CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Technical Disclosure Common

Technical Disclosure Commons

Defensive Publications Series

January 2020

PROTECTING AGAINST MALICIOUS LOGINS ON VIRTUAL MACHINES USING BLOCKCHAIN

Sarthak Sharma

Lovepreet Singh

Yegappan Lakshmanan

Follow this and additional works at: https://www.tdcommons.org/dpubs_series

Recommended Citation
Sharma, Sarthak; Singh, Lovepreet; and Lakshmanan, Yegappan, "PROTECTING AGAINST MALICIOUS LOGINS ON VIRTUAL MACHINES USING BLOCKCHAIN", Technical Disclosure Commons, (January 09, 2020)
https://www.tdcommons.org/dpubs_series/2858

This work is licensed under a Creative Commons Attribution 4.0 License.
This Article is brought to you for free and open access by Technical Disclosure Commons. It has been accepted for inclusion in Defensive Publications Series by an authorized administrator of Technical Disclosure Commons.

PROTECTING AGAINST MALICIOUS LOGINS ON VIRTUAL MACHINES USING BLOCKCHAIN

AUTHORS:

Sarthak Sharma
Lovepreet Singh
Yegappan Lakshmanan

ABSTRACT

Presented herein are techniques that involve utilizing a blockchain-based methodology that provides a defense against rollback attacks on a virtual machine and is scalable for other application areas. In one example, a solution is provided to avoid rollback attacks during restoration of a virtual machine under the assumption of a malicious host and hypervisor. Techniques presented herein also address problems associated with an untrusted host under certain assumptions.

DETAILED DESCRIPTION

Malicious login attempts on a virtual machine (VM) are typically rate limited by blocking the number of wrong attempts that can be entered by a user. However, this is not an effective method to stop the number of login attempts as the state of a virtual machine can always be restored and played back in order to restore the number of login attempts. By restoring a virtual machine in this manner, the malicious login attempts on the virtual machine can effectively become infinite.

The state of a virtual machine is easily accessible from the hypervisor and host. In some instances, the state of a virtual machine can also be gained through other side channels using various hacks or other malicious means. Thus, through the use of such hacks or malicious means, it may be possible to gain unlimited attempts at hacking a virtual machine, which opens a security vulnerability.

Consider an example in which a malicious entity performs ten password attempts to try and login to a virtual machine. In this example, the malicious entity would be locked out of future attempts unless the state of the virtual machine is replayed or forked, in which case the malicious entity could potentially have infinite attempts.

Techniques presented herein utilize a blockchain-based methodology that provides for the ability to defend against malicious agents gaining infinite login attempts to a virtual machine. Additionally, techniques presented herein address problems that may be associated with an untrusted host under certain assumptions.

A solution is provided to prevent a malicious agent from reusing the same state of a virtual machine in order to obtain infinite login attempts. In at least one implementation, a component providing secure computation (e.g., Intel® SGX, ARM® TrustZone®, etc.) can be utilized to perform techniques associated with the solution. A secure shared storage for blockchain with frequent updates from non-adversarial agents may also be utilized in order to avoid a host taking control of the blockchain ledger. For example, the secure computation component on the host can be utilized in order to validate the blockchain and also to perform operations on an input pre-committed on the ledger by a host. Using pre-commitment, the solution can limit the host from providing different inputs on the same virtual machine state, thereby securing the virtual machine against such attacks launched by the host.

Consider a mathematical description and definition of the set-up of the blockchain solution in which O denotes an object of interest. In one example, consider that this could be a disk image for a virtual machine that needs to be protected against a rollback attack. In this example, let TF denote a trapdoor function (note, this mathematical description may not follow the exact definition of a set, but a general understanding of trapdoor functions may be utilized for this description).

Further, let O' <- TF(O) and K be the key to invert the trapdoor function. Therefore, O <- TFInv(K, O') is a probabilistic polynomial-time (PPT) algorithm, but otherwise there does not exist a PPT algorithm to invert TF without the knowledge of K. Additionally, Let P be a PPT program that generates the valid key K based upon a given input. In one example, this input may be a user password or the like. Mathematically, a description can be provided such as: K <- P(I), where I is the input to the program (e.g., the password in this example).

Additionally, let EP <- E(K', P), where EP denotes an encrypted program which is encrypted using an authenticated encryption with associated data (AEAD) algorithm E

using key K'. The AEAD can be used in order to hide the details of converting I to K from the host (using encryption) and to avoid changes (such as bit flips) to P by changing EP.

Further, let VM <- {EP, O', ...} denote a VM image that can be restored on a host and let L denote a ledger maintained by a third party using a protocol similar to blockchain. The ledger is to be present at a shared location maintained by a third party in order to avoid any forgery by the possibly malicious host. In addition, let C be the public certificate of the third party maintaining the shared ledger. The public certificate (C) can be used to verify whether L is a fake ledger or not since the host may fake network communications and present a fake ledger. Let SL denote the signature by the third party over a hash of the ledger.

Finally, let OID be an identifier (ID) provided by a user to uniquely identify each distinct copy of O. This can be a publicly known identifier used to calculate how many times attempts have been made to compute O.

For the present example, consider various steps (in which certain steps may be highlighted in Figure 1, below) as follows:

- A user sets up K', C, OID and any other data for the secure computation unit (e.g., SGX : SetupSGX(K', C, OID, ...)), as generally shown in Step 1 for Figure 1.
- When the host would like to restore the VM image, the host is to execute P in order to restore O.
- The host pre-commits to a given input by declaring (H(I), OID) on a ledger L maintained by a third party and H is a one-way function (preferably a cryptographic hash function), as generally shown at Step 2 in Figure 1. Further, the host requests L and SL from the third party, which is sent to the host as shown at Step 3 in Figure 1.
- The host sends (EP, L, H, I, SL) to the secure computation unit, as generally shown at Step 4 in Figure 1.
- The secure computation unit verifies the authenticity of ledger L using C and SL and, further, verifies the integrity of ledger L using hash values present in the ledger.

- The secure computation unit verifies any state that it is to check. In this example, this can be the number of times a login has been attempted for the VM in which the latest commitment corresponding to OID corresponds to H(I).
- If everything looks fine, the secure computation unit computes P from EP using the stored key K': P <- D(K', EP), where D denotes the decryption function corresponding to E.
- The secure computation unit executes P upon input I to compute K in which K <- P(I), as generally shown at Step 5 in Figure 1.
- The secure computation unit returns K to the host (e.g., K <- SGX(EP, L, H, I, SL), as generally shown at Step 6 in Figure 1.
- The host computes O <- TFInv(K, O') and follows up with further processing on the VM. In this example, this can include restoring the snapshot of the VM.

Thus, the above steps and Figure 1, below, illustrate that blockchain can be utilized to protect against rollback attacks launched by the hypervisor on a virtual machine.

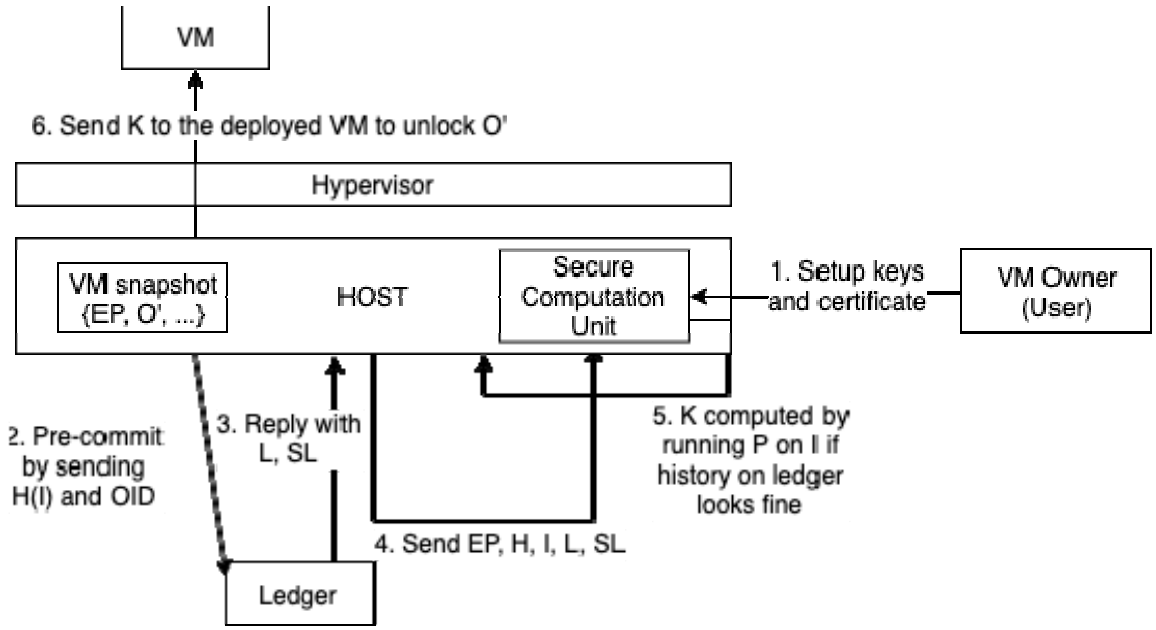

Figure 1

In some implementations, the techniques discussed above may be valid only to unlock an object of interest a first time. Once the object of interest is unlocked, additional security may not be provided for subsequent snapshotting/restoring. Thus, the techniques are assumed to be secure only for cases in which some secret object needs to be unlocked, similar to the case of unlocking a smart phone or the like.

Techniques presented herein may also be applied to other applications such as, for example, to provide limited execution of certain proprietary algorithms. This may be helpful for instances in which there is a need to provide trial versions of an algorithm that is limited by a number of times it can be used. For example, consider that P is a proprietary algorithm that an entity desires to keep secret from a host in which the host is only allowed to execute the proprietary algorithm an N number of times. In this example, there is no need for TF, O, and O' and K will denote the output of P that is of interest to the host. For such an application, the ledger can be used to limit the number of times the host executes the proprietary algorithm.

In summary, techniques presented herein utilize a blockchain-based methodology that provides a defense against rollback attacks on a virtual machine and is scalable for other application areas. In one example, a solution is provided to avoid rollback attacks during restoration of a virtual machine under the assumption of a malicious host and hypervisor. Techniques presented herein also address problems associated with an untrusted host under certain assumptions.