ISSN:2320-0790

An International Journal of Advanced Computer Technology

Data Derivation Investigation

S. S. Kadam, P.B. Kumbharkar
Department of Computer Engineering, SCOE, Sudumbare, Pune
University of Pune

Abstract: Malicious software is a major issue in today's computer world. Such software can silently reside in user's computer and can easily interact with computing resources. It is necessary to improve the honesty of host and its system data. For improvement in security and honesty of host, this work is introduced. This mechanism ensures the correct origin or provenance of critical system information and prevents utilization of host resources by malware. Using this mechanism the source where a piece of data is generated can be identified. A cryptographic origin approach ensures system properties and system- data integrity at kernel level. A frame work is used for restricting outbound malware traffic. This frame work identifies network activities of malware. This frame work can be used as powerful personal firewall for investigating outgoing traffic of a host. Specifically, our derivation verification scheme requires outgoing network packets to flow through a checkpoint on a host, to obtain proper origin proofs for later verification.

Index Terms: Authentication, malware, cryptography, derivation, networking.

I. INTRODUCTION

Compared to the first generation of malicious software in late 1980's, modern attacks are more stealthy and pervasive. Kernel-level root-kits are a form of malicious software that compromises the integrity of the operating system. Such root-kits stealthily modify kernel data structures to achieve a variety of malicious goals, which may include hiding malicious user space objects, installing backdoors and Trojan horses, logging keystrokes, disabling firewalls, and including the system into a botnet [2]. So, host-based signature-scanning approaches alone were proven inadequate against new and emerging malware [6]. We view malicious software or malware in general as entities silently residing on a user's computer and interacting with the user's computing resources. For example, the network calls may be issued by malware to send outbound traffic for denial-of-service attacks, spam. Goal of our work is to improve the reliability of the OS-level data flow; specifically, we provide mechanisms that ensure the correct origin or derivation of critical system data, which prevents antagonist from utilizing host resources [1]. We define a new security mechanism – data-derivation honesty. It verifies the source from which a piece of data is generated. For outbound network packets, we deploy special cryptographic kernel modules at strategic positions of a host's network stack, so that packets need to be generated by user-level applications and cannot be injected in the middle of the network stack. It gives low overhead. The implication of network-packet origin is that one can deploy a sophisticated packet monitor or firewall at the transport layer such as [7] without being bypassed by malware. The application of this system is for distinguishing user inputs from malware inputs, which is useful in many scenarios.

Contribution Work: A new cryptographic derivation verification approach is presented here. And its applications in understanding strong host-based traffic-monitoring are demonstrated.

The key exchange between the two modules is performed using asymmetric keys which is expensive due of their storage and computation cost. This requires RSA algorithm for public key generation and encryption which has high time complexity, so we replace this algorithm with general three-tier security framework for authentication and pair wise key establishment. This three-tier security architecture consists of three separate modules i.e. sign, verify, access module. Two polynomial identifier pools of size M and S are created. Sign and access module are randomly given Km (Km>1) and 1 identifiers from M respectively, similarly verify module and access module are randomly given Ks and Ks-1 identifiers from S respectively. To establish a direct pair wise key between sign module and verify module, a sign module needs to find a stationary access module in its neighborhood, such that, access module can establish pair wise keys with both sign module and verify module. In other words, a stationary access module needs to establish pair wise keys with both the sign module and the verify module. It has to find a common polynomial m (from M) with the sign module and a common polynomial k (from K) with the verify module.

II. LITERATURE REVIEW

Existing root-kit detection work includes identifying suspicious system call execution patterns, discovering vulnerable kernel hooks, exploring kernel invariants, or using a virtual machine to enforce correct system behaviors. For example, Christodorescu, Jha, and Kruegel collected malware behaviors like system calls and compared execution traces of malware against benign programs [9]. They proposed a language to specify malware behavior and an algorithm to mine malicious behaviors from execution traces. A malware analysis technique was proposed and described based on hardware virtualization that hides itself from malware. Although existing OS level detection methods are quite effective, they typically require sophisticated and complex examination of kernel instruction executions. To enforce the integrity of the detection systems, a virtual machine monitor (VMM) is usually required in particular for root-kit detection. TPM is available on most commodity computers.
Information flow control has been an active research area in computer security. As early as in the 70s, Denning et al [3][4]. has proposed the lattice model for securing the information flow and applied it to the automatic certification of information flow through a program. Data tainting, as an effective tracking method, is widely used for the purposes of information leak prevention and malware detection. Taint tracking can be performed at different levels.

Here use of TPM as a signature generator may be viewed as a special type of data tainting. In addition to conventional taint tracking solutions such as hardware memory bit or extended software data structure, here TPM-based solution uniquely supports the cryptographic operations to enforce data confidentiality and the integrity of taint information. The important feature about TPM is its on-chip secret key. Therefore, the client device can be uniquely authenticated by a remote server. Our paper focuses on a host-based approach for ensuring system-level data integrity and demonstrates its application for malware detection. In comparison, network trace analysis typically characterizes malware communication behaviors for detection. Such solutions usually involve pattern-recognition and machine learning techniques, and have demonstrated effectiveness against today's malware. Our work provides a hardware-based integrity service to address that problem. In comparison to NAB which is designed specifically for browser input verification, this work provides a more general system-level solution for keystroke integrity that is application-oblivious.

The element of human behavior has not been extensively studied in the context of malware detection, with a few notable exceptions including solutions by Cui, Katz, and Tan and Gummadi [5][8]. They investigated and enforced the temporal correlation between user inputs and observed traffic. BINDER describes the correlation of inputs and network traffic based on timestamps. It does not provide any security protection against the detection system itself, e.g., how to prevent malware from forging input events.

The work by Srivastava and Giffin on application aware blocking of malware traffic may bear superficial similarity to our solution [10]. They used a virtual machine monitor (VMM) to monitor application information of a guest OS without using any cryptographic scheme. Existing system use root kit-detection work which includes identifying suspicious system call execution patterns, discovering vulnerable kernel hooks, exploring kernel invariants or using a virtual machine to enforce correct system behaviors. The lattice model for securing the information flow through a system as an effective tracking method is widely used for the purposes of information leak prevention and malware detection and can be performed at different levels, for example within an application, within a system, or across distributed hosts. But this system lacks in cryptographic operations to enforce data confidentiality and the integrity.

III. DESIGN

Introduced derivation verification mechanism has a essential difference from the traditional cryptographic signature scheme. In most signature schemes the signer is assumed to be a person who exercises judgment in signing documents and also in protecting his or her signing keys. In the environment of malware detection, the signer and verifier are programs. Prevention against these attacks is critical. For network-traffic monitoring, malware may attempt to send traffic by directly raising functions at the network-layer, but not at the lower level data-link or physical layers. We assume that the Trusted Platform Module (TPM) is corrupt opposing; the cryptographic operations are applied suitably; and the remote server is trusted and secure. TPM provides the guarantee of load-time code integrity. It does not provide any detection ability for run-time compromises such as buffer overflow attacks [11]. Advanced attacks [12], [13] may still be active under this assumption, indicating the importance of our solutions.

We describe three actions for data-derivation investigation on a host: setup, sign and verify.
• Setup: the data producer sets up its signing key k and data consumer sets up its verification key k0 in a secure fashion that prevents malware from accessing the secret keys.
• Sign(D, k): the data producer signs its data D with a secret key k, and outputs D along with its proof sig.
• Verify(sig,D, k0): the data consumer uses key k0 to verify the signature sig of received data D to ensure its origin, and rejects the data if the investigation fails. Although simple, the cryptographic derivation investigation method can be used to ensure and impose correct system and network properties and appropriate workflow under a trusted computing environment.

IV. INVESTAGATING DERIVATION OF OUTBOUND TRAFFIC

The cryptographic derivation verification technique in a network setting, for ensuring the reliability of outbound

packets, as they flow through the host's network stack is illustrated here. Fig. 1 shows the network stack. Genuine traffic origins from the application layer whereas malware traffic may be injected to the lower layers..

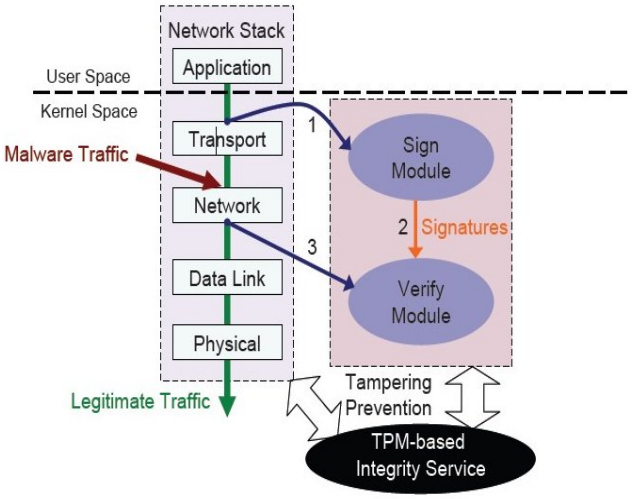

Fig. 1. Network stack. Genuine traffic origins from the application layer whereas malware traffic may be injected to the lower layers. Traffic checkpoints are placed at the Sign and Verify modules.

Traffic checkpoints are placed at the Sign and Verify modules. The design of a lightweight traffic monitoring framework is described. It can be used as a building block for constructing powerful personal firewalls or traffic-based malware detection tools. We demonstrate the effectiveness of our traffic supervising framework in identifying the network activities of quiet malware.

The network stack is part of the host's operating system and consists of five layers, application, transport, network, data link, and physical layers. User outbound traffic travels all five layers on the stack from the top to the bottom before being sent out. System services are typically implemented as applications, thus their network flow also traverses the entire Internet protocol stack. Specifically, our derivation investigation scheme requires outgoing network packets to flow through a checkpoint on a host, to obtain proper origin proofs for later investigation. Any traffic sent through disabling or bypassing the firewall can be detected, as the packets are unable to provide their origin proofs. And we can effectively prevent any traffic to be sent without passing through a certain checkpoint appreciably improving the assurance of traffic-based malware detection on hosts. Such a simple yet powerful traffic-monitoring framework can defer advanced detection on application-level traffic such as [14]. Genuine outbound network traffic passes through the entire network stack in the host's operating system. We develop a strong cryptographic procedure for enforcing the proper origin of a packet on a host.

A. Architecture of Traffic Provenance Verification

Here a general approach is described for improving the assurance of system data and properties of a host, which has applications in preventing and identifying malware activities. The host-based system security solutions against malware complement network-traffic-based analysis. We demonstrated application in identifying quiet malware activities of a host, in particular how to distinguish malicious/unauthorized data flow from valid one on a computer that may be compromised.

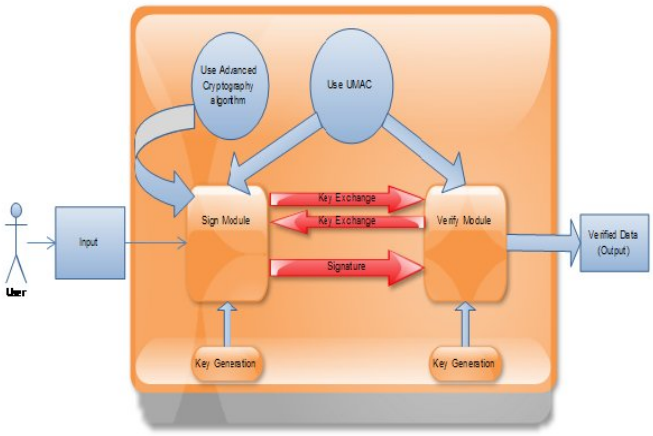

Fig. 2. System Architecture

Our design of the traffic-monitoring framework extends the host's network stack and deploys two kernel modules, Sign and Verify modules, as illustrated in Figure 2. Both signing and verification of packets take place on the same host but at different layers of the network stack – the Sign module is at the transport layer, and the Verify module is at the network layer. The two modules sharing a secret cryptographic key monitor the integrity of outbound network packets. All legitimate outgoing network packets first pass through the Sign module, and then the Verify module. The Sign module signs every outbound packet, and sends the signature to the Verify module on the same host, which later verifies the signature with a shared key. The signature proves the provenance of an outgoing packet. If a packet's signature cannot be verified or is missing, then the packet is labeled as suspicious.

V. IMPLEMENTATION

The system follows three-tier architecture. Our main checkpoints are created by polynomial identifier pools.

The three-tier security architecture consists of three separate modules i.e. sign, verify, access module. Two polynomial identifier pools of size M and S are created. Sign and access module are randomly given Km (Km>1) and 1 identifiers from M respectively, similarly verify module and access module are randomly given Ks and Ks-1 identifiers from S respectively. To establish a direct pair wise key between sign module and verify module, a sign module needs to find a stationary access module in its neighborhood, such that, access module can establish pair wise keys with both sign module and verify module. In other words, a stationary access module needs to establish pair wise keys with both the sign

module and the verify module. It has to find a common polynomial m (from M) with the sign module and a common polynomial k (from K) with the verify module.

Summary:

Input: In this module we will design GUI and Database of the system and we complete input module.

Sign: In this module we create sign module and transfer data input module to sign module and we add private key and public key of sign module and encrypt data.

Verify: In this module we create verify module, using this module and sign module we exchange the key and decrypt the data. In this module we use UMAC.

Output: In this module we develop output module it is use to get data and in that we will some option for save and reject data.

Fig. 3 shows process flow.

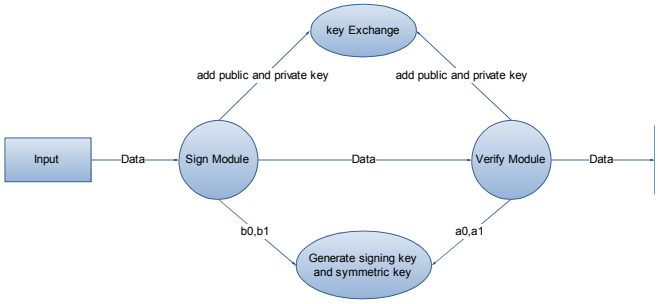

Fig. 3 Process flow.

RSA algorithm is used here for key generation, encryption and decryption.

RSA Key Generation Algorithm:

Begin
Select two large prime numbers p, q
Compute
$n = p \times q$
$v = (p-1) \times (q-1)$
Select small odd integer k relatively prime to v
gcd(k, v) = 1
Compute d such that
$(d \times k)\%v = (k \times d)\%v = 1$
Public key is (k, n)
Private key is (d, n)

End

RSA Encryption Algorithm:

Begin
Let Input: integers k, n, M
M is integer representation of plaintext message
Let C be integer representation of cipher text
Compute
$C = (M^k)\%n$
end
Output: integer C
– cipher text or encrypted message

RSA Decryption Algorithm:

Begin
Let Input: integers d, n, C
C is integer representation of cipher text message
Let D be integer representation of decrypted cipher text.
Compute
$D = (C^d)\%n$
end
Output: integer D
- decrypted message

VI. STATICTICAL MODEL

Set Theory Analysis:

Identify the Input User Data:
IN= {in1, in2, in3….}
Where 'IN' is main set of Input User Data like in1, in2, in3…inn

Identify the public key:
PK= {pk1, pk2, pk3….}
Where 'PK' is main set of public key User like pk1, pk2, pk3…pkn

Identify the Private key:
PriK= {prik1, prik2, prik3….}
Where 'PriK' is main set of private key User like prik1, prik2, prik3…prikn

Identify the Key Exchange:
KE= {ke1, ke2, ke3….}
Where 'KE' is main set of key Exchange like ke1, ke2, ke3…ken

Identify the Key Generation:
KG= {kg1, kg2, kg3….}
Where 'KG' is main set of Key Generation like kg1, kg2, kg3…kgn

Identify the Symmetric key:
SK= {sk1, sk2, sk3….}
Where 'SK' is main set of symmetric key like sk1, sk2, sk3…skn

Identify the Signing Key:
SIK= {sik1, sik2, sik3….}
Where 'SIK' is main set of singing key like sik1, sik2, sik3…sikn

Process:

We define a security property data origin integrity. It states that the source from which a piece of data is generated can be verified. We give the concrete illustration of how data-provenance integrity can be realized for system-level data.

Identify the processes as P.

P= {Set of processes}
P = {P1, P2, P3, P4……Pn}
P1 = {e1, e2, e3, e4, e5}
Where
{e1= input Data}
{e2= key generation}
{e2= key exchange}
{e3= create signature}
{e4= verify signature}
{e5=display Message}

Create Signature:

We use UMAC algorithm to generate signature for each packet.

UMAC signature = H_K(S) XOR F(nonce)

Where H = hash algorithm
K = Signing Key
S = Source
F = Pseudorandom number generator

Signature Encryption:

We use Advance cryptography algorithm for signature encryption.

CB1 = P (XOR) KB1
CB2= CB1 >> 3
CB3 = CB2 (XOR) KB2
CB4 = CB3 (XOR) KB3
CB4 is the encrypted data.
P – Plain text
CB – Cipher block
KB – Key block

The Three Tier Security Architecture:

Polynomial Identifier pool M= {m1, m2, m3…} of size m.
Polynomial Identifier pool S= {s1, s2, s3…} of size s.
Sign and access module are randomly given Km (Km>1) and 1 identifiers from M respectively. Verify and access modules are randomly given Ks and Ks-1 identifiers from S respectively.

VII. CONCLUSION

Here a general approach for improving the assurance of system data and properties of a host is described, which has applications in preventing and identifying malware activities. Defined host-based system security solutions against malware complement network-traffic based analysis. Here application of derivation investigation mechanism is demonstrated in identifying quiet malware activities of a host, in order to distinguish malicious/unauthorized data flow from genuine one on a computer.

Technical contributions are, the model and operations of cryptographic derivation identification in a host based security setting is proposed. It's important usage for achieving highly assured kernel data and application data of a host, and associated technical challenges are pointed out. And, origin investigation approach is demonstrated by a framework for ensuring the honesty of outbound packets of a host. This traffic-monitoring framework creates checkpoints that cannot be bypassed by malware traffic.

Reference

[1]. Kui Xu, Huijun Xiong, Chehai Wu, Deian Stefan, Danfeng Yao Data-Provenance Verification For Secure Hosts *In IEEE Transactions on Dependable and Secure Computing Vol.9 No.2 Year 2012*

[2]. A. Baliga, V. Ganapathy, and L. Iftode. Automatic inference and enforcement of kernel data structure invariants. *In 24th Annual Computer Security Applications Conference* (ACSAC), 2008.

[3]. D. E. Denning. A lattice model of secure information flow. *Commun. ACM, 19:236–243, May 1976.*

[4]. D. E. Denning and P. J. Denning. Certification of programs for secure information flow. *Commun. ACM, 20:504–513, July, 1977.*

[5]. W. Cui, R. H. Katz, andW. tian Tan. Design and Implementation of an extrusion-based break-in detector for personal computers. *In ACSAC, pages 361–370. IEEE Computer Society, 2005*.

[6]. M. G. Jaatun, J. Jensen, H. Vegge, F. M. Halvorsen, and R. W. Nergard. Fools download where angels fear to tread. *IEEE Security & Privacy, 7(2):83–86, 2009.*

[7]. H. Xiong, P. Malhotra, D. Stefan, C. Wu, and D. Yao. Userassisted host-based detection of outbound malware traffic. *In Proceedings of International Conference on Information and Communications Security (ICICS), December 2009.*

[8]. R. Gummadi, H. Balakrishnan, P. Maniatis, and S Ratnasamy. Not-a-Bot: Improving service availability in the face of botnet attacks. *In Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NDSI), 2009.*

[9]. M. Christodorescu, S. Jha, and C. Kruegel. Mining specifications of malicious behavior. *In ESEC-FSE '07: Proceedings of the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering, pages 5–14, New York, NY, USA, 2007. ACM.*

[10]. A. Srivastava and J. Giffin. Tamper-resistant, Application-aware blocking of malicious network connections. In RAID '08: *Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection, pages 39–58, Berlin, Heidelberg, 2008. Springer-Verlag*

[11]. S. Garriss, R. C´aceres, S. Berger, R. Sailer, L. van Doorn, and X. Zhang. Trustworthy and personalized Computing on public kiosks. In MobiSys '08: *Proceeding of the 6th international conference on Mobile systems, applications, and services, pages 199–210, New York, NY, USA, 2008. ACM.*

[12]. A. Baliga, P. Kamat, and L. Iftode. Lurking in the shadows: Identifying systemic threats to kernel data.

In IEEE Symposium on Security and Privacy, pages 246–251. IEEE Computer Society, 2007.

[13]. J. Wei, B. D. Payne, J. Giffin, and C. Pu. Soft-timer driven transient kernel control flow attacks and defense. *In ACSAC '08: Proceedings of the 2008 Annual Computer Security Applications Conference, pages 97–107, Washington, DC, USA, 2008. IEEE Computer Society.*

[14]. Z. Wang, X. Jiang, W. Cui, and X. Wang. Countering persistent kernel rootkits through systematic hook discovery. *In RAID '08: Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection, pages 21–38, Berlin, Heidelberg, 2008. Springer-Verlag.*