Protecting against piracy:
Building Tamper resistant software

By Amodha Wijekoon

Abstract

Software piracy has been a problem for software developers through out the history of computing. Today one of the main obstacles software vendors face while selling their products is software piracy. Although may precautions are taken while developing to make it difficult for software pirates to re-produce illegal copies of software most methods fails due to lack of complexity and stealthy ness.

Protecting against piracy takes a look at protecting both software that are designed to run in a local machine as well as those which run partially on a local machine as a part of a wider network. Several methods of tamper proofing are considered comparing advantages as well as disadvantages of such methods.

1 . Introduction

Software pricy has long been a problem in software development. Although through out time vendors has realized the importance of protecting against software piracy many companies still fail to take necessary measures to protect their software from the hackers community. Attacks on software mostly done disabling vendors copy protection mechanism. And any method employed by the vendor to authenticate the software if running over a wider network.

2. Methods of tamper resisting local software

2.1 Tamper resisting executable code through physical signatures

Vendors often attempt to protect their software by performing checks to verify that the product is and that the software has not being tempered to provide misleading information about the authenticity . Pre designed solutions such as Sony DADC SecuRom is available off the shelf for developers to use to authenticate installation medium, and therefore provide some form of security to their software distribution. However checking software has been tampered and piracy checks removed is still proving difficult.

SecuRom works by adding a unique electronic key code embedded into the CD-Rom physically placing the signature within the disk. This is done by using special glass- mastering equipment employed by CD mass producers. Secondly the actually executable file is encrypted and included in the CD which requires the embedded key code in order to decrypt its executable.

2.2 Tamper resisting executable code through execution signatures

Most software tempering is done not by altering most of the executable. As it is risky for the hacker to modify large portions of the code due to the fear of breaking the software most hackers attempt to neutralize copy protection code by patching system registers just before copy protection detection code returns its value. Such require slight modifications to the order of execution therefore tracing order of execution in a method such as oblivious hashing proves effective.

Oblivious hashing is designed to provide both results and stealth. Care is taken to blend executable code with hashing code, which makes it difficult for hackers to extract out the code that executes the program. Wile program is compiling at the stage of generating the tree of execution, hashing code is injected and therefore shares the same registers and executes the same way as the application making it virtually impossible to identify. Oblivious hashing computes a dynamic hash value depending on the order of execution, therefore is sensitive to the way software executes.

Firstly code is analyzed for predictable execution sequences and those sequences are selected for hashing. Hash values are computed by looking at the assignment operations and order code blocks are executed. At each point a signature is generated and is compared to check if code execute in the same order it is predicted to be. If a deviation is detected it can be assumed that executable is modified and therefore is no longer usable.

3. Methods of tamper resisting remotely distributed software

Remotely distributed similar to single workstation products heavily suffer from tampering. Unlike single machine software these mostly gets attacked through the communication channel between the two clients.

3.1 Authenticating client and server to prevent tampered clients and servers

Message passing in many cases is done between the remote server and the client sitting o the local computer. And common form of attacks is due to modified clients trying to communicate with the server through a network. In most networks message passing between client and server included a built-in authentication schema. Such as a challenge response mechanism which proves that the client server is talking to and the client server is receiving the message from is authentic.

In Oblivious hashing can also be used to verify that client server is talking to is authentic. This is done via client using a random seed and its execution trace hash computing a challenge and server generating the same by taking the same random seed. If software is not tampered both parties will generate the same hash then this value will be used to encrypt the channel between the two, the client and the server. As hash value is not transmitted it prevent any hackers from eavesdropping packets and if trace is different then encryption want match which will result in server rejecting the client.

3.1 Protecting code from theft

Protecting against piracy also occasionally involves parts of the software kept running on a trusted server. If a hacker gets hold of the client, although they can steal the data in it they will not be able to access the server therefore will have a client that they cannot use. Most cases online update is performed where a client side will download a stub to connect for the current session . Most online games such as Blizzard entertainment’s Diablo and WarCraft series use such technology to prevent modified connection code as each run new stub coming from the server will check the CD keys to make sure the legality of the game CD.

In the realm of palmtop and Pocket PC arena client, server separation is used to both provide a solution that will run in a machine with limited resources as well as protecting server code. In such case wireless communication channel being highly unreliable needs to have a method to verify the integrity of the received message. In such situation messages will contain a hash at the end which reception can verify the message against allowing reception to double check message before responding.

4. Discussion

4.1 Effectively protecting local software

As mentioned both SecuRom and oblivious hashing provide moderate security for the software, however alone is not sufficient to prevent hackers from exploiting the software. As physical signatures are used by duplicating a CD it is not possible to duplicate SecuRom signature, which provides moderate protection. However SecuRom executable can be used to gain the encryption key which compromises the security of SecuRom system. When SecuRom executable is running, it access physical signature of the disk, which can be extracted by recording interrupt calls to the CD-Rom device, and recording its responses. By duplicating the CD using a virtual drive and by emulating the interrupt call a copied CD can now successfully run the protected software as if it is running from the original source.

Similarly employing oblivious hashing alone does not guarantee the authenticity of the software. Although it makes it very difficult for the hackers, as As Yuqun Chen , in his article "Oblivious Hashing: A stealthy software Integrity Verification Primitive" suggest it does not prevent hackers from altering the code that calculate hashes. Even if hashing successfully protected, oblivious hashing becomes vulnerable in the stage where it checks if the hash code is the expected and deepening on it drop or continue execution. With the help of a tracing tool program stack can be traced to the last few calls executed and by looking at that it is possible to isolate and remove the check placed to halt program execution after detection of pirated software.

As mentioned in the article in order for oblivious hashing to work most of the software should be hash able, and therefore the way code executes should be predictable. How ever a multi threaded application might not satisfy this requirements and hence be difficult to identify.

As writer suggest due to the fact that hash code needed to be computed each time and due to the fact that code path depends on the input parameters not all code paths may be covered by one iteration of hashing. This could cause false negatives and therefore escape un-detected as path which is modified never got executed. It is also possible that modifications were done outside the hashed code area and therefore is not validated by the hashing.

However compared with SecuRom Oblivious hashing provides a much secured method of protection which is much more difficult to exploit.

4.2 Effectively protecting remote

Remotely distributed systems suffer from two types of attacks, theft of software by authorized people, which is similar to an authorized person making an unauthorized copy of the software, and Tampered clients trying to connect to a server and execute miscellaneous code.

Firstly to authenticate the client before actual transaction is required. This could be successfully achieved through the use of oblivious hashing which will clearly prevent tampered clients which are not identical to the actual server from successfully establishing a communication channel, however just relaying on a message hash or a challenge response could be fatal as through register patches and code injection it is fairly easy to modify the code in binary form to trick the recover.

However due to the nature of oblivious hashing it is virtually impossible to break through the encryption, and therefore provides much superior protection against tamper proofing the software.

Also separation of user interface and core functional code also provides superior protection against theft as then what is stolen will not be usable without the actual server .As server is beyond the hacker this method too will also provide superior protection against code theft. If code is correctly separated then having the client would bond be good enough to construct the server logic. Companies such as tray games make use of such technology to protect their online game clients by providing the gaming logic on server and user interface on the client machine. This makes sure some one without an account cannot access the game as critical parts of it are missing in the stolen copy.

5 conclusions

Software tempering has been a major problem however techniques such as oblivious hashing is providing much security those programmers can use to tamper proof there code. Although more can be improved in the field of software tamper proofing and current technology not providing all its needed what is provided if used correctly can reduce the risk of tampered software.

6.Bibliography

1. Bellare M, Canetti R , Krawczyk H (1996) “Keying Hash Functions for Message Authentication”

2. Xiangyu Zhang , R. G. (2003). "Hiding program slices for software security." ACM International Conference Proceeding Series;, San Francisco, California, IEEE Computer Society Washington, DC, USA.

3. Y. Chen, R. Venkatesan, M. Cary, R. Pang, S. Sinha, and M. Jakubowski, "Oblivious Hashing: A Stealthy Software Integrity Verification Primitive," in F.A.P. Petitcolas (Ed.): Information Hiding, Proc. of 5th International Workshop (IH 2002), Noordwijkerhout, The Netherlands, October 7-9, 2002. LNCS 2578, p. 400 ff.