Verlässliche forensische Live-Analyse

Christoph Settgast
Universität Erlangen-Nürnberg

***Zusammenfassung*—HyperSleuth ist ein Framework zur verlässlichen, forensischen Live-Analyse. Es nutzt hardwarebasierte Virtualisierungsunterstützung aktueller x86-Prozessoren, um ein laufendes Betriebssystem ohne merkbare Unterbrechung zu virtualisieren. Die forensische Analyse ist danach gegenüber Manipulationen geschützt, da der Hypervisor die endgültige Kontrolle über die Hardware besitzt. Sie ist gegenüber dem analysierten Betriebssystem transparent und kann daher während des Produktiveinsatzes mit nur geringen Leistungseinbußen durchgeführt werden. Zur Absicherung des Virtualisierungsvorgangs wird ein Prüfsummenverfahren über den Hypervisor mit Laufzeitmessung durch eine forensische Workstation durchgeführt.**

HyperSleuth ist modular aufgebaut und implementiert drei Analysemodule: ein Modul zur Erstellung eines Speicherabbilds, das nach dem copy-on-write-Prinzip arbeitet, ein Lügendetektor zur Erkennung von Schadsoftware und ein Modul zur Erstellung eines Systemaufrufprotokolls, mit dem auf die internen Abläufe unbekannter Software geschlossen werden kann.

I. Einleitung

Forensische Live-Analyse bietet gegenüber der Festplattenanalyse sowohl Informations- als auch Geschwindigkeitsvorteile. Es lassen sich so nicht nur Informationen über enthaltene Dokumente, sondern auch angemeldete Benutzer, offene Programme und Netzwerkverbindungen gewinnen. Diese Informationen stehen auch schneller als mit der klassischen Festplattenanalyse zur Verfügung. Problematisch hierbei war bisher die Verlässlichkeit der gewonnenen Informationen. Diese lässt sich, wenn das System und die Software des zu untersuchenden Systems verwendet wird, nur schwer garantieren. Vom System gelieferte Informationen können wissentlich oder unwissentlich durch Schadsoftware verfälscht werden.

Daher muss eine abgesicherte Ausführungsumgebung zur Analyse bereitgestellt werden, die die Kontrolle über das Betriebssystem des zu untersuchenden Systems erhält. Hardwareseitige Virtualisierungsunterstützung in modernen Prozessoren [1], [2] bietet die Möglichkeit der bedarfsorientierten Virtualisierung eines Hostbetriebssystems und der darin laufenden Programme. Die konzeptuale Schadsoftware *Blue Pill* nutzte diese Möglichkeit zuerst [3]. Auch Programme zur Erkennung von Schadsoftware griffen die Möglichkeit danach schnell auf [4], [5], um kernel-basierte Malware zu erkennen. Zur forensischen Untersuchung gehört jedoch eine verlässliche Absicherung des Virtualisierungsvorgangs, die nicht Teil bisheriger Arbeiten war.

Der hier gezeigte Ansatz *HyperSleuth* [6] der verlässlichen forensischen Live-Analyse nutzt Hardwareunterstützung zur Virtualisierung. Er setzt darüber hinaus eine softwarebasierte Methode [7] zur Absicherung des Virtualisierungsvorgangs ein und ermöglicht so ein modulares Framework für unterschiedliche Analysemodule. Beispielhaft für solche Analysemodule werden ein Lügendetektor, ein Modul zur Erstellung eines Speicherabbilds und ein Protokollierungsmodul für Systemaufrufe näher betrachtet.

In Abschnitt II werden Grundlagen und verwandte Arbeiten, die zu dem gezeigten Ansatz geführt haben, erläutert. In Abschnitt III wird der Ablauf der Virtualisierung und deren Absicherung genauer erläutert. In Abschnitt IV wird dann die Implementierung des Hypervisors beleuchtet und abschließend in Abschnitt V die zur Verfügung stehenden forensischen Analysemodule vorgestellt.

Dieser Beitrag entstand im Rahmen des Konferenzseminars „Sicherheit und Zuverlässigkeit in Systemsoftware“, das im Wintersemester 2011/12 an der Universität Erlangen-Nürnberg durchgeführt wurde. Es wurde organisiert vom Lehrstuhl für IT-Sicherheitsinfrastrukturen (Prof. Dr. F. Freiling).

II. Hintergrund

A. Definitionen

Hostbetriebssystem Ein direkt auf der Hardware arbeitendes Betriebssystem.

Gastbetriebssystem Ein unter einem Hypervisor arbeitendes virtualisiertes Betriebssystem.

Hypervisor Bei der Virtualisierung eingesetzte Software, auch Virtual Machine Monitor (VMM) genannt, die die Ausführung der Gastbetriebssysteme kontrolliert.

B. Entwicklung

Persistente Virtualisierungsansätze wurden bereits seit 2003 zur Erkennung von Schadsoftware eingesetzt. Hierbei ist das übliche Vorgehen folgendermaßen: Beim Start des Rechners wird zuerst ein minimaler Hypervisor gestartet, der dann das eigentliche Betriebssystem als Gast startet. Hierbei hat der Hypervisor die komplette Kontrolle über das Betriebssystem, und eventuell laufende Schadsoftware kann durch den Hypervisor erkannt und eventuell auch entfernt werden [8]–[11]. Ein Nachteil dieses Vorgehens ist jedoch, dass die Virtualisierung bereits vor dem Start des Betriebssystem vorhanden sein muss, was zu Leistungseinbußen führen kann.

Hardwareseitige Virtualisierungsunterstützung ist in modernen x86-Prozessoren seit 2005 enthalten [1]. Sie kann auch dazu eingesetzt werden, ein Betriebssystem im laufenden Betrieb sowohl zu virtualisieren als auch die Virtualisierung wieder rückgängig zu machen. Sobald das laufende Betriebsystem virtualisiert wurde, ist es nur noch Gastbetriebssystem, und der Hypervisor besitzt die endgültige Kontrolle.

Das Potential der hardwareseitigen Unterstützung zur Virtualisierung von bereits laufenden Betriebssystemen wurde zum ersten Mal 2006 von Rutkowska im Blue Pill Malware-Konzept [3] demonstriert, indem ein laufendes Windows Vista

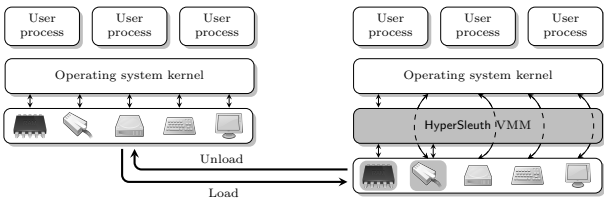

Abbildung 1. Der Ansatz von HyperSleuth [6].

System von einem Rootkit virtualisiert und anschließend kontrolliert wurde.

Für die forensische Analyse wird nun ein ähnlicher Ansatz gewählt. Das laufende Betriebssystem wird kurz unterbrochen und ein Hypervisor installiert. Danach wird die Ausführung des Systems als Gast fortgesetzt und die forensische Analyse durch den Hypervisor durchgeführt (Abbildung 1). Dieser sendet die Ergebnisse nach Abschluss der Analyse über das Netzwerk dann an eine forensische Workstation.

Um diesen Ansatz der bedarfsorientierten Virtualisierung für forensische Zwecke nutzen zu können, muss er um eine verlässliche Absicherung ergänzt werden. Es muss sichergestellt werden, dass der Virtualisierungsvorgang ohne Unterbrechung und ohne Veränderung durch laufende Schadsoftware durchgeführt wird. Dazu eignen sich hardwarebasierte Ansätze wie Trusted Computing [12], jedoch auch rein softwarebasierte Methoden können hierfür herangezogen werden. In HyperSleuth wird dazu *Conqueror* [7] verwendet, der mit Hilfe eines zweiten Systems die Ausführungszeit von obfuskierter Prüfsummenberechnung auf dem Zielsystem überwacht und so eventuelle Modifikationen während der Ausführung zu erkennen versucht.

Mit Hilfe der Virtualisierung nach Bedarf und softwareseitiger Verlässlichkeitsprüfung lässt sich so eine verlässliche forensische Live-Analyse eines Systems durchführen.

III. Virtualisierungsvorgang

Im folgenden Abschnitt wird der Vorgang beschrieben, ein Betriebssystem im laufenden Betrieb in ein Gastbetriebssystem unter einem forensischen Hypervisor zu verwandeln.

Die Hardwareunterstützung in x86-Prozessoren (Intel VT-x bzw. AMD SVM) unterscheidet grundsätzlich zwischen zwei Kontexten: *VMX root mode* und *VMX non-root mode*. Sobald ein Hypervisor im Prozessor registriert ist, kann ein Kontextwechsel zwischen beiden Modi ausgeführt werden. Dieser kann explizit über den Befehl *VMM call* oder durch bestimmte Ereignisse ausgelöst werden. Unter die Ereignisse fallen beispielsweise Interrupts, Ein-/Ausgaberoutinen oder die Ausführung von sog. privilegierten Befehlen, die bestimmte CPU-Register oder Tabellen der MMU verändern. So funktioniert das Gastsystem unmodifiziert weiter, es adressiert Speicher, CPU und Geräte direkt. Befehle des Gastsystems können nach einem Kontextwechsel vom Hypervisor verändert, unterbrochen oder verzögert werden.

Zur Virtualisierung sind vier Schritte notwendig. Zuerst wird Speicher alloziert, in den der Hypervisor geladen wird. Dann wird ein Kontextwechsel durchgeführt und der Hypervisor initialisiert. Abschließend wird wieder ein Kontextwechsel durchgeführt, und das Betriebssystem arbeitet normal weiter.

A. Virtualisierungsvorgang

Der Hypervisor benötigt einen Block Speicher, in dem die Verwaltungsstrukturen und die Programmlogik des Hypervisors abgelegt werden. Dieser Block wird *Virtual Machine Control Structure* (VMCS) genannt und enthält folgende Bestandteile. Er enthält einerseits eine *Host State Area*, die die Interrupt Descriptor Table (IDT), die Seitentabelle (Page Table), die Global Descriptor Table (GDT) und die Local Descriptor Table (LDT) sowie das Code- und Datensegment des Hypervisors referenziert. Darüber hinaus enthält die VMCS eine *Guest State Area*, die ebenso IDT, Seitentabelle, GDT und LDT des Gasts referenziert, und sog. *Control fields*, die eine Liste der Ereignisse enthalten, die einen Kontextwechsel auslösen.

Sobald der Speicher im VMX non-root mode alloziert wurde, wird ein expliziter Kontextwechsel ausgeführt. Um die weitere Ausführung des Betriebssystems als Gast zu ermöglichen, werden die Referenzen in der Guest State Area auf die bisherigen Tabellen gesetzt. Die Tabellen der Host State Area hingegen werden neu angelegt und referenziert. Ebenso wird der Code des Hypervisors in der Host State Area referenziert, sodass die forensische Analyselogik des Hypervisors nach Kontextwechseln ausgeführt werden kann. Der Zustand der VMCS nach der Initialisierung ist in Abbildung 2 gezeigt.

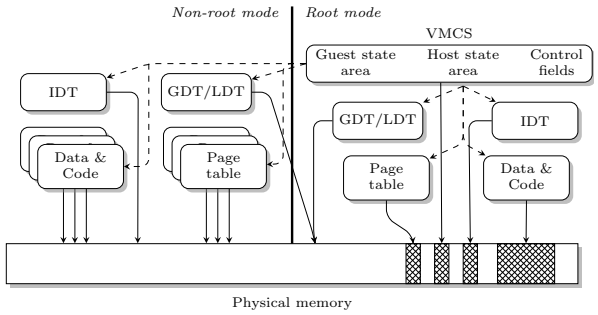

Abbildung 2. Überblick über den Speicher nach dem Virtualisierungsvorgang. Die dunkel schraffierten Stellen im Speicher müssen durch eine geeignete Seitentabelle für den Gast unzugänglich sein [6].

Nach der Initialisierung der VMCS wird wieder ein Kontextwechsel in den VMX non-root mode durchgeführt und das Betriebssystem kann als Gast weiterlaufen.

B. Absicherung des Virtualisierungsvorgangs

Die Initialisierung der VMCS muss atomar ablaufen, wenn die Virtualisierung für forensische Zwecke genutzt werden soll. Andernfalls könnte ein Angreifer die Initialisierung unterbrechen und die Kontrolle des Gastbetriebssystems durch den Hypervisor wäre nicht mehr gewährleistet. Aus Gründen der Portabilität verwendet HyperSleuth einen softwarebasierten Ansatz zur Garantie der Atomarität.

Dieser *Conqueror* [7] genannte Ansatz basiert darauf, die Laufzeit von Prüfsummenberechnung über das Programm

selbst von einem vertrauenswürdigen System aus zu überwachen und so eine evtl. Unterbrechung der Initialisierung zu erkennen.

Dazu wird auf dem vertrauenswürdigen System eine Menge von Blöcken zur Berechnung von Prüfsummen erstellt und obfuskiert. Eine Untermenge wird zur Ausführung vorgesehen und von diesen wird die Reihenfolge der Ausführung zufällig festgelegt. Die alle obfuskierten Blöcke und die Ausführungsreihenfolge werden an HyperSleuth gesendet. Blöcke, die nicht zur Ausführung vorgesehen sind, werden als Interrupt-Handler eingetragen. Wenn diese Initialisierung abgeschlossen ist, sendet das zweite System den Startbefehl an HyperSleuth und beginnt mit der Zeitmessung. Nur wenn die richtige Prüfsumme innerhalb eines bestimmten Zeitfensters beim zweiten System eintrifft, kann garantiert werden, dass die Initalisierung mit höchsten Ausführungsrechten, ohne Interrupts und ohne einen anderen Hypervisor ausgeführt wurde. Ein Überblick über den Ablauf wird in Abbildung 3 gezeigt.

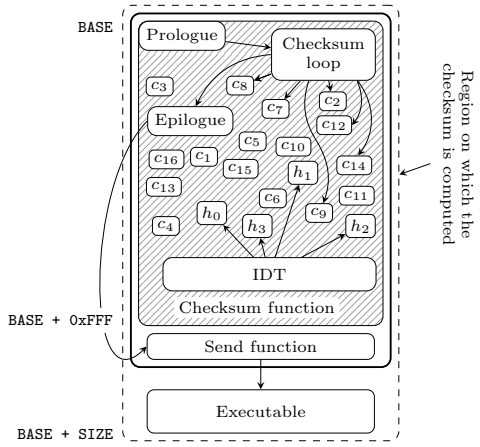

Abbildung 3. Überblick über die Absicherungsroutine Conqueror [7].

Das in HyperSleuth verwendete Absicherungsverfahren ist stark von der Hardwarekonfiguration des zu untersuchenden Systems abhängig. Bei Messungen der Laufzeiten der Prüfsummenberechnung von Conqueror zeigte sich, dass die Laufzeiten von manipulierter Berechnung nur um rund 2,8% von den Laufzeiten ohne Manipulationen abweichen. Daher empfehlen die Autoren von Conqueror, dass ein von der Hardware identisches System zur Verfügung steht, um die zu erwarteten Laufzeiten zu kalibrieren [7]. Dies stellt die praktische Einsatzfähigkeit der softwarebasierten Absicherung von HyperSleuth in Frage. Hardwarebasierte Ansätze des Trusted Computing wie *TPM late launch* [12] scheinen hier vielversprechender, um die Virtualisierung von laufenden Betriebssystemen für forensische Zwecke nachweisbar unverfälscht durchzuführen.

IV. Architektur

Im folgenden Abschnitt werden die notwendigen technischen Implementierungsdetails vorgestellt, um forensische Analysen durchzuführen, mit der Prämisse, dass das System erfolgreich virtualisiert wurde. Dabei muss bei der MMU-Virtualisierung sichergestellt werden, dass die Speicherbereiche des Hypervisors für den Gast unzugänglich sind. Außerdem muss die Gerätekommunikation mit nichtmodifizierten Treibern vom Gastsystem weiter durchgeführt werden können. Die Resultate der Analyse müssen über das Netzwerk ohne Verwendung der Treiber des Gastsystems an eine forensische Workstation übermittelt werden können.

A. MMU-Virtualisierung

Der wichtigste Aspekt der MMU-Virtualisierung ist die Sicherstellung, dass das Gastbetriebssystem nicht auf Speicherbereiche des Hypervisors, konkret die VMCS, zugreifen kann. Dazu muss das Gastbetriebssystem auf Speicherseiten (Pages) arbeiten, die mit Hilfe der Seitentabelle in physikalische Adressen übersetzt werden.

Der Hypervisor von HyperSleuth erstellt nun sogenannte Schattenseitentabelle für das Gastbetriebssystem, über die der Zugriff auf Speicherbereiche des Hypervisors verhindert wird. Des Weiteren wird bei allen Zugriffen des Gasts auf die Seitentabelle ein Kontextwechsel durchgeführt und so eine strikte Speicherkontrolle sichergestellt. Dieser Kontextwechsel erlaubt es darüber hinaus, die Erstellung eines Speicherabbilds leicht durchzuführen. HyperSleuth wählt mit dieser Implementierung einen einfach zu programmierenden Weg, der gewisse Leistungseinbußen mit sich bringt.

B. Ein- und Ausgabe-Virtualisierung

Bei üblicher Virtualisierung stellt der Hypervisor dem Gastbetriebssystem virtuelle Geräte zur Verfügung, die dann im Hypervisor auf echte Geräte übersetzt werden. Dateibasierte Festplattencontroller oder virtuelle Netzwerkkarten wären dafür ein Beispiel. Im Fall von HyperSleuth soll das Betriebssystem nicht feststellen, ob es virtualisiert wurde. Daher muss hier direkter Zugriff auf die bisherigen Geräte ermöglicht werden. Dies ist bei HyperSleuth leicht möglich, da HyperSleuth selbst außer der Netzwerkkarte keine Geräte verwendet und damit dem Gastsystem genau den unmodifizierten Zugriff auf Eingabe- und Ausgabegeräte anbieten kann, den das Gastsystem zur Ausführung benötigt. Auf die Frage, wie die Netzwerkkarte zwischen Gast und Hypervisor geteilt wird, wird im Folgenden eingegangen.

C. Netzwerkvirtualisierung

HyperSleuth kommuniziert aus zwei Gründen mit einer forensischen Workstation über das Netzwerk: einerseits, um den den Virtualisierungsvorgang abzusichern, und andererseits, um die Ergebnisse der Analyse auf der Workstation zugänglich zu machen. HyperSleuth verwendet hier keine Funktionen, auch keine Netzwerktreiber, des Gastsystems, damit die Integrität der gewonnenen Analysen nicht durch möglicherweise kompromittierte Funktionen im Betriebssystem gefährdet werden. Daher muss HyperSleuth selbst Treiber für die Netzwerkkarte und die Netzwerkkommunikation mitbringen.

Um auch dem Gastsystem weiterhin Zugriff auf die Netzwerkkarte zu ermöglichen, werden die PCI-Register der Netzwerkkarte bei jedem Kontextwechsel zum Hypervisor gesichert und später wiederhergestellt. Während der Hypervisor

Netzwerkaktivitäten ausführt, werden Interrupts abgeschaltet und aktiv gewartet, bis die Netzwerkkommunikation abgeschlossen ist. Auch diese Implementierung von HyperSleuth ist bewusst einfach gewählt, auch wenn sie Leistungseinbußen mit sich bringt.

Zusammenfassend lässt sich so feststellen, dass die eigentliche Virtualisierung der Hardware in HyperSleuth nicht komplex gestaltet ist, sondern sich an direkten, einfachen Implementierungen zur MMU-Virtualisierung, für den Zugriff auf Geräte und das Netzwerk orientiert.

V. Analysemodule

HyperSleuth selbst ist lediglich ein Framework zur Live-Analyse, es bietet alles zur Virtualisierung und zur Absicherung der Virtualisierung, aber lagert die konkreten forensischen Untersuchungsmethoden in Module aus. Drei solche Module werden in diesem Abschnitt vorgestellt: ein Modul zur Erstellung eines konsistenten Speicherabbildes, ein Lügendetektor und ein Modul zur Erstellung eines Systemaufrufprotokolls.

A. Speicherabbild

Beim Erzeugen eines Speicherabbilds ist es von besonderer Bedeutung, dass das Abbild konsistent ist. Es muss genau einen definierten Zustand des Speichers repräsentieren. Dies hat typischerweise zur Folge, dass das System zur Erstellung des Speicherabbilds angehalten wird. Gerade bei Serversystemen, die sich im Produktiveinsatz befinden, heißt dies Ausfallzeit für die Kunden des Systems. Wenn nur ein geringer Verdacht besteht, dass ein System kompromittiert wurde oder dass von einem System eine kriminelle Handlung ausgeführt wurde, so rechtfertigt dies in der Güterabwägung eventuell nicht die Ausfallzeit des Systems.

Daher erstellt HyperSleuth das Speicherabbild im Hintergrund, während das System weiter funktioniert. So kann auch bei geringem Verdacht eine forensische Analyse mit nur geringen Auswirkungen auf den Betrieb durchgeführt werden. Dies wird mit einem von *Copy-on-write* inspirierten Ansatz realisiert.

Dabei werden in der Schattenseitentabelle des Gastsystems die Zugriffsrechte so gesetzt, dass nur lesende Zugriffe möglich sind. Bei schreibenden Zugriffen des Gastsystems kommt es zum Seitenfehler, der einen Kontextwechsel zum Hypervisor auslöst. Im Hypervisor wird darauf die Seite des Speichers, die modifiziert werden soll, gesichert, und danach der schreibende Zugriff nach einem Kontextwechsel wieder zugelassen. In einer Tabelle werden alle bereits gesicherten Speicherseiten im Hypervisor protokolliert. Um die Erstellung des Abbilds zu beschleunigen, löst auch die `hlt`-Instruktion, die dem Idle-Zustand der CPU entspricht, einen Kontextwechsel aus. So können auch Idle-Phasen des Betriebssystems dazu genutzt werden, das Speicherabbild zu erstellen.

Sobald eine Speicherseite gesichert wurde, wird diese an die forensische Workstation gesendet, auf der das Speicherabbild nach und nach zusammengebaut wird. HyperSleuth erstellt dabei eine Datei, die mit den gängigen Werkzeugen zur Hauptspeicheranalyse, wie Volatility [13], kompatibel ist.

B. Lügendetektor

Eine weitere, nicht notwendigerweise forensische, Anwendungsmöglichkeit von HyperSleuth ist ein sog. Lügendetektor. Hierbei wird versucht, Schadsoftware, die sich im System versteckt und in Prozesslisten nicht erscheint, dennoch zum Vorschein zu bringen. Dazu werden Angaben des Betriebssystems, beispielsweise über laufende Prozesse, mit denen aus dem Hauptspeicher über den Hypervisor rekonstruierbaren Informationen verglichen. Eventuelle Widersprüche zwischen den Informationen, die der Hypervisor sieht, und den Informationen, die das Betriebssystem selbst bereitstellt, deuten dann auf Manipulationen durch Schadsoftware hin. So erscheint typische Schadsoftware nicht in der Liste der aktiven Prozesse oder in der Liste der aktiven Threads, die Prozesse sind aber durch signaturbasierte Erkennungsmechanismen im Hauptspeicher trotzdem auffindbar.

Da das Betriebssystem während der Analyse weiterarbeitet, können deswegen auch Widersprüche entstehen. Um diese zu minimieren, wird die Analyse mit unterschiedlichen Wartezeiten mehrfach durchgeführt.

C. Systemaufrufprotokoll

Eine andere Anwendungsmöglichkeit für HyperSleuth ist die Erstellung eines Systemaufrufprotokolls. Solch ein Protokoll hilft bei der Analyse unbekannter Software, indem jeder Systemaufruf protokolliert wird. Anhand dieses Protokolls kann auf die internen Abläufe unbekannter Software geschlossen werden.

Da bei der aktuellen Implementierung der Virtualisierungsunterstützung keine Auslösung eines Kontextwechsels in den Hypervisor durch die Systemaufrufe `sysenter`/`sysexit` möglich ist, geht HyperSleuth folgenden Weg: Jeder Systemaufruf mit `sysenter` löst über die Schattenseitentabelle einen Seitenfehler aus, über den in den Hypervisor gewechselt wird. Dort wird der Grund für den Seitenfehler erkannt, der eigentlich ausgeführte Systemaufruf gespeichert und anschließend die Ausführung des Gastsystems mit dem tatsächlichen Systemaufruf fortgesetzt. Der protokollierte Systemaufruf wird vom Hypervisor wie bei der Erstellung des Speicherabbilds über das Netzwerk an die forensische Workstation übertragen, auf der das eigentliche Protokoll erstellt wird.

VI. Zusammenfassung

HyperSleuth demonstriert eine Möglichkeit, wie die eigentlich aus der Schadsoftware kommende Idee der Virtualisierung eines laufenden Systems für forensische Untersuchungen nutzbar gemacht wird. Die Idee wird ergänzt um eine softwarebasierte Absicherungslösung, die garantiert, dass der Virtualisierungsvorgang ohne Modifikationen durch Dritte vollzogen wurde. Technisch basiert die Virtualisierung auf einfacher, aber effektiver Hardwareunterstützung in modernen Prozessoren und einem minimalen Hypervisor. Dieser virtualisiert die MMU, ermöglicht den Zugriff auf Eingabe-/Ausgabegeräte und betreibt die Netzwerkkarte in einer Art Time-Sharing Modus.

Damit sind unterschiedliche Analysefunktionen denkbar, von denen drei näher beschrieben wurden: ein Modul zur

Erstellung eines Speicherabbilds, ein Lügendetektor zur Erkennung von Schadsoftware und ein Modul zur Systemaufrufprotokollierung. So ist eine verlässliche, forensische Live-Analyse möglich, die gegenüber der Festplattenanalyse mehr, gezielter und schneller Informationen liefert.

Verbesserungspotential liegt in der Art der Ausführungsabsicherung, die in HyperSleuth verwendet wird. Das softwarebasierte Verfahren muss für jede Hardware einzeln kalibriert werden und ist daher in der Praxis nur bedingt tauglich. Hier könnten hardwaregestützte Verfahren eine ähnliche Vereinfachung wie die hardwarebasierte Virtualisierung bieten. Außerdem überschreibt das Laden des Hypervisors zumindest kleine Teile des Hauptspeichers, die dann für eine Analyse nicht mehr zur Verfügung stehen.

LITERATUR

[1] G. Neiger, A. Santoni, F. Leung, D. Rodgers, and R. Uhlig, "Intel virtualization technology: Hardware support for efficient processor virtualization," *Intel Technology Journal*, vol. 10, no. 3, 2006.

[2] AMD, "AMD Virtualization." [Online]. Available: http://sites.amd.com/us/business/it-solutions/virtualization/Pages/virtualization.aspx

[3] J. Rutkowska, "Subverting vista kernel for fun and profit," *Black Hat USA*, 2006.

[4] J. Butler and P. Silberman, "Raide: Rootkit analysis identification elimination," *Black Hat USA*, 2006.

[5] M. Sharif, W. Lee, W. Cui, and A. Lanzi, "Secure in-vm monitoring using hardware virtualization," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 477–487.

[6] L. Martignoni, A. Fattori, R. Paleari, and L. Cavallaro, "Live and trustworthy forensic analysis of commodity production systems," in *Recent Advances in Intrusion Detection*. Springer, 2011, pp. 297–316.

[7] L. Martignoni, R. Paleari, and D. Bruschi, "Conqueror: tamper-proof code execution on legacy systems," *Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 21–40, 2010.

[8] T. Garfinkel and M. Rosenblum, "A virtual machine introspection based architecture for intrusion detection," in *Proc. Network and Distributed Systems Security Symposium*, 2003.

[9] A. Dinaburg, P. Royal, M. Sharif, and W. Lee, "Ether: malware analysis via hardware virtualization extensions," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 51–62.

[10] R. Riley, X. Jiang, and D. Xu, "Guest-transparent prevention of kernel rootkits with vmm-based memory shadowing," in *Recent Advances in Intrusion Detection*. Springer, 2008, pp. 1–20.

[11] A. Seshadri, M. Luk, N. Qu, and A. Perrig, "Secvisor: A tiny hypervisor to provide lifetime kernel code integrity for commodity oses," in *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6. ACM, 2007, pp. 335–350.

[12] Trusted Computing Group, "Trusted platform module." [Online]. Available: http://www.trustedcomputinggroup.org/developers/trusted_platform_module

[13] Volatile Systems LLC, "Volatility." [Online]. Available: https://www.volatilesystems.com/default/volatility