly to the Internet (2). Via this machine's VPN client, the attacker can go straight into the corporate VPN, whose traffic is usually allowed to pass the Intranet firewalls without further inspection. The attacker will always be able to pass into the corporate VPN server along with the regular user because the user opens the authentication door for any application on the client and for any Intranet service offered to remote VPN users.

Whether or not the attacker can walk in and out of the corporate VPN via manipulated clients depends mainly on the security of the authentication used:

- Systems using user password authentication are completely insecure once the client is hacked. An attacker controlling the client can easily eavesdrop the VPN userid and password. Having userid and user password, attackers gaining access to the VPN client code can effectively access the corporate VPN from wherever they like.
- One-time passwords --e.g., produced by SecurID cards (RSA)-- disclosed to the client, e.g., entered into a pop-up window, offer more security, if the validity period is short enough and the timeout of VPN connections is reasonably short. The attacker must operate within the validity period of the one-time password after the user signs in and discloses a valid one-time password to the client machine; the security in this case is mainly defined by the expiration of a VPN connection, which determines the time window for attacks using a disclosed one-time password. The attacker cannot operate without the user logging into the VPN server.
- Disclosing one-time passwords or user passwords to the secure coprocessor only (via small keyboard or smart card), restricts attackers significantly. In this case, attackers cannot authenticate against the VPN server even if they possess the VPN client code (at least not by merely controlling the client). The attacker can operate only as long as the VPN client remains connected to the VPN server and the VPN client cannot be manipulated to keep open VPN connections hanging despite users closing them. Additionally, IDS, firewall and host monitoring system ensure that the client is very well supervised and protected against compromise right from the beginning.

The first two scenarios (user password and one-time password authentication) were less critical when clients were connected through the phone system to a single Internet Service Provider; in this case the client was either connected to the VPN or to the Internet. This posed security problems as manipulated clients could retrieve valuable information when connected to the VPN and later send the information to the attacker when connected to the Internet. However it was far more difficult to achieve:

- Clients had IP addresses assigned by the ISP and these addresses were dynamic. Thus attacking a dedicated client was difficult without real-time Internet Service Provider information -- although not impossible.
- Clients were not online all the time; thus networking attacks had to be synchronized with the client's connectivity.
- The gathered information had to be retrieved somehow by the attacker. As the IP address of the client changed, the client usually sent the information to a specific address, newsgroup or any other place; this addressing information is found on the client in case the manipulation is detected and raises the risk of disclosing the identity of an attacker.

Attacking clients that are connected to the Internet 24 hours and 7 days a week is far easier. First, IP addresses are static or do not change often, i.e., once a victim's machine has been successfully attacked, the attacker can address the machine remotely. Attacks can be scheduled any time the client is unobserved -- although, this could be countered by running IDS on the clients. Third, gathered information can be retrieved remotely without the need to disclose any remote address to the manipulated client (other than in forgotten log files). In a last step, attackers gain real-time connectivity to the company Intranet via the client and can hack other machines inside the Intranet to elevate their privileges.

Therefore, an effective security policy grants remote clients access to the corporate Intranet via VPN tunnels, if and only if the security status of the remote system is trustworthy to the company.

But how can we remotely determine the status and trustworthiness of a remote client?

At this point, we leverage our Mycroft device and its regularly updated evaluation of the client's security state. The remote access security policy could look as follows. A VPN client is connected to the corporate VPN server, if and only if the following conditions hold:

- The VPN client is running inside the secure coprocessor.
- The secure coprocessor is not tampered.
- The secure coprocessor is in telecommuter mode.
- The firewall and intrusion detection systems are activated (and then in telecommuter mode) and the logs suggest that the client is not compromised.
- The host monitoring system reports the client status as secure taking into account the history of the client since the last secure boot.
- The VPN tunnel setup is successful (IKE/IPSec in VPN client and server cooperate).
- The client authentication via the secure coprocessor is successful.

Applying our Mycroft device to protect a VPN client is shown in figure 4.

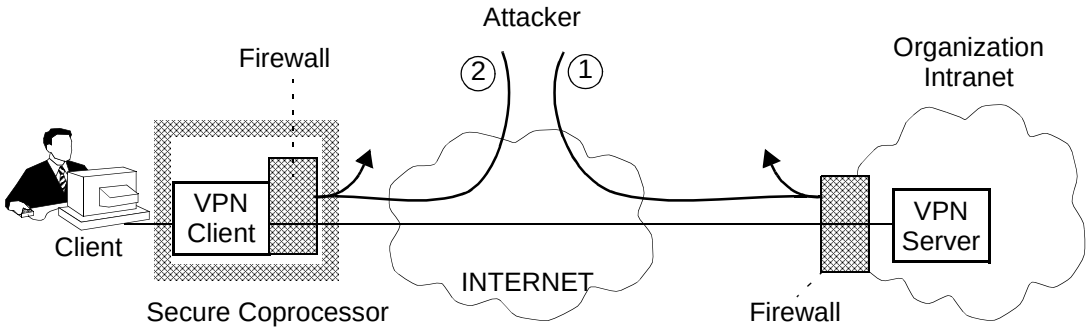

Figure 4: Firewall and VPN in client-side secure coprocessor

Implementing IKE and IPSec inside a secure device, any long-term secrets (such as keys for IKE authentication) are protected against the user and other potential attackers. Combining firewalls, intrusion detection systems, VPN client, and IKE/IPSec functionality inside the secure device enables secure remote management of the device.

This, in turn, gives organizations a valuable control over the remote end of their VPN tunnels. Decisions on whether or not to allow remote users access to the corporate VPN can be made based on the client's security status.

Additionally, the VPN client and VPN server can be enhanced to offer different security levels, in which different sets of Intranet services are available to remote users. The VPN client would tell the VPN server which security level it wants to connect and the VPN server would retrieve respective policy. Then, the VPN server would request the VPN client, who in turn requests the secure coprocessor, to establish this security level in the client (e.g. by rebooting it or changing configurations). In this way, the same client machine can participate in different security levels. To raise a security level, it will usually be necessary to reboot the client to re-establish an initial security level. Some security levels might not be achievable by clients managed by normal users.

We won't be able to completely prevent attackers using compromised clients to walk into the Intranet, but we can make it seriously hard for them and we can restrict access to services (e.g., differentiating access levels as WWW services, Mail, or Developer access).

CONCLUSION AND OUTLOOK

Secure, general-purpose, self-defending devices of various strengths provide the necessary platforms for ensuring trust and security. The examples elaborated above in the Applications section show how such a programmable device can be used in a variety of ways, controlled either by the user or remotely by an organization such as a service provider or an employer, for mutual benefit.

The embedded platform prototype of the Mycroft device is up and running. We chose to adapt Linux as the operating system on a PowerPC embedded platform, and to deploy FreeSWAN IPSec, IPTables, and open source IDS systems to implement the security services described above. We are going to integrate those applications, and develop useful configurations and remote policy management on the embedded system. At the same time, we will develop the host secure boot and the host monitoring application from scratch.

There are other applications which rely upon protecting sensitive data, or which are sensitive *per se*. Such applications may need to function in environments in which the host has weak security, or is controlled by parties with opposing interests. This area is eminently well suited to secure, general-purpose, coprocessors. Sensitive data (such as keys) used for authentication or signature services can be kept within the secure boundary. Sensitive programs can be stored and run within the secure coprocessor.

REFERENCES

Arbaugh, W.A., Farber, D.J., and Smith, J.M (1997). *A Secure and Reliable Bootstrap Architecture*.

Bace, R.G. (2000). *Intrustion Detection*, MacMillan Technical Publishing.

Baltimore SureWare Keyper. http://baltimore.com/products/sureware/keyper.html

Bellovin, S.M. (1999). *Distributed Firewalls*, login.

Bellovin, S.M. (2001). Computer Security – An End State? *Communications of the ACM*, 44, pp. 131-132.

Cheswick, W.R., and Bellovin, S.M. (1994). *Firewalls and Internet Security*, Addison-Wesley.

IBM 4758 PCI Cryptographic Coprocessor (1997). IBM Product Brochure G325-1118.

Kim, G., and Spafford, E. (1994). Experience with Tripwire: Using Integrity Checkers for Intrusion Detection. *System Administration, Networking, and Security Conference III.* USENIX.

Linux IPChains and Firewall HOWTOs. http://linuxdoc.org

nCipher nShield products, http://www.ncipher.com

Neal, S., Hartmen, M., Morgan, S., and Laue, M (2000). *Sherlock: Commercial High Assurance Network Computing*. 3rd International Workshop on Information Security – ISW2000, Australia.

Proctor, P.E. (2001). *The Practical Intrusion Detection Handbook*, Prentice Hall.

Ptacek, T.H., and Newsham, T.N. (1998). *Insertion, Evasion, and Denial of Service: Eluding Network Intrustion Detection*. http://secinf.net/info/ids/idspaper/idspaper.html

RSA SecurID. http://rsasecurity.com/products/securid

Zwicky, E.D., Cooper, S., and Chapman, D.B (2001). *Building Internet Firewalls*, O'Reilly & Associates.