“BAD LOGIN” as a Mechanism for Secure Code Execution and Behavior Analysis

Sudhirkumar Patel *, Shrikant Thakar **, Mahendrakumar Patel ***
MCA, Ph.D (Pursuing) Mewar University Chhittogarh (Rajasthan)

* *Asst. Prof & Head,* D.L.Patel Inst. of Mgt & Tech. MCA College, Himatnagar, Gujarat, India.
Affiliated to Gujarat Technological University (G.T.U) Ahmedabad.
* patel_sudhirkumar@gtu.edu.in, sudhirkumar.patel@gmail.com

** & *** *Asst. Prof*, I.N.S.B. Institute of Information Technology and Management Studies, BCA College, Idar, India.
Affiliated to Hemchandracharya North Gujarat University, Patan.
** shrikant.thakar@gmail.com
*** mmpatel_1972@yahoo.co.in

INTRODUCTION

Today, businesses and individuals are entrusting progressively greater amounts of security-sensitive data to computers, both their own and those of third parties. To be worthy of this trust, these computers must ensure that the data is handled with care (e.g., as the user expects), and protected from external threats. Unfortunately, today’s computer platforms provide little assurance on either front. Most platforms still run code designed primarily for features, not security. While it is difficult to measure precisely, multiple heuristics suggest that code quality has improved relatively little with respect to security. For example, the majority of coding projects on Source Forge employ type-unsafe languages (e.g., C or C++) [1]. The National Vulnerabilities Database [2] catalogues thousands of new software vulnerability reports each year, and recent studies indicate that over 25% of US computers are infected with some form of malicious software [3].

These vulnerabilities are particularly troubling when coupled with the increasingly hostile environment to which computers (and users) are exposed. Indeed, the lucrative and difficult-to-prosecute crimes that computers facilitate have given rise to a burgeoning criminal underground in which sophisticated, Financially-motivated attackers collaborate to monetize exploited computers and stolen user data [4]. These ne’er-do-wells can employ automated, turn-key packages to attack thousands of potential victims every second [5]. The victim computers are then often formed into coordinated “botnets” of tens or hundreds of thousands of machines and used to send spam or launch Distributed Denial of Service (DDoS) attacks [6,7].

As a result, from the moment a user digitizes her data, it is under constant assault from all sides. Malicious software on the user’s computer may snatch up the user’s private data and ship it to foreign lands. If the user entrusts her data or computations to a remote service, then the remote computers may be subject to all manner of physical attacks, as surveyed in Section. Furthermore, experience demonstrates that remote workers will attempt to return forged results even when the only payoff is an improvement of their standings in an online ranking [8]; when there is a potential to profit from such skullduggery, workers’ temptations can only increase.

Sadly, these attacks on user data succeed all too often. They contribute to the over 3.6 million U.S. households that were victims of identity in the year 2004 alone [9]. They also undermine users' trust in electronic systems, and hence inhibit both current and future systems. For example, a 2005 Consumer Reports survey found that 29% of consumers had cut back on – and 25% had stopped – shopping online due to fears of fraud and identity theft [10].

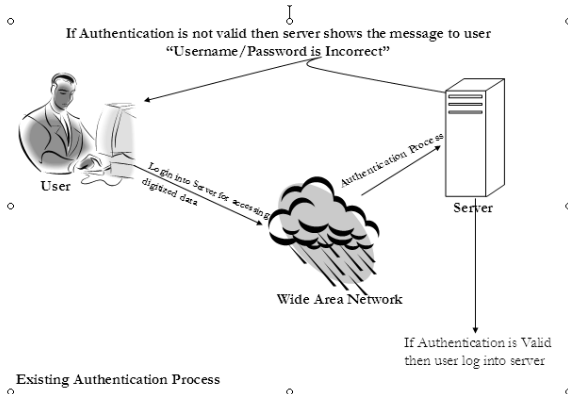
If Authentication is not valid then server shows the message to user
"Username/Password is Incorrect"
User
Login into Server for accessing digitized data
Authentication Process
Server
Wide Area Network
If Authentication is Valid then user log into server
Existing Authentication Process

Fig-1 shows the normal login system for any user. In normal login system user use username and password. If password or user id is wrong then system show the message to user "you username or password is wrong" and redirect to login page so user enter valid username and password. In this scenario may hacker immediately come to know password is not valid or valid and he try different password with same user-id and hack the user account based on trial and error method.

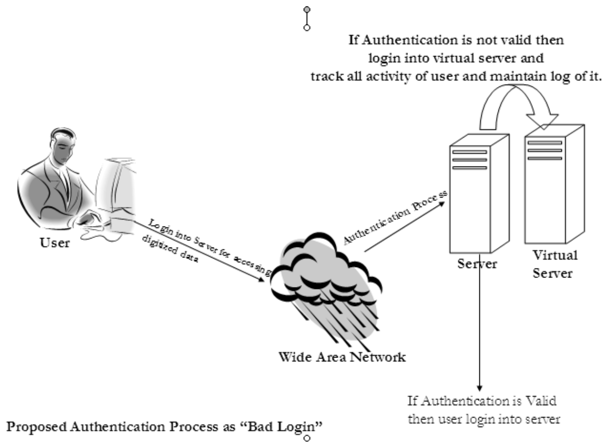
If Authentication is not valid then
login into virtual server and
track all activity of user and maintain log of it.
User
Login into Server for accessing digitized data
Authentication Process
Server
Virtual Server
Wide Area Network
If Authentication is Valid then user login into server
Proposed Authentication Process as "Bad Login"

Fig 2 "Proposed Authentication System"

As show in fig-2 we are implement new authentication system where normal user will track the hacker activity. In our proposed authentication system, user enter user-id and password if password is not valid then we log user into virtual system and track the all activity of that user and maintain the log of it. During the next valid login user will prompt with last bad login activity log so user come to know someone try to do un-authentication work in his system.

Bad login is concept which is used for developing trust platform module (T.P.M) on user local computer and secures the digital data of user.

LITERATURE REVIEW

Leveraging Secure Code Execution to Improve Network Protocols

If we can provide secure code execution on endhosts, the next frontier is to examine how such trust can be used to improve the performance and efficiency of network applications. In other words, if endhosts (or at least portions of each endhost) can be trusted, then network infrastructure no longer needs to arduously and imprecisely reconstruct data already known by the endhosts.

Through the design of a general-purpose architecture we call Assayer [11], we explore the issues in providing trusted host-based data, including the balance between useful information and user privacy, and the tradeoffs between security and efficiency. We also evaluate the usefulness of such information in three case studies: spam identification, distributed denial-of-service attack mitigation, and super-spreader worm detection.

To gain insight into the performance we could expect from such a system, we implement and evaluate a basic Assayer prototype. Our prototype requires fewer than 1000 lines of code on the endhost. Endhosts can annotate their outbound traffic in a few microseconds, and these annotations can be checked efficiently; even packet-level annotations on a gigabit link can be checked with a loss in throughput of only 3.7-18.3 %, depending on packet size.

Secure Code Execution despite Untrusted Software and Hardware

With Flicker, we assume that the user’s computer is physically secure. To generalize Flicker’s results, we need techniques to establish trust in code execution when even the hardware is untrustworthy. This scenario is particularly compelling as the growth of “cloud computing” and the proliferation of mobile devices contribute to the desire to outsource computing from a client device to an online service. In these applications, how can the client be assured that the secrecy of her data will be protected? Equally importantly, how can the client verify that the result returned is correct, without redoing the computation?

While various forms of homomorphic encryption can provide data secrecy [12,13], the results in demonstrate that we can efficiently verify the results of arbitrary tasks (abstracted as function evaluations) on a computational service (e.g., in the cloud) without trusting any hardware or software on that system. This contrasts with previous approaches that were inefficient or that could only verify the results of restricted function families. To formalize secure computational outsourcing, introduces the notion of verifiable computing [14]. Abstractly, a client wishes to evaluate a function F (e.g., sign a document or manipulate a photograph) over various, dynamically selected inputs x1, . . . , xk on one or more untrusted computers, and then verify that the values returned are indeed the result of applying F to the given inputs. The critical requirement, which precludes the use of previous solutions, is that the client's e.ort to generate and verify work instances must be substantially less than that required to perform the computation on her own. Drawing on techniques from multi-party secure computation, as well as some recent developments in lattice-based cryptography, we present the first protocol for verifiable computing. It provably provides computational integrity for work done by an untrusted party; it also provides provable secrecy for the computation's inputs and outputs. Moreover, the protocol provides asymptotically optimal performance (amortized over multiple inputs). Specifically, the protocol requires a one-time pre-processing stage which takes O(|C|) time, where C is the smallest known Boolean circuit computing F. For each work instance, the client performs O(|m|) work to prepare an m-bit input, the worker performs O(|C|) work to compute the results, and the client performs O(|n|) work to verify the n-bit result.

This result shows that we can outsource arbitrary computations to untrusted workers, preserve the secrecy of the data, and efficiently verify that the computations were done correctly. Thus, verifiable computing could be used, for instance, in a distributed computing project like Folding@home [15], which outsources the simulation of protein folding to millions of Internet users. To prevent cheating, these projects often assign the same work unit to multiple clients and compare the results; verifiable computing would eliminate these redundant computations and provide strong cryptographic protections against colluding workers.

Thus, even without secure hardware, these results show that we can leverage a user's trust in one device to verify (and hence trust) the results of computations performed by an arbitrary number of remote, untrusted commodity computers.

OBJECTIVE AND SCOPE

A Vision for a Better World

I envision a future in which average computer users can easily and securely use their computers to perform sensitive tasks (e.g., paying bills, shopping online, or accessing medical records), while still retaining the flexibility and performance expected of modern computers. Users will regard a computer attack not as a disaster that empties bank accounts or destroys documents, but as a minor annoyance; like a blown electrical fuse, it will be easy to detect and simple to remedy. Providing security as a largely invisible default will allow the information age to finally reach its true potential: users will submit their personal information to online sites, not blindly or with trepidation, but with confidence that it cannot be stolen or misused; businesses and consumers will feel perfectly secure outsourcing work to computational services; and remote, web-based applications will provide the same level of privacy, security, and availability that native applications do.

Achieving this goal will require advances on many fronts: better programming languages, better operating systems, better network protocols, and better definitions of security. More fundamentally, however, we must enable both computers and users to make accurate, informed trust decisions. After all, even if software does improve, we must be able to determine which systems employ the new and improved software! This applies both to users and to network components. In other words, it is critical that a user be able to judge whether a system (either local or remote) should be trusted before she hands over her sensitive data. Similarly, if a network element (e.g., a router) can trust information from an end host, then numerous protocol optimizations become possible.

In this work, we focus on a definition of trust similar to the definition of a Nash Equilibrium; for example, to trust an entity X with her private data (or with a security-sensitive task), a user Alice must believe that at no point in the future will she have cause to regret having given her data (or entrusted her task) to X. As a result, this dissertation examines techniques that provide firm evidence on which to base such a belief. As an additional constraint, we concentrate on average users and commodity systems, rather than on advanced users, special-purpose computers, or highly constrained environments (such as those found within the military).

Alas, previous efforts to construct trustworthy systems “from the ground up” have proven difficult, time-consuming, and unable to keep pace with the changing demands of the marketplace. For example, the VAX VMM security kernel was developed over the course of eight years of considerable effort, but in the end, the project failed, and the kernel was never deployed. This failure was due, in part, to the absence of support for Ethernet – an emerging and highly popular feature considered critical by the time the kernel was completed, but not anticipated when it was initially designed. Thus, such efforts have typically been doomed, and their methods have not been adopted into the mainstream of software development.

The concept of bad login is about to prevent the hacker to hack digital data through the Trail & Error method. Today we all know that, Internet user increasing day by day but they do not aware about hacker activity.

People who generally not aware about HACKER they will set weak password like birth-date, mobile-number, mom-dad name, school name etc. which is belong to user profile and easy available on social engineering website. Hacker will easily track the user activity by getting his password via some mechanism we called as HACKING.

Sometime HACKER use trial and error method, where Hacker enter different password with same user-id. PASSWORD entered by HACKER is relative information of user (birth-date, mobile-number, mom-dad name, school name etc).

RESEARCH METHODOLOGY

We can really say with certainty about hacker and hacker activity will continue to be unpredictable. And even more important using IP address based cyber space tracing.

The research process will include following phases for accomplishing the objective of research to study and analyze various prevention mechanism with an attempt to suggest a generalized prevention TPM (Trusted Platform Module) model, which will cover the advantages of the available process used to maintain security. In this research we propose a conceptual framework and an analytical methodology which might be used to comparatively analyze the prevention mechanism.

REVIEW EXISTING INFORMATION

As a first step, a thorough review of the secondary data available is conducted. This review should be comprehensive and seek to cover as wide a range of different data sources as possible. As a result of the review of existing information, an overview report is prepared. This report is structured according to the analytical framework and forms the basis of the final report.

RESEARCH DESIGN

Phase-1: In phase-1 we want indentify those servers who are responsible for un-authentication login. The study phase will emphasis on studying the various tracing mechanisms. This will be done with the help of various protocols and IP address base tracing.

Phase-2: Analysis of various activity of hacker like how the virus is generated, how they are spread and various effects of it. In this phase an analysis will be made to categories the prevention measure.

Phase-3: After collecting the analyzed data form phase-1 and phase-2 we want to design **Building up from a Firm Foundation** TPM (Trusted Platform Module) Rather than starting over, we can design techniques that allow users to leverage the trust they have in one device to securely use another device or service. As we describe in more detail below, we start from the assumption that the user has some small, portable, canonically trusted device, such as a special-purpose USB device or cell phone. The question of how this initial trust is established is outside the scope of this work, Given this canonically trusted device, we analyze the difficulties that arise when attempting to use it to establish trust in an ordinary computer, particularly one equipped with the latest security hardware enhancements. However, solving this problem merely reveals that the user’s computer is likely running millions of lines of potentially buggy (and hence untrustworthy) code. To enable secure functionality on such a platform, we design and implement the Flicker architecture, which provides strong security protections on demand, while still allowing users to enjoy the features and performance they have come to expect from general-purpose computers.

Given this more secure architecture for an individual computer, we next examine. The question of how we can extend that trust into the network. In other words, how can we improve the security and/or performance of network protocols if we can verify that at least some portion of the code on an end host can be trusted? Finally, we consider the protections we can offer to outsourced computations. In other words, if the user trusts her own machine, how can she extend that trust to computations done by a remote entity (for example, as part of a cloud computing service)? In particular, what guarantees can we provide with regards to the secrecy and integrity of the computations if we trust neither the software nor the hardware of the remote party?

BOOTSTRAPPING TRUST IN A COMMODITY COMPUTER

Initially, we focus on the problem of allowing a user to bootstrap trust in her own personal computer. This problem is fundamental, common, and should be easier than other potential scenarios. In other words, if we cannot establish trust in the user's computer, we are unlikely to be able to establish trust in a remote computer. When working with her own computer, the user can at least be reasonably certain that the computer is physically secure; i.e., an attacker has not tampered with the computer's hardware configuration. Such an assumption aligns quite naturally with standard human intuition about security: a resource (e.g., a physical key) that an individual physically controls is typically more secure than a resource she gives to someone else. Fortunately, the physical protection of valuable items has been a major focus of human ingenuity over the past several millennia.

If the user's computer is physically secure, then we can make use of special-purpose hardware to support the user's security decisions. While a full-blown secure coprocessor, such as the IBM 4758, might be appealing, cost and performance considerations make deployment difficult. However, for the last few years, many commodity computers have come equipped with a Bad login Trusted Platform Module (TPM) that can be used for a variety of security-related purposes.

Unfortunately, at present, no standard mechanism exists for establishing trust in the TPM on a local machine. Indeed, any straightforward approach falls victim to a cuckoo attack. In this attack, the adversary extracts the private keys from a TPM under his physical control. These keys can be given to malicious software present on the user's local computer, in order to fool the user into accepting reassurances from the adversary's TPM, rather than her own.

We propose a formal model for establishing trust in a platform. The model reveals the cuckoo attack problem and suggests potential solutions. We survey the usability challenges entailed by each solution, and suggest preferred approaches to enable a user to bootstrap trust in the secure hardware on her personal computer.

CONSTRUCTING THE EXPERIMENT

There are various aspects in our experiment when we want to trace IP address of any machine that is responsible for generating a spam, bulk e-mail and e-mail with viruses. We will try to ensure that the experiment is carried out properly and that the result reflects the real world, in the best possible way.

IMPLEMENTATION

In this phase the proposed model will be implemented to achieve the objective of the system so that the model can be referred to stop the hacker attack **(Bad Login)**. This phase use the document from the design phase and requirement document from the analysis phase to obtain and use the model.

SOLUTION

Unfortunately, at present, no standard mechanism exists for establishing trust in the **T.P.M (Trusted Platform Module)** on a local machine. Indeed, we want to build application or web service **(BAD LOGIN)** which is provide next level security to user and prevent the hacker to hack the user data and we can Analyze the hacker behavior. With help of BAD LOGIN user also track the hacker activity.

CONCLUSIONS

Motivated by the trend of entrusting sensitive data and services to insecure computers, we develop techniques that allow a user to extend her trust in one device in order to securely utilize another device or service. A key constraint is our focus on commodity computers, particularly the need to preserve the performance and features expected of such platforms. Using a logical framework, we analyze the perils of establishing trust in a local computer equipped with commodity (i.e., low-cost) security hardware and provide guidance on selecting a mechanism that preserves security while minimizing changes to existing computer designs. To make the results of such an interaction meaningful, we develop the Flicker architecture for providing an on-demand, secure execution environment for security-sensitive code. Building on recent improvements in commodity CPUs, Flicker provides strong isolation, reporting, and state preservation for security sensitive code. Because it runs only on demand, Flicker imposes zero overhead on non-security-sensitive code.

BOOK REFERENCE

1. J. Howell, J. R. Douceur, J. Elson, and J. R. Lorch. Leveraging legacy code to deploy desktop applications on the web. In Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI), Dec.2008 (Referenced on page 14)

2. NIST Computer Security Resource Center (CSRC). National vulnerability database. http://nvd.nist.gov/home.cfm.

3. Organization for Economic Co-operation and Development. Malicious software (malware): a security threat to the internet economy. http://www.oecd.org/dataoecd/53/34/40724457.pdf. (Referenced on page 14**)**

4. J. Franklin, V. Paxson, A. Perrig, and S. Savage. An inquiry into the nature and causes of the wealth of internet miscreants. In Proceedings of the ACM Conference on Computer and Communications Security (CCS).

5. Sophos. Do-it-yourself phishing kits found on the Internet, reveals Sophos.http://www.sophos.com/spaminfo/articles/diyphishing.html.

6. T.Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freiling. Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm. In Proceedings of the USENIX Workshop on Large-Scale Exploits and Emergent threats (LEET),

7. E. Messmer. Downadup/conflicker worm: When will the next shoe fall? Network World.

8. D. Molnar. _e SETI@Home problem. ACM Crossroads,. (Referenced on pages 15, 92 and 158.)

9. Department of Justice, Bureau of Statistics. Press release: Identity theft ¦. http://bjs.ojp.usdoj.gov/content/pub/press/it04pr.cfm, (Referenced on page 15.)

10. Princeton Survey Research Associates International. Leap of Faith: Using the Internet Despite the Dangers (Results of a National Survey of Internet Users for Consumer Reports Web Watch). http://www.consumerwebwatch.org/pdfs/princeton.pdf,

11. B. Parno, Z. Zhou, and A. Perrig. Help me help you: Using trustworthy host-based information in the network. Technical Report CMU-CyLab-09-016, Carnegie Mellon University, Cylab, Nov. 2009. (Referenced on page 19.)

12. C. Gentry. Fully homomorphic encryption using ideal lattices. In Proceedings of the ACM Symposium on the theory of Computing (STOC), (Referenced on pages 20, 50, 160, 161, 165, and 166.)

13. M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In Proceedings of EuroCrypt, June 2010. (Referenced on pages 20 and 50.)

14. R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computation: Outsourcing computation to untrusted workers. In Proceedings of CRYPTO, Aug. 2010. (Referenced on page 20)

15. Pande Lab. The folding@home project. Stanford University, http://folding.stanford.edu/. (Referenced on pages 21,74,158,162 and 181)

WEB REFERENCE

http://www.esecurity.org.in/online-security-issues.html

http://www.esecurity.org.in/online-data-security.html