ENFORCEMENT OF SECURITY POLICY COMPLIANCE IN VIRTUAL PRIVATE NETWORKS

Haidong Xia, Jayashree Kanchana and José Carlos Brustoloni
Department of Computer Science
University of Pittsburgh
210 S. Bouquet St. #6135
Pittsburgh, PA 15260 – USA
{hdxia,kanchana,jcb}@cs.pitt.edu

ABSTRACT

Virtual Private Networks (VPNs) enable an organization's members to telecommute from home or while traveling. Although members may use computers that are shared, borrowed, or rented from others to connect to a VPN, VPN protocols, such as IPsec, typically do not authenticate the configuration of users' computers. If a computer used for VPN access is compromised, an attacker can exploit it to gain unauthorized access. We propose the use of attestations *to overcome this vulnerability. An attestation is a disclosure of a computer's configuration, signed by a secure coprocessor. We contribute protocol enhancements that enable attestation to be combined with IPsec, such that only an organization's members that use uncompromised computers can gain and maintain access to the organization's VPN. Experiments demonstrate the efficacy and efficiency of our solution.*

1 INTRODUCTION

Many organizations maintain Virtual Private Networks (VPNs) that enable its members to connect via the Internet to the respective organization's intranet (Yuan and Strayer, 2001). Members may use VPNs, e.g., when they are at home or traveling. VPNs often use IPsec to secure such communication. IPsec can mutually authenticate connection endpoints and guarantee packet integrity and confidentiality (Kent and Atkinson, 1998) .

The computers used by members are often weak links in VPN security. VPN protocols, such as IPsec, typically do not authenticate the configuration of users' computers. However, while away from an organization, members often use computers that are shared with other household members, borrowed from a person they're visiting, or rented from a cybercafé. If an attacker has compromised such a computer, the attacker may be able to gain unauthorized access to an organization's intranet via a member's VPN connection.

We propose the use of secure coprocessors to overcome this vulnerability. The Trusted Computing Group (TCG) (Trusted Computing Group, 2005a) has standardized secure coprocessors (called TPMs – Trusted Platform Modules) (Trusted Computing Platform Alliance, 2002) that have low cost (about $4) and are embedded in an increasing number of computers from IBM, HP, Dell, and other manufacturers.

Secure coprocessors enable *attestation*, i.e., unforgeable disclosure of a computer's configuration to a remote party (Pearson, 2003). This paper contributes protocol enhancements that enable attestation to be combined with IPsec, such that only authenticated users with uncompromised computers can connect to an organization's VPN. Man-in-the-middle (MITM) attacks are possible if attestation and IPsec endpoints are not the same. We propose the use of *bound keyed attestation* (BKA) to thwart such attacks.

Modifications are necessary also in the operating system for properly supporting secure coprocessors. First, the integrity of the system's trusted computing base (TCB) needs to be measured and stored in the secure coprocessor. A system's TCB includes not only the operating system's kernel, but also loadable kernel modules and privileged user-level servers, scripts, and configuration files. We propose *TCB prelogging* to guarantee that measurements of all these components are properly stored in the secure coprocessor. Second, the operating system must prevent undetected modifications of the system's TCB. Administrative users typically can modify a system's TCB after the system boots. We propose *security association root tripping* for automatically closing attestation-based IPsec connections when an administrative user logs in, and ensuring that subsequent attestations reveal that the system may have been modified. Third, the operating system needs to prevent applications from using secure coprocessors in potentially harmful ways. If an operating system gives applications raw access to secure coprocessor functions, an application could bind to itself documents or other files created by the user. The user is then *locked in*, because he cannot access the information using competing or future applications (Felten, 2003). We propose

sealing-free attestation confinement to enable secure access to VPNs without software lock-in.

The rest of this paper is organized as follows. Section 2 reviews the secure coprocessor features that we use for protecting VPNs. Section 3 describes our operating system modifications for supporting secure coprocessors. Section 4 explains how we integrated attestation with IPsec. Section 5 presents our experimental results. Section 6 discusses related work, and Section 7 concludes.

2 AUTHENTICATED BOOT AND ATTESTATION

This section reviews the two secure coprocessor features, *authenticated boot* and *attestation*, that we use for authenticating the configuration of users' computers in VPNs.

Authenticated boot presupposes that when a host is reset, control of the CPU is transferred to a small, trusted, immutable software component. In a personal computer, this component is the BIOS boot block. The operating system's boot sequence is modified such that, before each software component *A* passes control to another software component *B* that has not yet been measured, *A* measures *B*'s digest, appends the digest to a *measurement log*, and compresses the digest into the TPM. *A* obtains *B*'s digest using SHA-1 (Secure Hash Algorithm) (NIST, 1995). SHA-1 digests are 20 bytes long, regardless of data length. This algorithm has properties such that it is infeasible to modify *B* without also modifying its digest, or to find a *B* whose digest is an arbitrary value. The TPM compresses a digest into one of its registers by concatenating the register's value and the digest, computing SHA-1 on this concatenation, and storing the resulting value back into the register. The TPM's registers are initialized to zero on host reset. They can be read, but cannot be otherwise modified. TPM register values can be used to authenticate the measurement log, whose plaintext is stored in main memory.

Attestation is a protocol that enables a remote party *R* to obtain and authenticate a host *P*'s measurement log. *R* sends to *P* a nonce, i.e., a cryptographically random number that is never reused. *P* asks its TPM to sign a so-called *quote*, containing the nonce and current values of the TPM's registers. Presence of the nonce in the quote guarantees that a quote cannot be later replayed. The signature uses an attestation identity key (AIK). *P* sends the corresponding AIK certificate to *R* together with the quote and measurement log. TPMs generate private and public AIK pairs internally. TPMs use private AIKs only to sign quotes (as defined above), and never reveal such keys externally. The host owner obtains the AIK certificate from a so-called *privacy certifying authority*, which verifies that the host contains a properly attached TCG-compliant TPM. *R* authenticates the AIK certificate using the certifying authority's public key, which *R* is assumed to know out-of-band. *R* then uses the nonce and public AIK to authenticate the quote and uses the quote to authenticate the measurement log. The authenticated log reveals securely to *R* what software booted in *P*.

3 OPERATING SYSTEM MODIFICATIONS

This section describes our operating system modifications for safely supporting secure coprocessors.

3.1 TCB prelogging

A computer's TCB is the part of the computer's configuration that a VPN needs to authenticate. A system's TCB can be defined as the set of components whose malfunction (e.g., due to a bug or attack) would allow violation of the system's (or the VPN's) security policies. The TCB includes not only a system's BIOS, master boot record, boot loader, operating system kernel, and loadable kernel modules, but also their configuration files and any privileged applications that could modify them, such as scripts or daemons and setuid applications owned by administrative users. Many TCB components may not be directly relevant to VPN security policies, but VPNs need to authenticate them because attackers could use them to compromise more relevant components. On the other hand, unprivileged applications and other files that cannot change a system's security policies are not part of the system's TCB.

The operating system needs to ensure that the respective computer's measurement log contains the digests of all components in the system's TCB. This can be difficult for two reasons. First, information about what files are or not part of the TCB is typically quite diffuse in existing systems. Second, certain TCB components may be loaded and started only when needed. Therefore, it does not suffice to measure TCB components that start at boot time.

We propose *TCB prelogging*, whereby the operating system contains a configuration file that lists the system's TCB components and respective digests. This TCB list is itself part of the system's TCB, since, e.g., omission of a TCB component in

the list could cause that component not to be authenticated. By convention, the digest of the TCB list file is computed considering its value in the file to be null, and then stored in the file.

The operating system uses the TCB list for *prelogging*. The kernel appends the entire TCB list to the system's measurement log and compresses the list's digests into the TPM at boot time, including components not started at that time.

Thereafter, whenever a file that is a TCB component or a script or daemon or setuid application owned by an administrative user is opened or executed, the kernel measures the file's digest. If the digest is different from that last logged for the file (if any), the kernel closes any attestation-based security associations that may exist, appends the new measurement to the log, and compresses the new measurement into the TPM.

Therefore, an authenticated measurement log will reveal all TCB components that are configured to exist in the system (according to the TCB list), whether active or not, as well as any unconfigured or modified TCB components that have started since the system booted. Moreover, a computer's operating system will automatically disconnect the computer from a VPN if the operating system loads a TCB component that was not disclosed in the log that the VPN used to authenticate the computer's configuration. The user may then attempt to reconnect the computer to the VPN, but a new attestation will reveal the previously undisclosed TCB component. Thus, only computers whose configuration a VPN accepts can connect and remain connected to that VPN.

3.2 Security association root tripping

Users with a regular/limited account cannot cause an operating system's kernel, loadable kernel modules, or daemons to compromise the system's policy enforcement. (If that is not true, the system has a bug that needs to be fixed.) However, users with an administrative account (e.g., root) *can* easily violate the system's policy enforcement, e.g. by using commands such as sysctl or ifconfig or by using a debugger to attach and modify privileged processes, after boot time. It can be difficult or impossible to guarantee that all such configuration modifications are captured in the system's measurement log.

We propose to modify the operating system such that it detects and takes appropriate action when an a user attempts to gain interactive access to an administrative account in the system (e.g. by logging in or using the su command). If there are any attestation-based security associations (e.g. to a VPN), the system warns the user that, if the user wants to continue, the system will immediately drop those security associations and destroy the respective keys used for packet-level cryptography. Furthermore, if the user wants to continue, the system appends this event to the measurement log with a well-known digest and compresses the latter into the TPM.

Thereafter, attestations will reveal that an administrative user has logged in interactively since the system booted. VPN administrators can configure VPN gateways to deny access to such VPN clients, who will need to reboot before again gaining access to the network (reboot erases the measurement log, as well as any non-persistent configuration changes; persistent changes are captured by the new measurement log after reboot).

Note that in contemporary operating systems, such as Windows XP and Linux, a same user typically may have multiple accounts, each with a different username and possibly a different privilege level. It is usually considered good security practice (principle of least privilege) for a user to perform as many tasks as possible using a regular/limited account, instead of an administrative one. Because errors or attacks on the latter can much more severely compromise a system, many enterprises give administrative accounts only to specially trained system administrators. Security association root tripping does not prevent users who have administrative accounts from connecting to a VPN, but while connected, such users must use regular/limited accounts only. Security association root tripping also does not preclude remote system administration or help through the VPN, as long as these are performed using in client computers daemons that VPN gateways are configured to trust.

3.3 Sealing-free attestation confinement

In addition to attestation, TPMs provide a feature, *data sealing*, that enables storing a file encrypted with a key that a TPM reveals only if the computer's configuration is the same as at the time the file was first stored.

An application can use data sealing to lock in a user's files. The application stores the latter encrypted with keys that the operating system makes available only to that application, and that the TPM makes available to the operating system only if the computer's TCB is the same as at the time of storage. The operating system denies the

files' keys to other applications. If the user tries to reconfigure or replace the operating system to circumvent this protection, the TPM's data sealing makes the keys unavailable.

Alternatively, an application can use attestation for the same purpose. The application stores file keys in a remote server. The server reveals such keys to a computer's operating system only if attestation shows that the computer's TCB is one that the server trusts, and the computer's operating system vouches to provide the keys only to the original application.

Software lock-in can benefit well-established software publishers. However, it can also harm users because it can block competition and make users unable to access their own data in other computers or using other applications (Felten, 2003). In the case of archival data, e.g., such access can be necessary when the computer and application originally used become obsolete.

To prevent software lock-in, we propose that the operating system not export the TPM's raw functions to applications. In particular, we propose *sealing-free attestation confinement*, whereby the operating system does not export data sealing to applications and supports attestation only in conjunction with network access control protocols, such as IPsec's IKE (Harkins and Carrel, 1998). Most organizations configure their security perimeter such that these protocols cannot go through firewalls. Consequently, attestation is unavailable for lock-in applications and is confined within each organization.

4 INTEGRATING ATTESTATION WITH IPSEC

This section describes our protocol enhancements for integrating attestation with IPsec's Internet Key Exchange (IKE) (Maughan et al., 1998; Harkins and Carrel, 1998).

IKE has two phases. In phase 1, the communicating parties authenticate each other and establish an IKE security association (SA). The IKE SA is bidirectional and determines what cryptographic algorithms and keys the parties will use to communicate during phase 2. In phase 2, parties use the IKE SA to negotiate one or more AH (Authentication Header) or ESP (Encapsulating Security Payload) security associations between them. AH and ESP SAs are unidirectional. AH provides packet authentication, while ESP can provide packet authentication and/or encryption. Phase 2 negotiations determine the cryptographic algorithms and keys used in each AH and ESP SA.

We propose performing attestation using the IKE SA between IKE's phases 1 and 2. Attestation is computationally intensive and reveals details about a computer's configuration that the computer's owner may wish to disclose only to selected parties. Because phase 1 authenticates both parties, it enables this desirable policy-based control. Phase 1 also establishes the IKE SA, which protects attestation from eavesdropping. On the other hand, because attestation happens before phase 2, it does not need to be repeated for each AH and ESP SA negotiation.

We define a new IKE exchange type for attestation. This *attestation exchange* comprises the *attestation payloads* that parties send to each other for unidirectional or mutual attestation. We also define attestation policy configuration options according to which each party initiates and/or accepts an attestation exchange with the other party at the end of IKE's phase 1.

Although we protect the attestation exchange with the IKE SA, a man-in-the-middle attack could be possible. If IKE SA and attestation endpoints do not coincide, an attacker may be able to gain VPN access using a compromised computer, despite the use of attestation. Consider an attacker that wishes to gain access to a VPN using a compromised computer C_1. If the attacker has another computer C_2 that conforms to the VPN's security policies, the attacker can have C_1 forward to C_2 attestation requests received from the VPN and, conversely, forward to the VPN attestation replies received from C_2. To the VPN, it will appear that C_1 conforms to the VPN's security policies, even though it does not.

To block this vulnerability, we propose the use of *bound keyed attestation* (BKA). BKA differs from ordinary attestation (Section 2) in that (1) BKA includes a Diffie-Hellman key exchange (Diffie and Hellman, 1976), whereby attestation endpoints derive a shared secret, and (2) attestation endpoints prove to each other that they know both BKA's and IKE SA's shared secrets. If attackers cannot obtain BKA's and IKE SA's shared secrets (e.g., because VPNs do not accept configurations where that would be possible), then man-in-the-middle attacks are not possible.

BKA uses two publicly known numbers for the Diffie-Hellman key exchange: a prime number q and an integer α that is a primitive root of q. These numbers do not need to change. The *attestation initiator* A (e.g., the VPN gateway) picks two

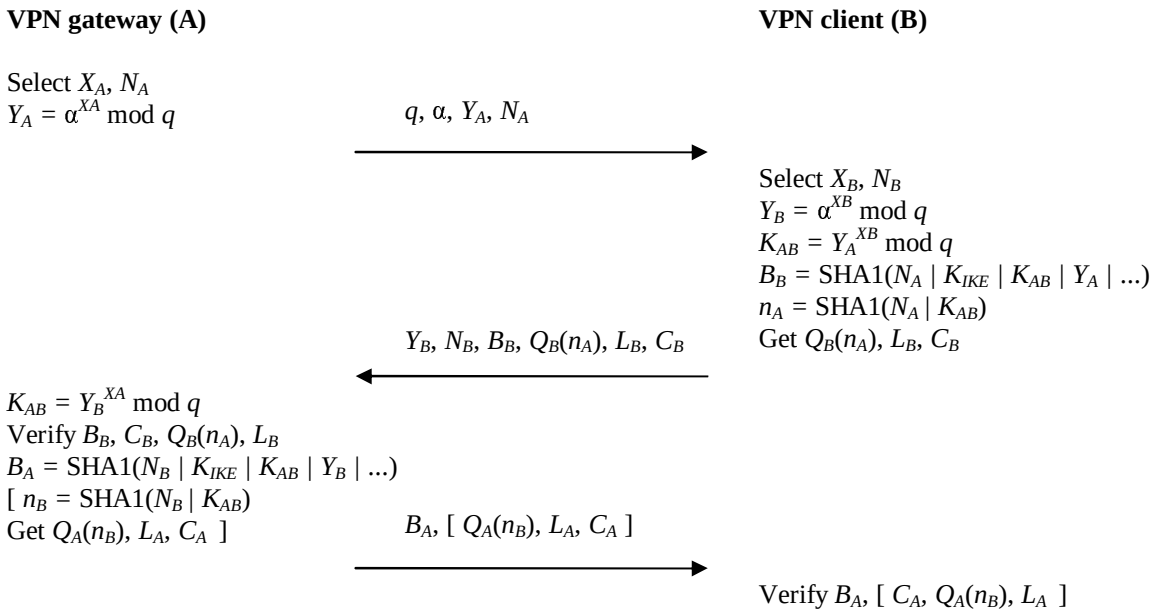

Figure 1: Bound Keyed Attestation (BKA). Items between brackets are used only in case of mutual attestation.

random numbers: a nonce N_A, which is never reused, and another integer $0 \leq X_A < q$, which may be reused. The initiator computes its public key according to:

$$Y_A = \alpha^{XA} \bmod q. \quad (1)$$

The initiator then sends to the *attestation responder* B (e.g., the VPN client) a *BKA request* message M_i containing q, α, Y_A, and N_A, as illustrated in Fig. 1.

The responder then picks a random integer $0 \leq X_B < q$, which may be reused, and computes its public key according to:

$$Y_B = \alpha^{XB} \bmod q. \quad (2)$$

The responder computes the *attestation shared secret* as:

$$K_{AB} = Y_A^{XB} \bmod q, \quad (3)$$

and its *attestation binding:*

$$B_B = \text{SHA1}(N_A \mid K_{IKE} \mid K_{AB} \mid Y_A \mid \text{“BKA response”}), \quad (4)$$

where '|' denotes concatenation and K_{IKE} is keying material derived from the IKE SA's shared secret. The responder also computes the *initiator's attestation nonce:*

$$n_A = \text{SHA1}(N_A \mid K_{AB}). \quad (5)$$

The responder then gets from its TPM its quote Q_B containing n_A. Finally, the responder sends to the initiator a *BKA response* message M_r containing Y_B, a fresh nonce N_B, B_B, Q_B, and B's measurement log L_B and AIK certificate C_B. Alternatively, if the responder also wishes to obtain the initiator's attestation, the responder sends to the initiator a *BKA response-request* message M_{rr} containing the same fields as M_r.

The initiator processes M_r as follows. First, the initiator computes the attestation shared secret as:

$$K_{AB} = Y_B^{XA} \bmod q. \quad (6)$$

The initiator then computes the expected value of B_B using K_{AB}. If the received B_B does not match this expected value, the initiator sends a *BKA error* message to the responder (possibly the IKE SA and BKA endpoints are not the same, and a MITM attack is happening). Otherwise, the initiator authenticates C_B using the respective certifying authority's public key (known securely out-of-band), authenticates the responder's quote Q_B using n_A and C_B, and authenticates the responder's measurement log using Q_B. If any of the authentications fails, or the initiator does not recognize a measurement in the responder's log L_B as that of a trusted software component, the initiator sends a BKA error message to the responder. Otherwise, the initiator computes its binding:

$$B_A = \text{SHA1}(N_B \mid K_{IKE} \mid K_{AB} \mid Y_B \mid \text{“BKA success”}). \quad (7)$$

The initiator then sends to the responder a *BKA success* message M_s containing B_A, and enables IKE's phase 2.

Processing of M_{rr} by the initiator is similar to that of M_r, except that:

(a) the initiator also computes the *responder's attestation nonce:*

$$n_B = \text{SHA1}(N_B \mid K_{AB}) \quad (8)$$

and gets from the initiator's TPM its quote Q_A containing n_B, and:

(b) instead of M_s, the initiator sends to the

Table 1: IKE latency and projected throughput with or without BKA for VPN client attestation (standard deviations represented between brackets: [σ])

Step	Unmodified IKE	IKE + BKA
Phase 1	130.7 ms [52.4]	108.1 ms [0.6]
BKA		2486.1 ms [229.4]
Phase 2	1050.2 ms [10.7]	1037.3 ms [4.8]
Total	1180.9 ms [63.0]	3631.5 ms [228.4]
CPU busy	125.3 ms [0.9]	216.3 ms [1.1]
Projected throughput	428 clients/min	277 clients/min

responder a *BKA success-response* message M_{sr} containing B_A, Q_A, and A's measurement log L_A and AIK certificate C_A.

The responder processes M_s by computing the expected value of B_A and verifying that the received B_A matches it. If so, the responder enables IKE's phase 2. Processing of M_{sr} is similar, except that, if the received B_A matches its expected value, (1) the responder also authenticates C_A, Q_A, and L_A, and (2) if all authentications succeed, and the responder identifies each measurement in the initiator's log as that of a trusted software component, then the responder enables IKE's phase 2.

BKA assumes that VPN gateways are configured to trust only client configurations that do not reveal SA secrets (including K_{AB} and K_{IKE}) to users with regular/limited accounts; otherwise, an attacker might be able to hijack the VPN connection of such a user. BKA messages are conveyed in attestation payloads of the attestation exchange. The attestation nonces n_x are one-way functions of not only the nonces N_x but also the attestation shared secret K_{AB}. Therefore, the quotes are bound to the attestation shared secret, which in turn is bound to the IKE SA's shared secret by B_x, where x is A or B.

5 EXPERIMENTAL RESULTS

This section reports the results of experiments performed to evaluate the proposed mechanisms. Reported results are averages of six measurements.

We implemented the operating system modifications described in Section 3 on FreeBSD 4.8 and installed this operating system on an IBM ThinkPad T30 computer with 1.8 GHz Pentium 4 CPU, 256 MB RAM, TPM version 1.1b, and TPM-aware BIOS. The master boot record and GRUB were modified for measuring digests and compressing them into the TPM, as necessary for authenticated boot. We measured a total boot time of 20.08 s (σ = 0.12) before and 20.15 s (σ = 0.17) after our modifications. Although TCB prelogging and file digest measuring impose overheads, they are completely dominated by other boot costs. During system operation, each file digest measurement can be cached, and need not be repeated while the file is not modified (Sailer et al., 2004a). Because TCB components change infrequently, file digest measurements can be expected to have little impact on steady-state performance. Security association root tripping affects only certain commands (e.g., login and su) and only when used for administrative accounts. Therefore, it also has negligible performance impact, as does sealing-free attestation confinement.

We integrated KAME racoon (KAME, 2005), an open-source implementation of IPsec's IKE, with BKA, as described in Section 4. We installed the modified racoon daemon in the aforementioned IBM T30 computer, which served as VPN client, and on a Dell Dimension 4550 computer with 2.4 GHz Pentium 4 CPU, 256 MB RAM, and unmodified FreeBSD 4.10. The latter computer served as VPN gateway. We used BKA with precomputed modulus (1024-bit) and primitive root.

Table 1 shows the latency for the VPN client to connect to the VPN, with or without VPN client attestation. Reported results are for IKE's main mode and authentication by RSA signatures. VPN client attestation increased latency from about 1.2 s to about 3.6 s. This latency increase is due mostly to the client's TPM. We measured that it takes 2.5 s (σ = 0.2) for the client to obtain a quote from its TPM. Even with the inexpensive TPM used, however, the total latency can be considered acceptable.

In order to evaluate BKA's impact on the VPN gateway's CPU utilization, we instrumented the operating system's idle loop and interrupt vector. The instrumented operating system uses the CPU's built-in cycle counter to measure CPU idle time, excluding interrupts. We also instrumented racoon to measure the time necessary for a VPN client to

gain access. Table 1 shows how long the VPN gateway's CPU was busy while processing a single VPN client's connection request, and the projected throughput (load for saturating the CPU). BKA reduces projected throughput from roughly 428 to 277 clients/minute. BKA's overhead is due to its use of public-key algorithms for key exchange and authentication of certificates and quotes, as well as repeated applications of SHA-1. Although BKA's overhead is significant, the projected throughput is acceptable for many applications. In case of higher loads, multiple VPN gateways can be used.

6 RELATED WORK

There are several recent industrial initiatives for creating mechanisms that verify a node's configuration before the node is accepted in a network, in a manner similar to what we propose in this paper. Cisco's Network Admission Control (NAC) (Cisco, 2005) is currently available on certain routers, but it uses a proprietary access control protocol. Microsoft has announced a similar architecture, Network Access Protection (NAP) (Microsoft, 2005a), but specifications or products are not currently available. TCG also has a similar initiative, Trusted Network Connect (TNC). Unlike NAC and NAP, TNC may use attestation to prevent forging of configuration information by attackers. Some TNC specifications have been published (Trusted Computing Group, 2005b), but they do not cover how TNC functionality will be integrated with IPsec. Moreover, operating system modifications are outside the scope of TCG's specifications.

Our TCB lists are similar to what Tripwire uses to verify host integrity (Tripwire.org, 2005). However, unlike Tripwire, we enable VPN administrators to verify host integrity remotely.

TcgLinux (Sailer et al., 2004a) is a Linux-based operating system with TPM support. It uses, however, techniques that are quite different from what we propose. Because tcgLinux does not have a TCB list, it logs and compresses into the TPM digests of *all* files that are executed, and requires shells and other programs to be modified to do the same with their security-sensitive scripts and configuration files. This design makes it harder to verify that all TCB configuration files are being measured. It also unnecessarily exposes in attestations the execution of unprivileged programs, reducing user privacy. TcgLinux has mechanisms to prevent administrative users from making system modifications that might not be detected by attestation. It does allow, however, any modifications that would be detected by attestation. TcgLinux attestations have been used to secure VPN access (Sailer et al., 2004b). However, that solution has several shortcomings. First, it does not examine how to integrate attestation with IPsec so as to avoid MITM attacks, as we have done in Section 4. Second, tcgLinux may be vulnerable to MITM attacks even if it used BKA, because it does not prevent administrative users from reading secret keys and other parameters of VPN tunnels. In contrast, we use security association root tripping to close VPN tunnels before such reading would be possible. Third, the VPN gateway has to verify fresh attestations of each client frequently. If attestation frequency is low, users may be able to connect an insecure node to the VPN long enough to cause harm. On the other hand, our experimental results suggest that high attestation frequencies could hurt the VPN gateway's throughput. In contrast, security association root tripping achieves security with a single attestation at the end of each client's IKE phase 1.

Microsoft's Next Generation Secure Computing Base (NGSCB) project (Microsoft, 2005b) uses TPMs and divides the system into trusted and untrusted halves. The untrusted half runs a conventional operating system, including file system and network protocol stack. On the other hand, the trusted half secures user credentials and keys for digital rights management. It uses storage and communication services provided by the untrusted half. NGSCB requires special CPUs with Intel's LaGrande Technology (LT). Terra (Garfinkel et al., 2003) is an operating system with similar architecture, but uses a virtual machine monitor instead of LT. It is unclear how NGSCB or Terra would be used for securing network access, because those systems cannot guarantee the integrity of the untrusted half's configuration.

The techniques described in this paper can be used to better control access not only to VPNs, but also to enterprise local area networks (LANs). The latter may be desirable, e.g., to prevent mobile computers contaminated with viruses while on a trip from propagating malware to other computers when back in the office. We discuss in another paper the integration of BKAP with PEAPv2, a protocol commonly used for access control in enterprise LANs (Xia, Kanchana and Brustoloni, 2005).

7 CONCLUSIONS

Current VPN protocols, such as IPsec, do not authenticate the configuration of users' computers. Consequently, by compromising the latter, attackers can gain unauthorized access to VPNs.

We described how attestation can be integrated with IPsec, such that only authenticated users with uncompromised computers can obtain and maintain VPN access. Attestation is a disclosure of a computer's configuration, signed by a secure coprocessor. Attestation requires mechanisms against undetected configuration changes, tampering by administrative users, man-in-the-middle attacks, and software lock-in. We described operating system and protocol modifications for solving these problems and implemented them on the open-source FreeBSD operating system and KAME racoon IKE daemon. Experiments show that our operating system modifications have negligible overhead. Experiments also show that our protocol modifications have significant impact on VPN connection latency and VPN gateway throughput. The overhead is not excessive, however, and is justified by the extra security that attestation provides.

BIBLIOGRAPHY

CISCO. *Network Admission Control*. [Online] http://www.cisco.com/en/US/netsol/ns466/networking_solutions_package.html (retrieved 8/7/2005).

DIFFIE, W. and HELLMAN, M. New Directions in Cryptography. *Transactions on Information Theory*, IEEE, 1976, 22:644-654.

FELTEN, Edward. Understanding Trusted Computing. *Security and Privacy*, IEEE, May/June 2003, pp. 60-62. [Online] http://-www.princeton.edu/~echi/ele572/Felten%20-%20Understanding%20trusted%20computing.pdf

GARFINKEL, T. et al. Terra: A Virtual Machine-Based Platform for Trusted Computing. In: 19th SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES, *Proceedings…* ACM, 2003. [Online] http://www.stanford.edu/-~talg/papers/SOSP03/terra.pdf

HARKINS, D. and CARREL, D. *The Internet Key Exchange (IKE)*. IETF, RFC 2409, 1998. [Online] ftp://ftp.rfc-editor.org/in-notes/-rfc2409.txt

KAME. *Homepage*. [Online] http://www.kame.net/ (retrieved 8/7/2005).

KENT, S. and ATKINSON, R. *Security Architecture for the Internet Protocol*. IETF, RFC 2401, 1998. [Online] ftp://ftp.rfc-editor.org/in-notes/rfc2401.txt

MAUGHAN, D. et al. *Internet Security Association and Key Management Protocol (ISAKMP)*. IETF, RFC 2408, 1998. [Online] ftp://ftp.rfc-editor.org/in-notes/rfc2408.txt

MICROSOFT. *Network Access Protection*. [Online] http://www.microsoft.com/-windowsserver2003/technologies/networking/nap/default.mspx (retrieved 8/7/2005).

MICROSOFT. *Next Generation Secure Computing Base – Technical FAQ*. July 2003. [Online] http://www.microsoft.com/technet/security/news/ngscb.mspx (retrieved 8/7/2005).

NIST. *Secure Hash Standard*. Federal Information Processing Standards Pub. 180-1, Apr. 1995. [Online] http://www.itl.nist.gov/fipspubs/-fip180-1.htm

PEARSON, S. (ed.). *Trusted Computing Platforms – TCPA Technology in Context*. Prentice Hall, 2003.

SAILER, R. et al. Design and Implementation of a TCG-based Integrity Measurement Architecture. In: USENIX SECURITY SYMPOSIUM, *Proceedings…* USENIX, Aug. 2004. [Online] http://www.usenix.org/-publications/library/proceedings/sec04/tech/sailer.html

SAILER, R. et al. Attestation-based Policy Enforcement for Remote Access. In: 11th CONFERENCE ON COMPUTER AND COMMUNICATIONS SECURITY (CCS), *Proceedings…* ACM, Oct. 2004. [Online] http://portal.acm.org/citation.cfm?id=1030083.1030125

TRIPWIRE.ORG. *Homepage*. [Online] http://-www.tripwire.org/ (retrieved 8/7/2005).

TRUSTED COMPUTING GROUP. *Homepage*. [Online] https://www.trustedcomputinggroup.org/home (retrieved 8/7/2005).

TRUSTED COMPUTING GROUP. *Trusted Network Connect*. [Online] https://-www.trustedcomputinggroup.org/downloads/TNC/ (retrieved 8/7/2005).

TRUSTED COMPUTING PLATFORM ALLIANCE. *Main Specification Version 1.1b*. [Online] https://www.trustedcomputinggroup.org/downloads/specifications/TCPA_Main_TCG_Architecture_v1_1b.zip (retrieved 8/7/2005).

XIA, H.; KANCHANA, J.; BRUSTOLONI, J. Using Secure Coprocessors to Protect Access to Enterprise Networks. *Lecture Notes in Computer Science*, 3462:154-165. Springer-Verlag, 2005. [Online] http://www.cs.pitt.edu/-~jcb/papers/net2005.pdf

YUAN, R. and STRAYER, W. T. *Virtual Private Networks: Technologies and Solutions*. Addison-Wesley, April 2001.