Leveraging the Trusted Platform Module for improving authentication systems

Arun Viswanathan

Computer Science Department, University of Southern California

Abstract— **Authentication has been the mainstay of security for many decades and is an accepted means for achieving the end goal of authorization in security. Security researchers over the years have proposed several authentication systems like Kerberos [4], Andrew Secure RPC [6], Ottway-Rees [6], CCITT X.509 [6] and others. These have proven to be extremely robust and attacker safe for a lot of practical purposes inspite of the flaws found in them. Authentication protocols have traditionally based their threat models on the assumption that the end hosts are largely secure and have focused on handling attacks against the protocol on wire. Unfortunately, with the continuous rise in threats from rootkits, keyloggers and other exotic types of malware, the threat of the end host being easily compromised and modified is no more unreal. Such malware has the potential of hiding surreptitiously inside a system and stealing user credentials like keys and passwords, thus rendering the authentication services meaningless. Solutions don't exist yet to address the above threats. Trusted Computing, an evolving computing paradigm, promises solutions to the above problems by providing a more secure and trusted environment for implementing such protocols. The paper evaluates the authentication systems in the light of new threats and proposes solutions for addressing the same using trusted computing concepts. A brief discussion on problems and possible solutions for threats against authentication in ad-hoc networks is also presented at the end. The paper focuses merely on the functionality aspects and not on the performance implications of using trusted computing.**

***Index Terms*—Trusted Computing, Security, Authentication, Kerberos, Remote Attestation, Integrity Measurement, Trusted Platform Module, Malware.**

Arun Viswanathan is a graduate student in the Computer Science Department, University of Southern California, Los Angeles, CA 90089 USA, (e-mail: aviswana@usc.edu).

I. Introduction

A. Problem scope and definition

Eugene Spafford quotes [3]:

"Using encryption on the Internet is the equivalent of arranging an armored car to deliver credit card information from someone living in a cardboard box to someone living on a park bench".

The above quote aptly describes the threat model our present computing environments are based on. A lot of work has been done on making the network secure but the hosts themselves are just like the cardboard box in the above quote. This is more so true of the

current generation authentication protocols. The threat models that the authentication protocols are based on do not account for security of the end hosts and place an implicit trust in them. Unfortunately, this assumption is very trivially rendered useless today because of the wide scale proliferation of malware like rootkits, keyloggers and other exotic malicious code. These malicious programs propagate via common attack vectors like email and browser and install automatically without the users knowledge. Once active, these malware can surreptitiously steal personal information like keys and passwords thus rendering any form of authentication or external security devices useless. A hacker can then easily impersonate a legitimate user.

Current solutions to tackle such threats rely on the user or the system administrator to configure the system securely, by selecting the right configurations and by keeping it up to date with the latest patches. Nevertheless, these solutions are very weak defenses against the growing sophistication of threats. A newly emerging computing paradigm called Trusted Computing promises to be atleast a *pewter* bullet if not a *silver bullet* in fighting these threats.

Trusted Computing, developed by the Trusted Computing Group (TCG) [1], is a set of security guidelines to mitigate the risks of participating in an interconnected world while also ensuring interoperability and protecting privacy. It is an open, vendor-neutral, industry standard specification for trusted computing building blocks and software interfaces across multiple platforms. It provides a framework with a set of primitives like sealed storage for keys, tamper resistance and remote attestation for devices to interoperate in a secure and trusted manner. One such building block is the TPM or the Trusted Platform Module [2] which is a piece of hardware capable of doing encryption, decryption, key generation, secure key storage and hashing. As described in [11], the TCG trust model is based on establishing a common assurance root and function definition for these trust characteristics. The TCG Trusted Platform Module, or TPM, serves as the starting point, or root, for this transitive trust model. The TPM as Core Root of Trust for Measurement, or CRTM, can measure additional system attributes and then later verifiably report them as a basis for determining the overall trustworthiness of a platform. This paper explores the fundamental services provided by Trusted Computing to alleviate the threats.

B. Contributions

The paper leverages the Trusted Computing concepts and proposes solutions for the problems with authentication protocols in light of newer threats. There is currently no such reported work done in this area and hence the following objectives also are the novelties of the paper.

a) *Present a generic authentication model based on existing taxonomy of authentication protocols.*
b) *Identify issues with current generation authentication protocols in the light of new and emerging threats.*
c) *Identify existing services built using trusted computing concepts and propose new ones. The paper proposes a new service called "Secure Credential Caching" which can be used to thwart some of the threats presented.*
d) *Propose solutions to the threats using Trusted Computing Services.*
e) *Extend the discussion to authentication in wireless networks which are more pertinent to the current generation.*

The paper focuses only on the functionality aspects and not on the *performance* overheads induced by the solutions.

C. Outline

The remainder of this paper is organized as follows. Section II discusses existing authentication protocols and their weaknesses in general. Section III reviews the existing threat model used by the authentication protocols and proposes an enhanced threat model in the light of newer threats. Section IV discusses the Trusted Computing paradigm. Section V discusses services built using Trusted Computing concepts. Section VI discusses threat mitigation using the services. Section VII extends the discussion to authentication problems in adhoc networks and alludes to possible solutions using trusted computing. Section VIII concludes the paper with a discussion of risks involved and open issues.

II. Overview of Authentication Protocols

An authentication protocol is a sequence of message exchanges between *principals* i.e. communicating entities, which either distributes secrets to some of those principals or allows the use of some secret to be recognized [6]. Such information is then used for subsequent communication in either validating claims of interacting principals or for encryption or decryption of communication. This section discusses the taxonomy of authentication protocols, presents a generic model based on the taxonomy and concluding with a discussion of well known attacks on authentication protocols.

A. A taxonomy of Authentication Protocols

This section summarizes the classification presented in [6] and the reader is advised to read the original report and the related references cited in the report for a more detailed exposition to the taxonomy. Other types of classification, like the classification based on Public Key methods [12] were also studied, but this was chosen because of its generality.

1. *Symmetric Key Protocols without Trusted Third Party* **-** examples: ISO Symmetric Key One pass Unilateral Authentication Protocol, Andrew Secure RPC protocol.
2. *Authentication using cryptographic check functions* **-** examples: ISO one pass unilateral authentication with Cryptographic Check Functions.
3. *Symmetric Key Protocols involving Trusted Third Party* - examples: Needham Schroeder protocol with conventional keys, Denning Saco protocol, Ottway-Rees Protocol, Yahalom, Wide Mouthed Frog Protocol.
4. *Signatures with conventional key encryption* - example: Needham-Schroeder Signature Protocol.
5. *Symmetric Key Repeated authentication protocol* - example: Kerberos V5.
6. *Public Key protocols without Trusted Third Party* - examples: ISO public key one pass unilateral authentication exchange, Diffie Hellman Exchange.
7. *Public Key Protocols with Trusted Third Party* - examples: Needham-Schroeder Public Key Protocol.
8. *Mutual Authentication Protocols* – example: Splice/AS, CCITT X.509.
9. *Miscellaneous* - example: Shamir Rivest Adleman Three pass protocol, Encrypted Key Exchange – EKE.

B. A Generic Authentication Model

Based on the above classification, the following authentication model is proposed which will be used as a reference throughout the paper. The model consists of three components as shown in Figure 1.

1. *Client (C)* - The client initiates the authentication sequence with either the AS or S to prove its identity.
2. *Server (S)* – The server offers a particular service to clients and also verifies the identities of the clients connecting to it.
3. *Authentication Server (AS)* – The Authentication Server is a Trusted Third Party trusted by both the client and server for facilitating the authentication. It may contain a repository of shared secrets shared between it and the client/server or it may also act as a certification authority for public key based systems.

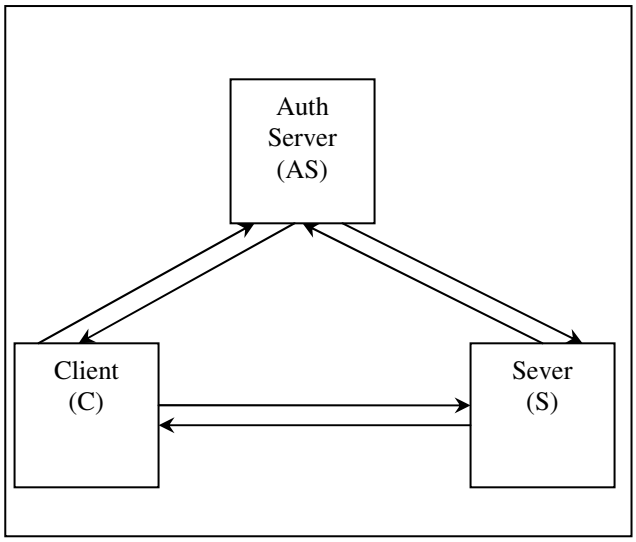

Figure 1 A Generic Authentication Model

The lines connecting the components are used to indicate the *communication channels* carrying messages in either direction. The model can be used to represent interactions amongst any of the type of protocols listed above. For example, to model the *Symmetric Key Protocols without Trusted Third Party* only the client and server components along with the bidirectional messages need be used. To model *Kerberos,* the client, server and the authentication server are used but the interaction lines between the server and authentication server are removed.

C. Known weaknesses and attacks on authentication protocols

Authentication protocols have been subjected to a variety of attacks over time, both theoretical and practical. The summary of attacks presented here are applicable to a broad category of authentication protocols.

1. *Freshness Attack* – As defined in [9], a freshness attack occurs when a message (or message component) from a previous run of a protocol is recorded by an intruder and replayed as a message in the current run of the protocol. Denning and Sacco showed that the original Needham-Schroeder symmetric key protocol suffered from this flaw and suggested using timestamps to overcome the attack.
2. *Type Flaws* – As defined in [9], a type flaw arises when the server accepts a message as valid but imposes a different interpretation on the bit sequence than the client who created it. This kind of flaw was found in the Andrew Secure RPC protocol and the Ottway-Rees protocol. The Andrew Secure RPC protocol had the nonce field and the key field of equal length which allowed an attacker to supply a nonce value as a key to the protocol thus leading to a potential compromise of the session.
3. *Multiciplicity Attacks* – As defined in [8], in a multiplicity attack an intruder is able to replay a message so as to trick the recipient into thinking that the client is in fact trying to establish two (or more) simultaneous sessions. Its been shown that the Wide-Mouthed Frog Protocol, the Denning-Sacco Shared Key Protocol, CCITT X.509 and SPLICE/AS protocol all suffer from the above attacks.
4. *Implementation dependent attacks* – The actual implementation of a particular protocol may affect the security of the protocol greatly. For example, choosing a

method to generate nonces can lead to either random or predictable values. Similarly, improper use of encryption algorithms in the context of protocols can also lead to attacks. For example, Shamir et.al.'s work in [13] was used to show that an implementation flaw in WEP made the underlying RC4 protocol easily crackable and hence rendering the WEP encryption useless.

5. *Binding Attacks* - Binding of a public key to its "correct" owner is of paramount importance in public key cryptography. A client using a server's public key must be confident that the public key is really of the server and not of an intruder. The binding attack happens when an intruder fools the client to use its public key to encrypt the messages instead of the servers.

III. Threat Model for Authentication Protocols

This section discusses the existing threat models on which systems are based and proposes a revised threat model based on a study of the latest threats.

A. Existing threat models and its weaknesses

Based on the discussion above in section II, the existing threat model can be represented using Figure 2. The black ovals represent an intruder who is capable of taking over the communication channel between the hosts in the system using the attacks described above. All protocols till date have considered only this threat vector. It is assumed that the nodes in the system are implicitly protected by other outside means like access controls and the like.

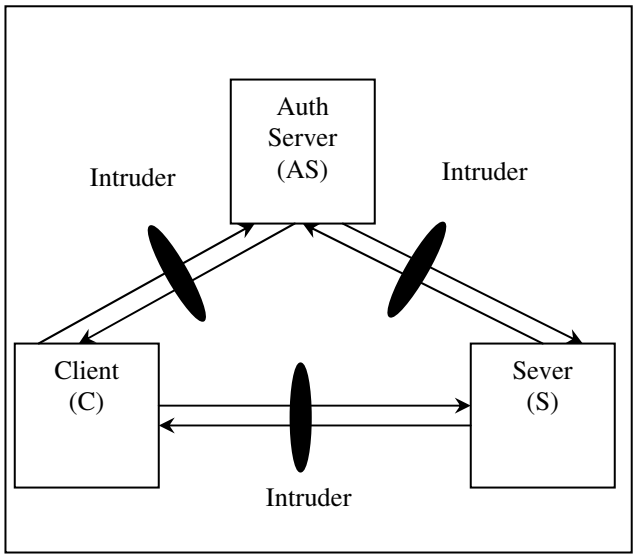

Figure 2 Existing Threat Model

It is also known that protocols like Kerberos [4] depend heavily on the security of the Authentication Server to guarantee security. Compromise of the security server can compromise the whole Kerberos network as now the intruders have access to all the keys. The protocol implicitly assumes that the System Administrator of the system must have configured the AS securely using security best practices.

As pointed out by Bellovin and Meritt in [10], the Kerberos protocol caches credentials (*tickets)* that are used during the authentication process. This caching makes Kerberos vulnerable on multi-user systems where a rogue user who manages to get a privilege escalation can compromise other user's tickets. This threat is more so applicable in today's environments due to the growth of malware like rootkits, trojans and various other exotic types of malicious code. These malicious programs propagate via common attack vectors like email and web browser and install automatically without the users knowledge. Once active, these malware can surreptitiously steal keys, cached tickets and passwords thus rendering any form of authentication or

external security devices useless as now any intruder can impersonate a legitimate user

Also, Neuman [4] points out that the Kerberos system cannot prevent against password guessing attacks. A client computer infected with a keystroke logger can easily sniff out all users passwords. A hacker can then easily impersonate as a legitimate user on the network. The authentication system is trivially rendered useless because of the ease with which these malware can propagate and install.

It can be seen that it is no safer for an authentication protocol to place implicit trust in the integrity of the end hosts. The following section revises the threat model to accommodate these latest and greatest threats.

B. A revised threat model

The revised threat model covering the threats presented in the previous section is shown in Figure 3. The model extends the threat boundary from the communication channels to the nodes in the network and does not place any implicit trust in them. The nodes are assumed to be vulnerable to intrusions inspite of the supposedly high levels of security hardening done on them by the system administrator.

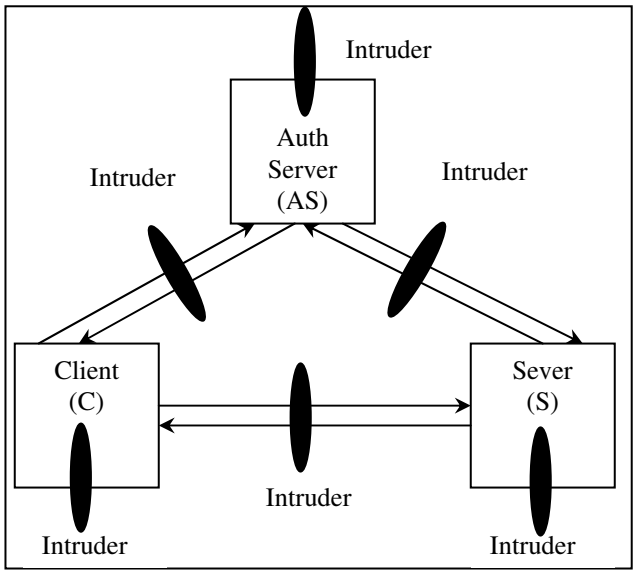

Figure 3 Revised Threat Model

With this new threat model in mind, the following new threats emerge:

1. *Nodes can be easily compromised using innocuous attack vectors like email and the web browser.*
2. *Malware like rootkits, trojans and keyloggers can be dynamically installed on a system without the knowledge of the user of the system.*
3. *Trojans can be planted in trusted programs like syslogd to perform malicious activities.*
4. *Malware has capability to sniff users passwords.*
5. *Malware has the capability to sniff cached credentials used by the authentication protocols.*
6. *Master keys or master passwords which are supposedly stored on secure areas of disk or memory are easily accessible to a malware as it has complete control of the operating system.*
7. *Techniques like using timeouts for cached credentials can be tricky because any decent timeout value is good enough for a malware to finish its job.*

IV. THE TRUSTED COMPUTING PARADIGM

Current solutions to tackle the new threats rely on the user or the system administrator to configure the system securely and by keeping it up to date with the latest patches. These defenses are nevertheless very weak against the growing sophistication of threats. A newly emerging computing paradigm called Trusted Computing promises to be atleast a *pewter* bullet if not a *silver bullet* to fight these threats. This section explains the trusted computing paradigm and explains how some of the features of trusted computing are used in practice.

Trusted Computing (commonly abbreviated TC) is a technology developed and promoted by the Trusted Computing Group (TCG). In this technical sense, "trusted" does not necessarily mean the same as "trustworthy" from a user's perspective. Rather, "trusted computing" means that the computer can be trusted by its designers and other software writers not to run unauthorized programs [16].

Trusted computing encompasses five key technology concepts, all of which are required for a fully trusted system [1].

1) *Endorsement Key* - The endorsement key is a 2,048-bit RSA public and private key pair, which is created randomly on the chip at manufacture time and cannot be changed. The private key never leaves the chip, while the public key is used for attestation and for encryption of sensitive data sent to the chip.
2) *Secure Input and Output*- Secure input and output refers to a protected path between the computer user and the software with which they believe they are interacting. Secure I/O reflects a hardware/software protected and verified channel by using checksums to verify that the software used to do the I/O has not been tampered with. Malicious software injecting itself in this path could be identified.
3) *Memory curtaining / Protected execution* – Memory curtaining extends the current memory protection techniques to provide full isolation of sensitive areas of memory — for example locations containing cryptographic keys. Even the operating system doesn't have full access to curtained memory, so the information would be secure from an intruder who took control of the OS.
4) *Sealed and Secure storage*- Sealed storage protects private information by allowing it to be encrypted using a key derived from the software and hardware being used. This means the data can be read only by the same combination of software and hardware.
5) *Remote attestation*- Remote attestation allows changes to the user's computer to be detected by authorized parties. It works by having the hardware generate a certificate stating what software is currently running. The computer can then present this certificate to a remote party to show that its software hasn't been tampered with.

A few examples of systems built using trusted computing primitives are presented below. Research has been done on using the remote attestation feature of trusted computing to provide an attestation based policy enforcement for remote access to clients [6]. In this system, remote attestation is used to verify the client integrity properties and establish trust upon the client before allowing the client to remotely access the corporate network. In another instance, trusted computing principles of remote attestation and sealed storage have been applied to provide privacy protection to RFID enabled devices [7]. Remote attestation is used by readers to prove to concerned individuals that they are running a specific

version of the reader software and that the RFID reader is not a compromised one.

V. SERVICES USING TRUSTED COMPUTING

This section starts by describing some low level TPM features and then explores various fundamental services that can be built upon these concepts. The services presented here are: *Secure Credential Caching, Integrity Measurement and Remote Attestation*.

A. Basic TPM Services

1. The TPM provides facility to create "non-migratable" storage keys inside the TPM that can be used to encrypt objects.
2. These storage keys can be created during TPM initialization for a particular OS or during application initialization. This key would be a child of the Storage Root Key which is a key per OS created in the TPM.
3. The TPM provides a set of registers called PCR which are used to hold the results of a hash operation. These hashes are typically taken over executable code in a process defined as "measurement" [11].
4. Additionally, the keys can be "sealed" by associating a set of PCR values with the keys. This makes sure that only the application that created the key can access it.
5. Using such storage keys objects can be encrypted and then stored on disk.
6. Accessing TPM objects can be done securely via authorization protocols like OIAP and OSAP defined by TCG [1] and details of which are not within the scope of this paper.

B. Secure Credential Caching (SCC)

Authentication systems use Credential Caching as a technique to speed up the protocol and free the users from supplying credentials frequently. The credentials are cached either on disk or on memory. Kerberos uses credential caching of tickets given by the *Ticket Granting Server [4]* for future use by the clients. A particular limitation of the Kerberos authentication system [4] as outlined in [10] is that Kerberos is vulnerable on multi-user systems where a rogue user who manages to get a privilege escalation can compromise other user's tickets. Though the tickets have a lifetime, given the increasing threats presented due to malware like rootkits, the caching of tickets in the clear presents a serious problem. One must note that the problem is not specific to Kerberos and the solution presented here applies to all applications that use cached credentials.

Secure Credential Caching is a secure way of caching credentials aimed at deterring the threats presented by intrusive malware. Using the set of TPM features presented in section A, a secure credential caching scheme can be designed. The assumption here is that the cache is NOT persistent. A few modifications will be necessary to make it persistent. Cache management issues like timeouts and coherency remain unchanged. The only aspects of caching that are changed are insertion and removal from the cache. The Secure Credential Caching scheme can be presented in 3 stages as follows:

1. Application Init

a. When an application starts up it creates a new storage key within the TPM using a TPM Authorization Protocol.

b. The protocol allows the application to associate a passphrase (say P) with the newly created key.

c. Additionally the key is sealed by associating the hash of the executable with the key. The hash is stored in the PCR. This procedure is performed by

the loader that loads the application. It is assumed that the loader is trusted.

d. Every time the application opens a session with the TPM to perform operations it has to provide the correct passphrase to the TPM. The TPM also checks that the PCR values are correct for the operation to succeed.

2. Object Insertion

a. Application creates a secure session with the TPM using an authorization protocol. A pass phrase is required for accessing the TPM key along with the correct state of the PCR registers. This makes sure that only the valid application can access the key and not any malware.
b. Application loads the required storage key pair into the TPM.
c. The credential to be cached is sent to the TPM to be encrypted with the loaded key handle.
d. The TPM uses the public key to encrypt the credential.
e. The encrypted blob is returned by the TPM to the application.
f. The application can now store the blob either in file or memory.

3. Object Removal

a. Application creates a secure session with the TPM using an authorization protocol. A pass phrase is required for accessing the TPM key along with the correct state of the PCR registers.
b. Application loads the required storage key pair into the TPM. The TPM accepts this only if the previous step succeeds.
c. The blob to be decrypted is sent to the TPM with the loaded key handle.
d. The TPM uses the private key to decrypt the blob.
e. The decrypted blob is returned by the TPM to the application.

The advantages of the Secure Credential Cache are as follows:

1. As the objects are always kept encrypted on disk or memory other processes cannot access those credentials without the knowledge of key.
2. As the key is sealed, a user cannot bootup from other OSes to access the keys.
3. Rootkits running in kernel can manipulate memory easily but because the cache objects are now stored encrypted the rootkit cannot read the credentials.
4. The TPM transaction is always authorized only after proper pass phrase associated with the key is provided. Also, the state of the PCRs has to be correct for the key to be released. Thus, rogue processes cannot access the keys arbitrarily.

C. Integrity Measurement

As defined in [11], the *integrity* of a program is a binary property that indicates whether the program and/or its environment have been modified in an unauthorized manner. Such an unauthorized modification may result in incorrect or malicious behavior by the program, such that it would be unwise for a remote entity to communicate with it. The Trusted Computing Group (TCG) has defined a set of standards [1] to take integrity measurements of a running system and store the result in a separate trusted coprocessor i.e. the TPM. The state of the TPM cannot be presumably compromised by a potentially malicious host system. This mechanism is called *Trusted Boot*.

The way *Trusted Boot* works is as follows:

1. On system power up, the control is first transferred to the TPM.
2. The TPM measures the BIOS by computing a SHA1 secure hash over its contents and protects the result by using the TPM.
3. This procedure is then applied recursively to the next portion of code until the OS has been bootstrapped.
4. This initial measurement is stored as the first entry in the measurement list (Fig.4).

It is the duty of the operating system to then take measurements of code that it loads. Researchers from IBM [11] have defined an Integrity Measurement Architecture (IMA) for the Linux 2.6 series of kernels. The architecture defines how the measurements are taken after the operating system has been bootstrapped. IMA measures the following:

a. Kernel modules
b. Executables and shared libraries
c. Configuration files
d. Other important input files that affect trust into run-time software stack, e.g., bash command files, Java Servlets, and java libraries.

All the individual measurements are stored in a measurement list [Fig. 4] maintained by the kernel. Additionally, the TPM takes a SHA1 over the measurement list and keeps it in a TPM PCR register. This is to detect illegal modifications to the measurement list itself.

```
 # SHA1(160bit)   File          Type
000:D6DC…D3DB n/a       [boot aggregate]
001:84AB…DA4F init          [exec]
002:9ECF…BE3D ld-2.3.2.so   [library]
003:3365…2342 libc-2.3.2.so [library]
004:A4DC…C12B bash          [exec]
…
027:2AC8…980D clock         [bash-src]
028:C0F7…9A3D hwclock       [exec]
…
070:01B3…9A1E rc            [bash-cmd]
071:CEBA…1AA4 runlevel      [exec]
072:2998…8ED4 egrep         [bash-cmd]
073:6846…B72D kudzu         [bash-cmd]
…
080:147D…8168 parport       [module]
081:F940…0115 parport_pc    [module]
…
244:D312…DA7C rc.local      [bash-cmd]
245:BB2C…AAB3 mingetty      [exec]
```

Figure 4 Sample Measurement List (from [11])

The IMA [11] ensures the following about the measurements:

a. *Fresh and complete*, i.e., includes all measurements up to the point in time when the attestation is executed,
b. *Unchanged*, i.e., the fingerprints are truly from the loaded executable and static data files and have not been tampered with.

D. Remote Attestation

TPM-based attestation represents a powerful tool for establishing the trust attributes of a system. Attestation based information about the device hardware, firmware, operating system, and applications can all be dynamically assessed to determine if the system should be trusted prior to granting a privilege (network / resource access, service, etc). Remote attestation relies on the Integrity Measurement Architecture for reporting the correct measurement values.

The *Remote Attestation* process as defined in [11] works as follows:

1. The challenging party requests the set of PCR values that define the state of the application.
2. The *attested system* first validates the authorization of the challenger and then the attested system returns its current list of measurements (in the order they where collected) and a quote from its TPM.
3. The TPM will quote its PCR registers by signing them with a 2048bit RSA signature key that was created inside the TPM and to which the public key was securely certified as belonging to this TPM.
4. On receipt of the PCR values and the quote the verifier determines i) whether the quoted PCR values are tampered with or not, and ii) whether the quoting TPM is actually the one of the attested system.
5. The verifier then computes calculates the boot aggregate by computing SHA1 (PCR0

|| … || PCR7). This is the step which verifies the boot upto the operating system.

6. It then compares it to the first measurement of the measurement list, which is supposed to be exactly this boot aggregate. If they don't match, the attestation fails.
7. For the stages above the operating system, it recalculates virtually the PCR value for the runtime measurements in the measurement list as follows:
    a. virtPCR=0
    b. Let M = value from list (initial value = boot aggregate)
    c. virtPCR := SHA1(virtPCR || M)
    d. Continue with the next measurement until the measurement list is consumed.
8. The resulting value in virtPCR must now match the value of the signed TPM PCR that was used by the attested system to protect the integrity of the measurement list.
9. If the values don't match, then the measurement list must be assumed tampered and the attestation fails. This can happen if the attested system is compromised and tries to cheat.

VI. Mitigating Threats Using Trusted Computing

The previous section developed three critical services using trusted computing and showed how these services are implemented. This section utilizes the above services to propose solutions to the threats that were outlined in section III. Existing Authentication protocols need to embed the services provided by TC into their framework to make sure that they are protected from the latest threats. The solution here is presented with respect to Kerberos but it applies equally well to others.

It is assumed that all the nodes shown in Fig. 1 have now a TPM embedded within them. The modified Kerberos protocol is now as follows:

1. The AS, C and S all run trusted software i.e. software whose measurement lists are well known to each other in advance.
2. The AS, C and S all boot up in a trusted fashion. The respective TPM's store the integrity measurements of all the software loaded using the Integrity Measurement Architecture.
3. When the client communicates to the AS for a ticket, the AS remotely attests the client to verify that the client is indeed a trusted unmodified client.
4. The client then has the option of remotely attesting the AS to make sure that the AS is a real one and an unmodified one. This step also thwarts *Man-In-The-Middle* type of attacks because if an intruder tries to impersonate an AS, the client will know immediately.
5. The client may then additionally contact another AS server called TGS in the case of Kerberos for getting a new ticket.
6. Kerberos caches this ticket into its Secure Credential Cache. The Secure Credential Cache will make sure that the tickets are accessible only to the valid application and not to a malware. Even if the tickets are persistently cached on disk they cannot be accessed by booting into any other OS.
7. The protocol then proceeds as normal and the client can now talk to the server. The client and the server can also remotely attest each other additionally to make sure that they are talking to valid entities.
8. Password or key sniffers will be thwarted because the objects are stored sealed in the secure credential cache and only valid applications can access those.
9. On the AS side, the AS can store all its key database by sealing keys using the TPM such that only it has access to the keys.

VII. Threats and Solutions For Authentication in ad-hoc networks

This section takes a quick detour into the emerging area of Wireless ad-hoc Networks where trusted computing can play a very major role in addressing issues related to authenticity of devices. It just briefly alludes to the possibility of using Trusted Computing to establish greater levels of trust between the devices in an ad-hoc network using ideas developed throughout the paper.

As stated in [12], from a security standpoint, ad hoc networks face a number of challenges. The devices that form the ad-hoc network have to interact and cooperate with each other in various ways to accomplish a task. Some devices may route data packets to other nodes or some may even act as data aggregators for other devices. The broadcast nature of the transmission medium and the dynamically changing topology add even more complications. Furthermore, the reliance on node collaboration as a key factor of network connectivity presents another obstacle. Rogue nodes induced by an attacker in such a network can sabotage the operations of the whole network rendering it useless.

As discussed in [12], in order to provide network security, authentication as such becomes the cornerstone service because it helps the nodes to trust each other before communicating. As discussed throughout the paper, authentication built on top of trusted computing services provides higher degrees of assurance. Some of the ideas discussed in this paper are applicable directly to this domain.

For example, the devices within an ad-hoc network can be equipped with tamper resistant TPM modules each containing its unique Endorsement Key which would identify every device on the network. Remote attestation can be then used by devices to determine trusted vs. untrusted nodes before transferring sensitive data. A remote management station, if possible, can also be used to continuously poll the devices to know their current measurements and thus detect rogue nodes.

VIII. End Notes and Conclusions

Trusted computing is still a very nascent technology and not ready to be consumable by the masses. The whole premise of “Trust” is based on the *Tamper Resistance* feature of the TPM. System designers who use trusted computing to propose solutions must make sure that their systems *fail-safe* in the event that the TPM is compromised.

A caveat in the measurement technology is that that measurements are one-time and there is a chance that the system might get infected after taking the measurements. Also, the measurements are always taken by the previous stage which loads the application. If a running application gets compromised by buffer overflows then the measurement would not be able to record that. This is still an open area of research. As pointed out earlier in the paper, performance overheads introduced by trusted computing can be significant and careful optimizations are required. This is another active area of research.

The paper started out with a Eugene Spafford quote alluding to the pitfalls in the present day authentication protocols. The current generation threats were analyzed and the authentication protocol threat model was revised in the light of the new threats. Three key trusted computing services were discussed to mitigate the threats in authentication protocols. Hopefully, the paper has demonstrated the potential of the trusted computing paradigm to thwart next-generation threats for authentication protocols.

REFERENCES

[1] Trusted Computing. https://www.trustedcomputinggroup.org/home

[2] Trusted Platform Module. https://www.trustedcomputinggroup.org/specs/TPM

[3] Eugene Spafford's Analogies. http://homes.cerias.purdue.edu/~tripunit/spaf-analogies.html

[4] J. G. Steiner, B. C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. In Proceedings of the Winter 1988 Usenix Conference, pages 191-201, February 1988

[5] Michael Burrows, Martin Abadi, and Roger Needham. A Logic of Authentication. Technical Report 39, Digital Systems Research Center, February 1989.

[6] J. Clark and J. Jacob. A survey of authentication protocol literature: Version 1.0. Available via http://www.cs.york.ac.uk/jac/papers/drareview.ps.gz, 1997.

[7] R. M. Needham and M. D. Schroeder. *Using encryption for authentication in large networks of computers*. Communications of the ACM, 21(12):993--999, 1978.

[8] G. Lowe. *A family of attacks upon authentication protocols*. Technical Report 1997/5, Department of Mathematics and Computer Science, University of Leicester, 1997

[9] J A CLARK and J L JACOB, Attacking Authentication Protocols, High Integrity Systems 1(5):465-474, August 1996.

[10] S. M. Bellovin and M. Merritt. Limitations of the Kerberos Authentication System. In Proc. of the Winter 1991 USENIX Conference, pages 253--267, 1991

[11] Sailer, R., Zhang, X., Jaeger, T., van Doorn, L. Design and Implementation of a TCG-based Integrity Measurement Architecture. Proceedings of the 13th USENIX Security Symposium, San Diego, CA, August 2004. 26

[12] Nidal Aboudagga, Mohamed Tamer Refaei, Mohamed Eltoweissy, Luiz A DaSilva, Jean-Jacques Quisquater, Authentication protocols for ad hoc networks: Taxonomy and research issues**,** *Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks*, October 2005

[13] S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of RC4. In *Eighth Annual Workshop on Selected Areas in Cryptography*, Toronto, Canada, Aug. 2001.

[14] R. Sailer, T. Jaeger, X. Zhang, and L. van Doorn Attestation based policy enforcement for remote access. In Proceedings of ACM Conference on Computer and Communications Security (CCS), Oct. 2004.

[15] David Molnar, Andrea Soppera, and David Wagner. Privacy For RFID through Trusted Computing. WPES 2005, November 7, 2005.

[16] Trusted Computing - http://en.wikipedia.org/wiki/Trusted_Computing