

# Distributed Trusted Computing and its applicability in Prevention of Botnets

Abijit Bej  
abej@usc.edu

*Department of Computer Science, University of Southern California*

## **Abstract –**

Trusted computing has become a buzz word in the recent times because of the growing concerns over vulnerabilities exploited in either operating system or other application level software. The need for a hardware based authorization mechanism triggered the formation of Trusted Computing Group. TCG develops and promotes open specifications for secured computing platform. Manufacturers have come up with TPM chips that can transitively calculate checksums of each piece of software that is installed on the machine thereby giving some level of confidence that the software has not been compromised. The applicability of Trusted Computing has mostly been limited to Digital Rights Management (DRM) and restricting other kinds of copyright issues. In this paper, I explore

1. How trusted computing can be used to provide Distributed Authorization of resources spread over the internet
2. How Trusted Computing can be applied to Distributed systems to prevent hijacking of computer systems by attackers in construction of Botnets, thereby to launch Distributed Denial of Service (DDoS) and other kinds of attacks.

## **Keywords**

Trusted Computing, Distributed Computing, Botnet, Authorization, Distributed Denial of Service.

## **1. INTRODUCTION**

The growth of internet has spawned new opportunities and businesses, but at the same time introduced new ways of launching

malicious attacks on target machines. Most users are not technically sound or aware of backdoor present in their home computers. Software updates, antivirus software, firewalls etc. are recommended precautions against these attacks but they have been compromised in the past and cannot guarantee a full proof security. There is certainly a need to have additional level of security mechanisms to ensure that the integrity of the system is maintained and can perform tasks only for its intended users.

The paper gives an overview of next generation Distributed Authorization mechanism using Trusted Computing. We discuss the strengths and weaknesses of this scheme, and how TC can be applied to prevent the spread of Botnets. The paper is divided into 9 sections. Section 2 discusses briefly about Trusted Computing platform, Section 3 and 4 cover Distributed Trusted computing and proposed System architecture. Section 5 gives a broad understanding of Botnets and DDoS, and Section 6 discusses about the Applicability of Distributed Trusted computing and Authorization to prevent these kind of attacks. The paper concludes with Discussion on the pros and cons of TC when applied to Distributed Computing followed by Related Work in this area (Section 8) and Future work and Conclusion in Section 9.

## **2. TRUSTED COMPUTING**

Trusted Computing is an evolving secured computing platform technology promoted by Trusted Computing Group (TCG) [2]. It sets the requirement that the system performs in an

expected way so that end user can have “trust” on the system. The hardware is loaded with a pair of encryption keys used to authenticate the genuineness of each piece of software. At its core has a Trusted Platform Module (TPM) capable of performing encryption, decryption, secure storage, key generation and also acts as the start point of transitive trust mechanism. With the use of TC (using TPM), the hardware and the software at each level will enforce the computer to behave in a consistent manner ensuring good state of the overall system.

Trusted computing consists of six key technology concepts, of which all are required for a fully trusted system, that is, a system compliant to the TCG specifications [2][4]:

- i. **Endorsement Key:** This is a 2048-bit randomly generated RSA public and private pair that is embedded in the chip at the time of manufacturing. The keys cannot be changed and the private key always remains inside the chip, while public key is used for attestation and for encryption of sensitive data sent to the chip.
- ii. **Secure I/O:** The data is authenticated at each level from the application to the secured platform down under. A secured I/O channel is established by checking a checksum of the software at every level from bottom to top.
- iii. **Memory curtaining / protected execution:** The idea here is to isolate sensitive areas of memory to provide extended memory protection so that even if the Operating system gets compromised, the attacker still does not get access to the curtained memory area that was isolated.
- iv. **Sealed Storage:** Sealed storage binds private information to platform configuration information that includes software and hardware. This restricts the data to be only read by the same combination of software and hardware as the

data is encrypted using key bound to trusted platform module.

- v. **Remote attestation:** Remote attestation is performed by authorized remote parties, like the licensor of the software to check if the user’s computer software has been tampered with. This is carried out by first, TC generating a certificate stating what software is currently running and sending it to the remote party which can then check if it has been tampered with.
- vi. **Trusted Third Party (TTP):** The entity works as an intermediary between a user and its communicating entity while maintaining anonymity and still providing a “trusted platform”. The trusted third party in most cases is the manufacturer of the software or hardware component. It has the ability to validate the received signed certificate with that of the good state of that component.

Some of the benefits from Trusted Computing include [2]:

- Protect Business Critical Data and Systems
- Secure Authentication and Strong Protection of User IDs
- Establish Strong Machine Identity and Integrity
- Ensure Regulatory Compliance with Hardware-Based Security
- Reduce Total Cost of Ownership Through "Built In" Protection

## 2.1 Trusted Computing Applications

Because of its numerous advantages and applicability, Trusted Computing have been used in some scenarios where there are no definite means to trust the underlying software. Some of the most widely used applications of Trusted Computing are [4]:

- i. **Digital Rights Management (DRM):**

The most popular use of trusted computing lies in the music industry where the Record companies enforce certain policies that are required to be fulfilled for

playing their music. Most cases it is required that media file can only be played in a curtained memory by a specific Media player that can enforce company rules. According to [2][4], it goes even further by preventing the user from making unrestricted copies of the file while it is being played. Secure I/O prevents capturing of media that is being sent to the sound system.

For circumventing such stringent requirements the computer's hardware needs to be modified or by breaking the encryption algorithm (which is extremely difficult). Though this concept can be troublesome for normal users, it has the potential to spawn new business models. As per [4], one could base a business model of renting programs for a specific time periods or "pay as you go" models. As a result, a user could download a certain media file and use it only for a certain specified amount of times or can play it only for a limited number of times. After this period either the file becomes unusable or will ask user to renew permissions.

ii. **Copyright protection in Gaming consoles:**

Trusted computing has been useful in preventing use of pirated versions of gaming software on the gaming consoles. Using remote attestation, secure I/O and memory curtaining the manufacturer can detect if the players connected to the server are running genuine piece of gaming software.

However there are "mod-chips" available in the market that can make modifications to Gaming console hardware so that prevents remote attestation of new games that user tries to play on them. Microsoft's X-Box, Sony's Playstation, Nintendo Wii have all been victims of these modifications.

iii. **Resource verification in Distributed computing:**

The resources (computational nodes) in a distributed computing environment may be geographically separated. Without remotely verifying the participants it cannot be guaranteed that the output of computation has not been forged. Research has been going on to include principles of trusted computing in Grid computing (especially in the context of Globus Alliance) [16]. Trusted Computing can be used to ensure integrity protection for resources and services by taking a system wide distributed approach.

**3. TRUSTED COMPUTING IN DISTRIBUTED COMPUTING**

The applicability of trusted computing to distributed resources has been quite limited. Part of the reason being use of conventional credential based authorization even in case of distributed systems. This provides no information about the requestor's platform and thus cannot guarantee it has not been compromised.

**3.1 Problems with Traditional mechanisms:**

According to Nagarajan et al. [1], Traditional mechanisms use Access control lists to grant access to a resource. It suits systems that have a central authority that control access policies and the requestors are known. But in case of Distributed systems, the requestor and authorizer can be completely unknown to each others. This brings the requirement for a reliable third party that authorizer trusts and that can identify the requestor. The absence of central authority, access control requirements, reliance on third parties and delegation make traditional access control unsuitable in the context of Distributed systems.

### 3.2 Need for Trusted Computing Platform:

Large scale systems are becoming increasingly prone to security vulnerabilities as only software based approaches have been used for system protection so far. As indicated in [1], a hardware based approach can be a useful in protecting information. Trusted Computing is a hardware based approach that uses a Trusted Platform Module (TPM) embedded in the hardware at time of manufacture. It provides attestation of every piece of software that is installed on the computer by checking its checksum.

### 3.3 Use of Trusted Computing in Distributed Authorization:

As described before and in [1], the typical authorization systems consider user identities, role identities, temporal, spatial and contextual information associated with the requestor for security information. The role of computing platform in authorization has been quite limited due to complexities in identification and validation of platforms when distributed. Trusted Computing can play an important role in providing computing platform information along with the request.

[1] Further describes how Property Manifest that can be used for specifying authorization policies. When hardware or software components are manufactured for trusted platforms they are provided with a 160-bit binary measurement (or hash) value that indicates the components good working state. These values are called the reference values and are represented using the TCG Reference Manifest (RM) [7] structure. It usually contains information regarding the identity, version and manufacturer of the component along with the measurement. The trusted platform module also creates a key pair (public-private) called Attestation Identity Key (AIK). The public key AIK is used to identify trusted platform

associated with the user, while the private AIK is used by TPM for signing purposes.

Integrity measurement mechanism in trusted platform: One of the most important characteristic of using TC technology is its ability to provide integrity of running hardware and software components. Processes starting from boot measure the hash of the next level processes which in turn measure for successive levels till all processes have been validated. The measurements are in the form of a 160-bit hash and are stored in Platform Configuration Register (PCR) present within TPM [1].

### 3.4 Attestation in Trusted Platforms:

The process of attestation as described in [1] is as follows; when a host wants to know about the state of trusted platform, it initiates 'attestation' process. The underlying TPM creates a 'quote' by collating the requested PCR and then signing them using private AIK. The Integrity report is then generated using platform trust service using TCG Integrity structure that point to Reference Manifests and measurement log. The host after receiving the report can validate each measurement inside log against corresponding Reference Manifest value. If all values match then the host will believe that the trusted computing is integrity proof.

Some issues that have come up with attestation are mostly because of the properties of hash functions that are used. Hash for even a slightest modified piece of software can be completely different than that of the original software. This creates problems when software patches and updates are installed on to the system. Property based attestation (described in the next section) can be an effective way to counter these problems. Attestation starting from BIOS all the up to application level, can be used to provide a hierarchical framework of attestation [12]. A detailed discussion on these issues has been covered in the Discussion section of this paper.

### 3.5 Steps in Authorization:

Usually in case of web services or distributed systems, there are two primary entities, the service requestor (SR) and service provider (SP). When dealing with distributed systems, service requestor may be the coordinator node while service provider may be the node that executes the request. As per [1], when request is made to a service provider, it has to validate two important characteristics from the request – is the service requestor the entity it claims to be and does it have the necessary privileges to use the service.

These two authentication and authorization steps may look sufficient for allowing the requestor to use the services provided by SP, but they are not enough. Consider a case where one of the machines has been compromised and modifications have been made to the operating system to instantiate a connection using authentication information that was used during the last legitimate request. There are no means for the service provider to know if it is communicating with the party it claims to be unless it has additional information about the computing platform from where the request was generated. The authorization policies may be checking the IP or MAC addresses of the requested host, but these can be spoofed and thus do not reliably represent the entity.

Trusted computing can effectively be used to address these issues raised from unreliable computing platform. Each TPM has a Attestation Identity Key (AIK) that is embedded in to the TPM that uniquely identifies a platform. This key cannot easily be spoofed unless the hardware itself is tampered with. However, the host needs to trust the manufacturer of the TPM chip which in many cases is a reliable source. The attestation in this case is provided by the hardware which can provide a better security compared to software based approaches as the hardware cannot easily

be compromised like software components, unless the adversary is in possession of it, which is an entirely different scenario.

Nagarajan et al. [1] argues that since attestation provides a mechanism to validate the measurements of components against the reference values, when they are combined with property based attestation like the one described above, one can guarantee the existence of certain security properties associated with the host. This in essence is true as it combines the authentication aspects associated with the user's account and the reliability aspects of the platform. It will guarantee that the communicating entity has not been compromised.

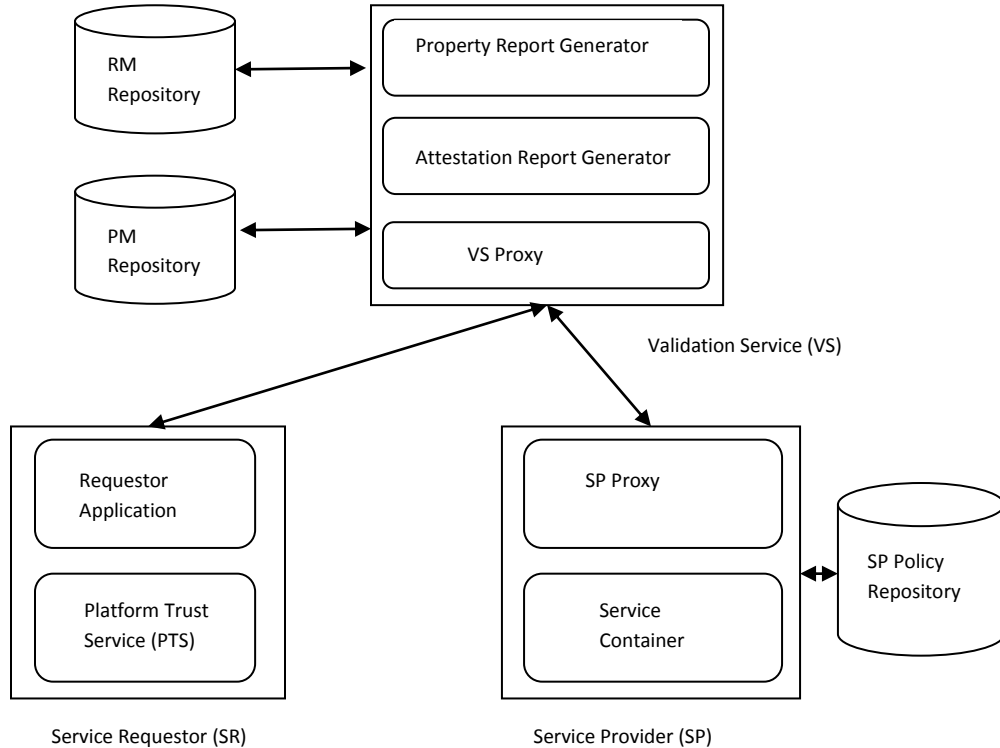
### 3.6 Property Manifest:

Property Manifest (PM) is the representation of platform's security properties [1]. It consists of PropertyID, Name, Value and Type. The Certification Authority, which is a trusted third party and is responsible for validating the integrity reports.

The Property manifest supports the mapping between components and their security properties. When a component needs to be verified for its integrity, its generated property manifest should be compared with the reference manifest. This reference manifest is obtained from a trusted third party. Property manifests are represented using XML format when used for web applications.

As per [1], the property manifests for each component will differ and will remain unique to that component. This may raise some privacy concerns as it might be detected at the other end. To keep the information about the components private, different granularity levels could be used. Depending on the level of granularity (coarse grain, for least information or fine grain, for detailed information), appropriate level of information can be revealed.

Referred Nagarajan et al. [1]



#### 4. SYSTEM OVERVIEW

As depicted in the figure above [1], there are three main entities in the system. The service requestor (SR) requests a service from a service provider (SP) which might have advertised services in some way. The validation service (VS) is a trusted third party that performs the validation of integrity report on behalf of SR or SP. It can also be used for verification of authorization policies. The RM repository stores reference manifests of different components of the trusted platform whereas the PM repository stores Property manifests. These details are made available from the component manufacturer. The authorization policies of the service provider are stored in the Policy Repository.

The platform trust service (PTS) generates the Integrity report by taking the TPM signed PCR values. The attestation report generator verifies the Integrity report. The property report

generator has the responsibility of validating the binary values of the Integrity Report with that of the Reference Manifests.

##### 4.1 Working of the overall system:

When the requestor initiates a request, the integrity report generated by Platform Trust Service is sent along with the request. The VS-proxy after receiving the request invokes Attestation Report Generator of the VS through VS-Proxy. Attestation report of the Integrity Report is generated by ARG using Reference manifests from RM repository. The Property Report Generator after receiving the Attestation report validates the binary measurements by comparing them with Property manifests from Property Manifest Repository. The signed report is sent back the SP-Proxy which in turn sends the report along with the original request from the requestor to its own decision engine (an XACML engine). This verifies access control policies that

are stored in the Policy repository of the Service Provider. It then takes a decision whether to allow or deny the request.

The above model describes the working of Pull Model [1]. The second model described in it is “Push Model” where the initial request along with Integrity report is sent to the SV-Proxy. The third and final model called “Delegation Model” where the Service Provider delegates all the work on its behalf to the Validation Service.

The same kinds of authentications step would be required if the SR wants to validate SP (having a TPM) through trusted third party VS.

## **5. BOTNETS**

### ***5.1 What are Botnets?***

Botnet is a set of compromised machines located at geographically dispersed locations under control of an adversary (or control bot) with the owners of those machines being unaware of it. These compromised machines are also termed as “Zombie”. The software for these botnets are created and spread with mal-intent for attackers own advantage. The term ‘bot’ refers to these machines because of their autonomous and automatic behavior. They are created and operated by criminal entities for spamming, Distributed Denial of Services (DDoS) among other kinds of attacks.

### ***5.2 What is Distributed Denial of Service or DDoS?***

If a machine launches some kind of malicious software attacks (as example spamming, excessive download of important files etc.) on a target system in order to block resources or to prevent it from legitimate use, the kind of attack is termed as Denial of Service (DoS) attack. These have comparatively definite pattern and can be detected by the target system using conventional monitoring software. Once detected, the IP or MAC of the attacker can be blocked for future.

However, if the attack is launched from multiple sources, it becomes extremely difficult for the target system to differentiate it from legitimate users. The kind of attack to overwhelm the target system so as to stall it from normal operation is termed as *Distributed Denial of Service (DDoS)* attack. The intention of doing this is to make the system unavailable to its intended users. As pointed out in [8], there is a whole underground economy run by the botnet attackers for financial, commercial or personal gains.

A large part of the botnet machines usually belong to Home users with broadband connections, high speed educational computers/servers or others that do not have effective mechanisms to guard their systems from network attacks. They maintain a low level communication with the control node which is in a different administrative domain.

### ***5.3 How do they spread or get installed in first place?***

Some of the know means of spreading botnets is through IRC chats, drive by downloads, worms, Trojan horses, command and control infrastructures or by exploiting web browser vulnerabilities [21]. In all of the above cases, the user is completely unaware of the new piece of software that gets installed on their computer. There have been cases where the bot software comes attached to a legitimate software that user downloads through the internet. When user tries to install it, the malicious code also gets through and installs itself. Neither the antivirus software nor anti-Trojan software will be able to effectively guard against this.

## **6. PREVENTIVE METHODS**

### **6.1 Conventional preventive measures against DDoS:**

A large majority of solutions to prevent DDoS are based on monitoring network traffic in order to filter out malicious traffic. It may seem

reasonable but absence of a clear distinction between legitimate and malicious traffic results in large number of false positives. The way to detect DDoS is to look for a sudden increase in traffic. There have been cases where it later turned out to be legitimate traffic during post analysis. But so far this has been used widely in the absence of effective ways to detect DDoS. Prevention from DDoS based attacks is still under research.

## **6.2 Use of Trusted Computing to Prevent Botnets and thereby DDoS:**

The first step towards incorporating principles of trusted computing in the context of Distributing computing is to have the TPM module embedded in motherboard of each node or computer in the system. There are several advantages to this. Firstly, any new software that needs to get installed on the system will have to go through the authorization mechanism involving the Platform Trust Service (PTS) of the requestor and a trusted third party (Verification Service) to verify the Integrity report. Any changes if made to the software in question can be checked and verified by the trusted third party (usually the manufacturer of the component under test) and can be detected by validating its Reference manifests and Property manifests as described in Section 3 and 4.

Secondly, in a trusted platform, the disk, Operating system, printer, web browser and other components negotiate an encryption scheme so as to ensure a secured path of communication. This ensures that the data can nowhere along the line get exposed to the adversary.

## **7. DISCUSSION**

Trusted Computing as described before has the advantage of ensuring security right from the hardware level that cannot be easily compromised as compared to software. There are however some potential problems, as an example, for every new software patch or update, there will be an entirely different hash generated. At one

end it will be extremely difficult for an adversary to know the pattern of the new checksum, but it will also place some challenges to the trusted platform to validate the integrity of the software after update. As described in Section 4, Property based validation proves out to be advantageous as they validate the properties rather than hash values, but still one has to design the mechanism allowing some level of flexibility for changes due to software updates and patches. But then this completely defeats the purpose of Trusted Computing that works by exact attestation of components.

There have been implementations of trusted computing in the field of Digital Rights Management (DRM), but it still has not picked up that well. TC has remained controversial and some of its opponents have referred it as “treacherous computing” [4]. They claim that apart from preventing violation of copyright issues for Music companies they do not provide any kind of security to the end user.

Third issue with trusted computing platform is related to how much “trust” a user places on TPM chip manufacturer. The manufacturer of the TPM would certainly not expose the encryption keys embedded in the TPM chip in order to prevent itself from prosecution and causing potential risk to its business, but it still remains a loop hole in the process. If an adversary follows the embedded keys for each series of production chips, it might be possible for him to come up with the algorithm for key generation. Though it has a remote possibility, but it is possible.

Fourth, Trusted computing is still a growing field. It still does not have widespread awareness and reach. The technology is promoted and developed by Trusted Computing Group along with a handful of companies. It certainly requires heavy participations both from industry and academia to promote and endorse the technology.

Another potential problem is related with privacy. Over the course of time it is possible that the challenging host (service provider in our case) learns about the components and states of the requestor's platform through the requests that it receives. One can argue that property based attestation can only reveal abstract information about the components. But they have a high probability of revealing more information either by "reverse engineering" or by "deeper inspection". A proposed solution to this problem is to use Zero Knowledge Protocol [20].

## 8. RELATED WORK

Some of the early work that I came across during my research was to do with "Decentralized Trust Management" by Blaze et al. It shows that the authorization credentials are closely related to permissions and they are created, administered and distributed by Trust Management system. Nagarajan et al. work on Trusted Computing for Distributed Authorization using Web Services describes a noble approach of achieving distributed authorization using trusted computing. They introduce the notion of Property manifests in solving authorization problem in distributed computing. Other approaches that I came across during my research were KeyNote [7], REFEREE [15] and IBM's Trust Management framework. The analysis carried out by Friess et al. on Black Market Botnets gives an insight into the way the underground market for botnet operates. There is a brief mention about applicability of trusted computing but only as a preventive measure for protecting private data through line of encryption scheme.

## 9. FUTURE WORK AND CONCLUSION

The requirements mentioned in this paper throw light in an unexplored avenue. The use of trusted computing should not only be limited to Digital Rights managements but can effectively be used in botnet prevention. The mechanism

described in the paper is not a complete list and requires further work to incorporate trusted computing into distributed systems, but it gives a motivation to think. More research and careful implementation is definitely needed in order to make it a reality.

## ACKNOWLEDGEMENT

I am thankful to Dr. Clifford Neuman, for encouraging me to write a paper on this topic with his positive feedback on my proposal. The idea to do research on this topic came during one of his "Security Systems" course lectures at University of Southern California when we discussed about Botnets, Distributed Denial of Services and its potential problems. It was during the lecture on Trusted Computing that I started thinking about doing my research on these two topics.

## REFERENCES

- [1] A. Nagarajan, V. Varadharajan, M. Hitchens, S. Arora. On the Applicability of Trusted Computing in Distributed Authorization Using Web Services. In proceedings of the 22nd annual IFIP WG 11.3 working conference on Data and Applications Security (2008).
- [2] Trusted Platform Module. <https://www.trustedcomputinggroup.org/specs/TPM>
- [3] B. Clifford Neuman. Proxy-Based Authorization and Accounting for Distributed Systems. In Proceedings of the 13th International Conference on Distributed Computing Systems, pages 283-291, May 1993.
- [4] Trusted Computing. Wikipedia. [http://en.wikipedia.org/wiki/Trusted\\_Computing](http://en.wikipedia.org/wiki/Trusted_Computing)
- [5] Bishop, M. Computer Security - Art and Science. Addison-Wesley, 2003.

- [6] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," Tech. Rep. 96-17, February/August, 1996.
- [7] Blaze, M., Feigenbaum, J., Ioannidis, J., Keromytis, A.: The KeyNote Trust-Management System Version 2 (RFC 2704). Internet Engineering Task Force (September 1999)
- [8] N. Friess, J. Aycock. Black Market Botnets. TR 2007-873-25, July 2007
- [9] Microsoft's Next-Generation Secure Computing Base. <http://www.microsoft.com/resources/ngscb/default.msp>
- [10] J. G. Steiner, B. Clifford Neuman, and J.I. Schiller. Kerberos: An Authentication Service for Open Network Systems. In Proceedings of the Winter 1988 Usenix Conference. February, 1988. (Version 4)
- [11] Trusted Computing Benefits. [http://www.trustedcomputinggroup.org/trusted\\_computing/benefits](http://www.trustedcomputinggroup.org/trusted_computing/benefits)
- [12] R. Sailer, T. Jaeger, X. Zhang, and L. van Doorn, "Attestation-based policy enforcement for remote access," in CCS '04: Proceedings of the 11th ACM conference on Computer and communications security. New York, NY, USA: ACM, 2004, pp. 308-317.
- [13] Yoshihama, S., Ebringer, T., Nakamura, M., Munetoh, S., Maruyama, H.: WSAttestation: Efficient and Fine-Grained Remote Attestation on Web Services. Technical report, IBM Research (February 2005)
- [14] Poritz, J., Schunter, M., Herreweghen, E.V., Waidner, M.: Property Attestation: Scalable and Privacy-Friendly Security Assessment of Peer Computers. Technical report, IBM Research (May 2004)
- [15] Chu, Y.-H., Feigenbaum, J., LaMacchia, B., Resnick, P., Strauss, M.: Referee: Trust Management for Web Applications. World Wide Web J. 2(3), 127–139 (1997)
- [16] J. Zhan, H. Zhang, F. Yan,: Building Trusted Sub-domain for the Grid with Trusted Computing, Information Security and Cryptology: Third SKLOIS Conference, Inscrypt 2007
- [17] Lionel Litty and David Lie. Manitou: A Layer-Below Approach to Fighting Malware. In Proceedings of the Workshop on Architectural and System Support for Improving Software Dependability (ASID), held in conjunction with ASPLOS 2006. Pages 6-11. October 2006.
- [18] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, "Terra: a virtual machine-based platform for trusted computing," in Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03). New York, NY, USA: ACM Press, 2003, pp. 193-206.
- [19] Tal Garfinkel , Mendel Rosenblum , Dan Boneh. Flexible OS Support and Applications for Trusted Computing. IN 9TH HOT TOPICS IN OPERATING SYSTEMS (HOTOS-IX (2003)
- [20] Chen, L., Landfermann, R., Löhner, H., Rohe, M., Sadeghi, A.R., Stubble, C.: A Protocol for Property-Based Attestation. In: STC 2006: Proceedings of the first ACM workshop on Scalable Trusted Computing, New York, NY, USA, pp. 7–16 (2006)
- [21] Botnets - <http://en.wikipedia.org/wiki/Botnet>