Analyse von Würmern und Bots

Cornelius Diekmann
Betreuer: Lothar Braun
Seminar Future Internet WS09/10
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik
Technische Universität München
Email: diekmann@in.tum.de

***Kurzfassung*— Die Bedrohung durch Würmer und Bots im Internet nimmt stetig zu. Sowohl im akademischen, als auch im kommerziellen Umfeld werden Wurm- und Botdateien über verschiedene Honeypots gesammelt. Um die Bedrohung, die von ihnen ausgeht, zu verstehen und zu bekämpfen, stellt sich die zentrale Frage, was diese Dateien auf dem Hostcomputer und im Netzwerk machen. In dieser Arbeit wird gezeigt, wie man gesammelte Würmer und Bots statisch und dynamisch analysieren kann. Es zeigt sich, dass ein Großteil der Analyse automatisch durchgeführt werden kann, jedoch ist zumindest für die Auswertung und Nutzung der Ergebnisse ein menschlicher Operator nötig.**

***Schlüsselworte*— Würmer, Bots, Botnet, Bot-Netz, statische Analyse, dynamische Analyse, Malware Analyse**

I. Einleitung

Die Anzahl und Artenvielfalt von Würmern und Bots steigt seit geraumer Zeit rapide an. Längst sind Würmer und Bot-Netze kein Spielzeug von interessierten Jugendlichen mehr, sondern eine Bedrohung, die aus dem professionellen Umfeld mit kriminellen Absichten betrieben wird. Im Internet vergeht beispielsweise kein Tag ohne eine sogenannte „DDoS Attacke“, wie McIntyre in [1] zeigt. Durch Honeypot Systeme wird versucht, ein Exemplar dieser Schädlinge zu erhalten. Die große Anzahl bekannter Schädlinge[1] zeigt, dass diese Malware dann so schnell wie möglich analysiert werden muss, um der Bedrohung entgegenzuwirken. Die Analyse sollte aufgrund des hohen Malwareaufkommens automatisch geschehen.

Der folgende Abschnitt II beschäftigt sich mit der Begriffserklärung von Würmern und Bots. Nachdem geklärt wurde, welche möglichen Funktionen diese Programme aufweisen und welches Verhalten von ihnen bei der Analyse zu erwarten ist, wird in den darauffolgenden Abschnitten III und IV auf die Untersuchung von unbekannten Wurm- und Bot-Binärdateien eingegangen. Dieser Punkt untergliedert sich in die statische und dynamische Analyse. Schließlich fasst Kapitel V zusammen, wie die gesammelten Ergebnisse im Kampf gegen Bedrohungen eingesetzt werden können.

II. Überblick über Würmer und Bots

Als Wurm wird laut Kohring und Schafmeister in [3] ein Computerprogramm bezeichnet, dass sich selbst von einem Computer auf einen anderen kopiert. Dabei ist es nicht auf die aktive Hilfe des Benutzers angewiesen. Nach dieser Definition ist ein Wurm keine Schadsoftware, jedoch werden Würmer in den meisten Fällen nur entwickelt, um Schadcode auf den betroffenen Computern zu installieren. Als Bot-Netz bezeichnet man ein großflächiges Netzwerk von Computern, welche eine spezifische Bot-Software ausführen. Diese Computer werden „Zombies“ genannt. Durch die Bot-Software lassen sich die befallenen Computer fernsteuern, wodurch ein Bot-Netz-Operator seine Armee von Zombies über einen oder mehrere Command-and-Control (C&C) Server kontrollieren kann [4]. Heutzutage verschwimmt jedoch die klare Trennung zwischen Würmern und Bots, denn beide Gattungen werden unter dem Begriff Malware zusammengefasst. In dieser Arbeit werden deshalb die Wörter Bot und Wurm weitgehend als Synonym verwendet.

Für einen Wurm gibt es verschiedene Möglichkeiten, sich zu verbreiten. Während Bots von sogenannten „Scriptkiddies“ sich bevorzugt automatisch über Sicherheitslücken auf den niederen OSI-Schichten verbreiten [5], verbreiten sich kommerzielle und professionelle Bot-Netze bevorzugt auf der Anwendungsschicht, beziehungsweise direkt über Social Engineering [1], [4]. Eine einfache Methode hierfür ist der Massenversand von präparierten Emails, mit dem Wurm im Anhang oder Links auf die Malware. Beide Formen sind heutzutage anzutreffen.

Damit Würmer ihre Arbeit ungestört und dauerhaft erledigen können, nisten sie sich meist sehr tief auf dem befallenen System ein. Dazu gehört beispielsweise, dass die Malware bei jedem Systemstart ausgeführt wird und meistens dauerhaft als Hintergrundprozess aktiv bleibt. Des Weiteren können Würmer versuchen, ihre Präsenz auf dem System vor dem Benutzer und vor Sicherheitssoftware zu verstecken. Damit Virenscanner die Malware nicht finden können, kann die Binärdatei polymorph[2] sein. TimeLock Puzzles, wie sie von Nomenumbra in [6] vorgestellt werden, stellen eine weitere mächtige Möglichkeit dar, um vor Virenscannern unentdeckt zu bleiben. Würmer haben auch die Möglichkeit, laufende Schutzsoftware zu deaktivieren. Rootkits hingegen nisten sich sehr tief in das Betriebssystem ein und können dafür sorgen, dass Systemfunktionen in bestimmten Fällen falsche Antworten geben. So kann

[1] Die Datenbank von Symantec's „Endpoint Protection 11“ enthält am 15.09.09 insgesamt 4568898 bekannte Bedrohungen und Risiken [2]

[2] Polymorphie bedeutet in diesem Zusammenhang, dass ein Programm unterschiedliche Binärrepräsentationen annehmen kann, semantisch dabei jedoch gleich bleibt.

beispielsweise die Anfrage an das Betriebssystem, ob sich eine bestimmte Datei auf der Festplatte befindet, oder ein bestimmter Prozess läuft, von dem Rootkit abgefangen werden, um die Existenz der Malware zu verbergen. Durch Bootkits [7] kann sogar sichergestellt werden, dass die Malware bereits vor dem Systemstart ausgeführt wird und somit die absolute Kontrolle über das laufende Betriebssystem hat. Hinzu kommt, dass Bot-Netze ihr C&C Protokoll verschlüsseln können und zur Kommunikation Protokolle wie HTTP verwenden können, wodurch Bot-Netz-Traffic in einem Netzwerk sehr schwer zu erkennen ist. Es ist auch denkbar, dass Bots ihr Host-System nach Debuggern absuchen und eine vm[3]-Erkennung durchführen, um der automatischen Analyse zu trotzen. Durch automatische Updates der Botsoftware können die Bot-Netz-Operatoren dem Benutzer immer einen Schritt voraus sein.

Die Hauptaufgabe einer Malware ist jedoch meistens eine Andere, einige mögliche Funktionen seien hier kurz aufgelistet:

- Zerstörung von Daten
- Zerstörung von Hardware
- Erpressung der Benutzer, zum Beispiel durch Verschlüsselung ihrer Daten
- Diebstahl sensibler Daten
- Installation und Transport von Viren
- Eingliederung des betroffenen Rechners in ein Bot-Netz

Besonders auf den letzten Punkt wird diese Arbeit genauer eingehen, da dieser Angriff sehr weit verbreitet ist und weitreichende Auswirkungen für die Opfer haben kann.

Bot-Netze werden unter Anderem für die folgenden Aufgaben eingesetzt [5]: Durch Distributed Denial-of-Service(DDoS) Attacken wird versucht, den Dienst eines Ziels in die Knie zu zwingen. Dies kann durch eine einfache Flut von Paketen und der daraus resultierenden Überladung der Bandbreite des Opfers geschehen, oder durch spezielle Attacken, die alle Rechenressources des Opfers aufbrauchen.

Der Versand von ungewollten Massenemails (Spamming) erfolgt fast ausschließlich über Bot-Netze. In [4] wird gezeigt, dass circa 79% aller Spamnachrichten von nur sechs verschiedenen Bot-Netzen verschickt werden.

Durch Traffic Sniffing, Keylogging, und Man-in-the-Browser Attacken [8] können sensible Informationen wie Benutzernamen, Passwörter und Kontodaten gestohlen werden, oder Überweisungen auf fremde Konten umgelenkt werden. Weitentwickelte Malware wie ZeuS kann sogar Screenshots aller Benutzereingaben erstellen, um speziell Loginvorgänge beim Onlinebanking abzufangen, wie Kurtenbach in [9] aufgedeckt hat. Bot-Netze können eingesetzt werden, um Identitätsdiebstahl im großen Umfang zu betreiben. Ein Zombie-PC lässt sich auch zum Betreiben einer Phishing-Seite missbrauchen.

Oft nutzt ein Bot-Netz-Operator sein Bot-Netz, um neue Zombies in sein Netz einzugliedern und somit noch mächtiger zu werden.

[3] virtuelle Maschine

Viele Bot-Netze haben die Funktionalität, beliebige Programme nachzuladen und auszuführen. So kann ein Bot-Netz-Operator nicht nur Updates für seine Zombies verteilen, er kann seinem Netz befehlen, eine spezielle Seite zu besuchen und dort Werbung anzuklicken. Die Werbeeinnahmen erhält somit der Bot-Netz-Operator. Zusätzlich kann auch Werbung auf den Zombie-PCs installiert werden, wodurch der Bot-Netz-Operator eine weitere Einnahmequelle hat. Im Speziellen kann auch ein Werbedienstleister (wie beispielsweise Google AdSense) missbraucht werden, um gezielt Werbung eines konkurrierenden Unternehmens anzuklicken. Somit wird der Klick-Zähler künstlich erhöht und das angegriffene Unternehmen muss entweder hohe Werbekosten an Google zahlen, oder das Kontingent wird aufgebraucht und es wird keine Werbung mehr für dieses Unternehmen im laufenden Monat angezeigt.

Da jeder Zombie eine eigene IP-Adresse hat, eignen sich Bot-Netze auch hervorragend für die Manipulation von Online-Umfragen/Spielen.

Oft werden Bot-Netze von ihren Operatoren verkauft oder vermietet. Viele Bot-Netze bieten die Möglichkeit, beliebige ausführbare Dateien nachzuladen und sind damit für jeden erdenklichen Zweck einsetzbar. Wie das Beispiel ZeuS [9] zeigt, werden Bot-Netze vermietet, verkauft und gestohlen. Es existiert sogar ein ganzer Bereich der sich nur mit Support, wie dem Einrichten eines neuen Bot-Netzes, dem Verkauf eines Bot-Netzes und dem Mieten eines Bot-Netzes beschäftigt. In [9], [10] wird gezeigt, dass Bot-Netze von jedem – sogar ohne technische Vorkenntnisse – betrieben werden können.

Während Bacher et al. in [5] zu dem Schluss kommen, dass viele Bot-Netze von sogenannten „Scriptkiddies“ geleitet werden, zeigen die Studien [4], [9], dass Bot-Netze im großen Stil zu kommerziellen Zwecken genutzt werden. Dabei muss häufig zwischen den technisch-fachkundigen Malwareautoren und den oft technisch unwissenden, kriminellen Nutzern des Bot-Netzes unterschieden werden.

A. Kommunikation mit dem C&C Server

Es gibt verschiedene Möglichkeiten, wie ein Bot mit seinem C&C Server Kontakt aufnehmen kann. Seit den Anfangszeiten der Bot-Netze verbinden sich die Zombies zu einem IRC-Server und erhalten dort ihre Befehle. Modernere Bots nutzen Protokolle wie HTTP oder SMTP, da so der Bot-Traffic schwerer zu erkennen ist. Die meisten Bots heutzutage verbinden sich zu einem zentralen C&C Server, wobei die Möglichkeiten der Bots diesen Server zu finden, sehr unterschiedlich sind. Einfache Bots haben die IP-Adresse ihres C&C Server statisch in ihrem Code eingebettet, während sehr weit entwickelte Ansätze, wie DNS Fast Flux und algorithmisches Durchprobieren mehrerer DNS-Namen zu finden sind. Auch p2p Netzwerke zur Bot-Netz-Steuerung sind anzutreffen [4]. Einen detaillierten Überblick über die Command und Control Strukturen eines Bot-Netzes bieten D. und S. Dietrich in [11].

III. Statische Analyse

Die statische Analyse versucht eine Whitebox-Sicht auf den Code und die Logik der Malware zu erlangen, das bedeutet,

man versucht die interne Arbeitsweise der Malware zu verstehen. Der Reverse-Engineering-Prozess kann in zwei Phasen unterteilt werden: Disassemblierung und Dekompilierung. Die Aufgabe der Disassemblierungsphase besteht darin, aus der binären Repräsentation des Programms eine symbolische Darstellung in Form von Assembler Code zu gewinnen. Dekompilierung ist der Prozess, aus diesem Code die Logik und die semantischen Strukturen (und sogar den Quellcode in einer menschenlesbaren Hochsprache) zu extrahieren [12].

A. Disassemblierung

Die Disassemblierung kann heutzutage zu großen Teilen automatisiert mit Hilfe von Programmen durchgeführt werden. Ein sehr bekannter Disassembler ist IDA Pro [13]. Jedoch versuchen Programmautoren seit vielen Jahren ihre Binärdateien vor einer automatischen Disassemblierung zu schützen. Beispielsweise werden sogenannte „Junk Bytes“ an nicht erreichbare Stellen der Binärdatei eingefügt, wodurch viele Disassembler verwirrt werden. Durch die variable Länge der Maschinenbefehle auf x86-Rechnern, wird der komplette darauf folgende Maschinencode falsch interpretiert.

```
Adresse     Befehl
00000000    jmp ZIEL
00000002    <Junk Bytes>
00000004    Rücksprungadresse:
            weiter im Programm

——— Das Disassemblierungsergebnis ———
00000000    jmp ZIEL
00000002    AND
00000004    Parameter 1 für AND
00000006    Parameter 2 für AND
00000008    Falschinterpretation ab hier
```

Abbildung 1. Der obere Teil zeigt einen mit Junk Bytes beispielhaft präparierten Code. Der untere Teil zeigt, wie die Disassemblierung dabei fehlschlagen kann.

Abbildung 1 zeigt im ersten Teil des Beispiels einen durch zwei Junk Bytes präparierten Code. Die Junk Bytes sollen der Binärrepräsentation des Maschinenbefehls AND entsprechen. Im originalen Programm werden die Junk Bytes niemals angesprungen. Ein schlechter Disassembler kann dies jedoch nicht immer erkennen und interpretiert diese Bytes als Maschinenbefehl, wodurch die darauf folgenden Bytes als Parameter interpretiert werden und die Disassemblierung fehlschlägt. Jedoch gibt es auch automatisierte Techniken, die dies erkennen und behandeln können [12].

B. Dekompilierung

Allerdings wird der Einsatz eines Menschen nötig, wenn der Code der Malware verschleiert[4] ist. Dabei liefert der Disassembler einen Code, der korrekt, jedoch nicht zu gebrauchen ist. Das Beispiel des Rustock Bots [14] zeigt, dass eine Malware sich auf verschiedene Arten verschleiern kann, um ihre eigentliche Logik zu verbergen. Rustock nutzt einerseits verschachtelte Schleifen, um den Code zuerst zu entschleiern bevor er ausgeführt wird. Zusätzlich ist ein zweiter Teil mit XOR verschleiert und muss erst entschlüsselt werden, bevor Rustock ihn ausführt. Eine weitere beliebte Methode von Malwareautoren ist es, den eigentlichen Payload mit einem Packer zu verändern und dadurch zu verschleiern. Vor der Ausführung entpackt sich die Malware selbst. Durch die Verschleierung gelingt es einer Malware auch, verschiedene semantisch äquivalente Versionen zu verbreiten, die jedoch komplett unterschiedliche binäre Repräsentationen haben. So können viele signaturbasierte Virenscanner ausgehebelt werden.

Für einen Menschen ist es normalerweise nicht möglich, aus dem verschleierten Code nützliche Informationen herauszulesen. Deswegen bietet es sich an, mit Emulatoren wie [15], die Malware so weit laufen zu lassen, bis sie sich selbst entschlüsselt hat und mit dieser entschlüsselten Binärdarstellung des Payloads die statische Analyse von vorne zu beginnen.

Wenn nun die Malware in ihrer reinen Form vorliegt, lassen sich sehr schnell sehr viele Informationen herauslesen. Eine Suche nach Zeichenketten kann Aufschluss geben, welche Kommandos beispielsweise ein Bot unterstützt. Die Liste der importierten und exportierten Funktionen gibt Hinweise auf die Arbeitsweise der Malware, manchmal lässt sich sogar ein geheimer Schlüssel, der zu der Kommunikation mit einem C&C Server genutzt wird, extrahieren.

C. Vorteile der statischen Analyse

Durch die statische Analyse kann man eine vollständige Sicht auf die Malware erhalten. Selbst Codefragmente der Malware, die in Testläufen nicht ausgeführt werden, können entdeckt werden. Zusätzlich besteht keine Gefahr für andere Computer im Netzwerk.

D. Nachteile der statischen Analyse

Die Analyse ist sehr aufwendig und bei verschleierten Binärdateien in den meisten Fällen nicht automatisierbar. Der resultierende Assemblercode ist sehr schwer zu verstehen und eine Rückführung in einen möglichen Quellcode ist sehr zeitraubend wenn nicht unmöglich. Oftmals ist die Aussagekraft der statischen Analyse auch nicht groß genug, da die Malware zum Beispiel dynamisch neue Befehle von ihrem C&C Server erhält, oder Stromchiffren wie RC4 zur Verschlüsselung verwendet (bsp. Rustock, ZeuS), bei denen sich der Schlüssel erst zu Ausführungszeit extrahieren lässt.

IV. Dynamische Analyse

Bei der dynamischen Analyse wird die Malware in einer Testumgebung ausgeführt. Dabei wird der Schadcode auf einem jungfräulichen Host-System installiert. Während und nach der Laufzeit des Wurms wird versucht, alle Aktionen, die die Malware durchführt, zu protokollieren. Das folgende Kapitel IV-A beschäftigt sich mit der Analyse der Handlungen der Malware auf dem Host, Kapitel IV-B mit der Analyse der Tätigkeiten im Netzwerk. Auf diese Weise können Rückschlüsse auf die Arbeitsweise der Malware getroffen werden. Außerdem kann

[4] Verschleierung - von dem Englischen ’obfuscation’

ein Profil der Malware erstellt werden, wodurch infizierte Computer schnell erkannt werden, Firewallregeln aufgestellt werden und Statistiken und Gefahrenanalysen erstellt werden können. Das Kapitel V geht genauer auf die Verwendung der Ergebnisse ein.

A. Analyse der Aktionen auf dem Host

Eine sehr einfache Möglichkeit um Rückschlüsse der Aktivitäten der Malware auf dem Host zu erhalten, ist es, die Malware auf einem frisch installierten System auszuführen und die Änderungen am System vor und nach der Ausführung zu vergleichen. So lässt sich leicht herausfinden, wie weit die Malware sich ins System integriert und ob sie sogar ein Rootkit installiert. Außerdem kann die Malware keine Auffälligkeiten – wie Analysesoftware – im laufenden System finden und verhält sich somit garantiert wie auf einem normalen System. Diese Analyse ist jedoch sehr aufwendig und liefert keine Aussagen darüber, was die Malware während der Laufzeit macht.

Eine weitaus bessere Möglichkeit stellt spezielle Analysesoftware wie CWSandbox dar [16].

CWSandbox führt die verdächtige Datei in einer kontrollierten Win32 Umgebung aus. Es werden alle System Calls, Änderungen am Dateisystem und der Registry, gestarteten Prozesse, sowie Netzwerkaktivitäten aufgezeichnet. Dabei kann das Ergebnis sowohl in menschenlesbarer als auch in maschinenlesbarer Form ausgegeben werden. Abbildung 2 zeigt einen Ausschnitt eines Analyseprotokolls im XML-Format. Zur Protokollierung der Aktionen der Malware nutzt CWSandbox die Techniken API Hooking[5] und DLL Injection. Das bedeutet, CWSandbox übernimmt die Kontrolle über alle Aufrufe der Windows-API, die die Malware während ihrer Laufzeit macht.

CWSandbox besteht im Wesentlichen aus zwei Teilen: cwsandbox.exe und cwmonitor.dll. Der erste Teil ist die eigentliche Analyseplatform. Er startet die zu untersuchende Malware im suspended mode, führt als Urlader alle nötigen API Hooks aus und bereitet die DLL Injection vor. Danach wird die Malware gestartet und cwmonitor.dll in den Adressraum der Malware injiziert. cwmonitor.dll enthält alle nötigen Hook Funktionen um alle Windows API Aufrufe der Malware zu protokollieren und an cwsandbox.exe zu senden. Zusätzlich kann sich cwmonitor.dll in jeden neu erzeugten Prozess und Thread der Malware selbstständig injizieren.

Abbildung 3 zeigt beispielhaft einen gehookten Aufruf einer Windows API-Funktion der Malware. Beim Start der Malware – und jedem neuen Prozess, den die Malware startet – werden alle Aufrufe der Windows API im Speicher der Malware gehookt: Die ersten sechs Bytes jeder API Funktion werden mit einen Sprung auf den jeweiligen Hook Code überschrieben (1). Hier können nun alle API Aufrufe und Parameter protokolliert werden und das Ergebnis direkt an die Analyseplatform cwsandbox.exe geliefert werden (2). Danach werden die überschriebenen Anweisungen der Windows

[5]Unter API Hooking versteht man, dass bei jedem Aufruf einer API-Funktion – anstatt der originalen Funktion – eigener Code ausgeführt wird.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This analysis was created by CWSandbox (c) CWSE GmbH / Sunbelt Software-->
<analysis cwsversion="2.1.17" time="06.10.2009 16:20:54" file="c:\7z465.exe" md5="
        fcd1b1472302fc7283147f4df471f402" sha1="
        c36012e960fa3932cd23f30ac5b0fe722740243a" logpath="c:\cwsandbox\log\7z465.exe
        \run_1\" analysisid="885548" sampleid="762367">
<calltree>
<process_call index="1" pid="2656" filename="c:\7z465.exe" starttime="00:01.422"
        startreason="AnalysisTarget"/>
<process_call index="2" pid="984" filename="C:\WINDOWS\system32\svchost.exe"
        starttime="00:03.328" startreason="DCOMService"/>
</calltree>
<processes>
<process index="1" pid="2656" filename="c:\7z465.exe" filesize="939956" md5="
        fcd1b1472302fc7283147f4df471f402" sha1="
        c36012e960fa3932cd23f30ac5b0fe722740243a" username="Administrator"
        parentindex="0" starttime="00:01.422" terminationtime="00:13.078" startreason
        ="AnalysisTarget" terminationreason="NormalTermination" executionstatus="OK"
        applicationtype="Win32Application">
<com_section>
<com_create_instance inprocserver32="%SystemRoot%\system32\browseui.dll" clsid="{00
        BB2765-6A77-11D0-A535-00C04FD7D062}" interfaceid="{00000000-0000-0000-C000
        -000000000046}"/>
<com_create_instance inprocserver32="%SystemRoot%\system32\browseui.dll" clsid="{03
        C036F1-A186-11D0-824A-00AA005B4383}" interfaceid="{00000000-0000-0000-C000
        -000000000046}"/>
<com_create_instance inprocserver32="%SystemRoot%\system32\browseui.dll" clsid="{00
        BB2763-6A77-11D0-A535-00C04FD7D062}" interfaceid="{EAC04BC0-3791-11D2-BB95
        -0060977B464C}"/>
<com_create_instance inprocserver32="shell32.dll" progid="lnkfile" clsid="
        {00021401-0000-0000-C000-000000000046}" interfaceid="{000214EE-0000-0000-C000
        -000000000046}" quantity="2"/>
</com_section>
<dll_handling_section>
<load_image filename="c:\7z465.exe" successful="1" address="$400000" end_address="
        $434000" size="212992"/>
<load_dll filename="C:\WINDOWS\system32\ntdll.dll" successful="1" address="$7
        C910000" end_address="$7C9C9000" size="757760"/>
<load_dll filename="C:\WINDOWS\system32\kernel32.dll" successful="1" address="$7
        C800000" end_address="$7C908000" size="1081344"/>
<load_dll filename="C:\WINDOWS\system32\USER32.dll" successful="1" address="$7
        E360000" end_address="$7E3F1000" size="593920"/>
<load_dll filename="C:\WINDOWS\system32\GDI32.dll" successful="1" address="$77
        EF0000" end_address="$77F39000" size="299008"/>
<load_dll filename="C:\WINDOWS\system32\SHELL32.dll" successful="1" address="$7
        E670000" end_address="$7EE91000" size="8523776" quantity="4"/>
<load_dll filename="C:\WINDOWS\system32\ADVAPI32.dll" successful="1" address="$77
        DA0000" end_address="$77E4A000" size="696320"/>
<load_dll filename="C:\WINDOWS\system32\RPCRT4.dll" successful="1" address="$77
        E50000" end_address="$77EE2000" size="598016"/>
<load_dll filename="C:\WINDOWS\system32\Secur32.dll" successful="1" address="$77
        FC0000" end_address="$77FD1000" size="69632"/>
<load_dll filename="C:\WINDOWS\system32\msvcrt.dll" successful="1" address="$77
        BE0000" end_address="$77C38000" size="360448"/>
<load_dll filename="C:\WINDOWS\system32\SHLWAPI.dll" successful="1" address="$77
        F40000" end_address="$77FB6000" size="483328"/>
<load_dll filename="C:\WINDOWS\WinSxS\X86_Microsoft.Windows.Common-
        Controls_6595b64144ccf1df_6.0.2600.5512_x-ww_35d4ce83\" successful="1"
        address="$773A0000" end_address="$774A3000" size="1060864" quantity="3"/>
<load_dll filename="C:\WINDOWS\system32\ole32.dll" successful="1" address="$774
        B0000" end_address="$775ED000" size="1298432" quantity="3"/>
<load_dll filename="C:\WINDOWS\system32\VERSION.dll" successful="1" address="$77
        BD0000" end_address="$77BD8000" size="32768"/>
<load_dll filename="C:\WINDOWS\system32\IMM32.DLL" successful="1" address="
        $76330000" end_address="$7634D000" size="118784" quantity="2"/>
<load_dll filename="C:\WINDOWS\system32\oleaut32.dll" successful="1" address="$770
        F0000" end_address="$7717B000" size="569344"/>
<load_dll filename="C:\WINDOWS\system32\pstorec.dll" successful="1" address="$5
        E490000" end_address="$5E49D000" size="53248"/>
<load_dll filename="C:\WINDOWS\system32\ATL.DLL" successful="1" address="$76AD0000"
        end_address="$76AE1000" size="69632"/>
<load_dll filename="C:\WINDOWS\system32\uxtheme.dll" successful="1" address="$5
        B0F0000" end_address="$5B128000" size="229376" quantity="3"/>
<load_dll filename="C:\WINDOWS\system32\RichEd20.dll" successful="1" address="$74
        DB0000" end_address="$74E1D000" size="446464"/>
<load_dll filename="C:\WINDOWS\system32\msctfime.ime" successful="1" address="$11
        F0001" end_address="$11F0001" size="0" quantity="2"/>
<load_dll filename="C:\WINDOWS\system32\msctfime.ime" successful="1" address="
        $75250000" end_address="$7527E000" size="188416" quantity="2"/>
<load_dll filename="C:\WINDOWS\system32\browseui.dll" successful="1" address="$75
        F20000" end_address="$7601D000" size="1036288" quantity="3"/>
<load_dll filename="C:\WINDOWS\system32\ntshrui.dll" successful="1" address="
        $76940000" end_address="$76966000" size="155648"/>
<load_dll filename="C:\WINDOWS\system32\NETAPI32.dll" successful="1" address="$597
        D0000" end_address="$59825000" size="348160"/>
<load_dll filename="C:\DOKUME~1\ADMINI~1\LOKALE~1\Temp\nss5.tmp\InstallOptions.dll"
        successful="1" address="$10000000" end_address="$10009000" size="36864"/>
<load_dll filename="C:\WINDOWS\system32\MSCTF.dll" successful="1" address="$746
        A0000" end_address="$746EC000" size="311296"/>
</dll_handling_section>
```

Abbildung 2. Ausschnitt eines Reports der CWSandbox im XML-Format

API ausgeführt und es wird auf die originale API Funktion zurückgesprungen (3). Der Hook Code befindet sich in der cwmonitor.dll, welche durch DLL Injection in den Speicher der Malware geladen wurde.

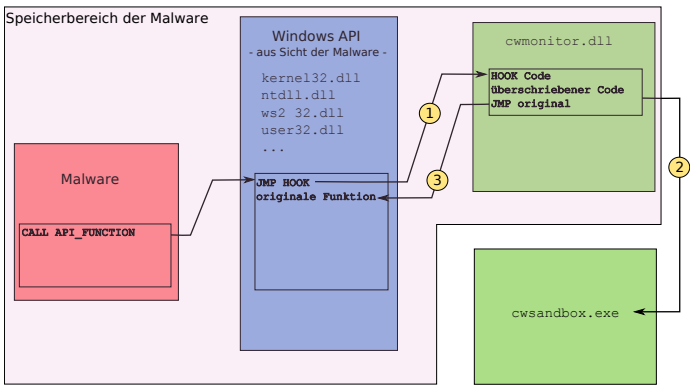

Abbildung 3. CWSandbox Funktionsweise

Die aktuelle Version von CWSandbox kann jedoch keine direkten Zugriffe auf den Kernel erkennen. CWSandbox braucht auch einen menschlichen Operator, um gegebenenfalls das Analysefensters manuell zu setzten, falls die Malware erst nach einer gewissen Zeit aktiv wird oder Techniken wie TimeLock Puzzles [6] einsetzt. Auch für die finale Auswertung der Ergebnisse ist oft ein Mensch von Nöten.

Die Verwendung von virtuellen Maschinen erleichtert die Arbeit des Wiederherstellens des ursprünglichen Systemzustands. Jedoch existiert Malware, die das laufende System untersucht und gegebenenfalls ihre Aktivitäten einstellt oder sich auf einem Analysesystem anders verhält. Dieses Verhalten ist beispielsweise bekannt, falls die Malware einen laufenden Debugger erkennt, oder feststellt, dass sie in einer virtuellen Maschine ausgeführt wird. Zu diesem Zweck muss getestet werden, ob die Malware eine vm-Erkennung durchführt. Dies kann zum Beispiel durch den Vergleich der „network fingerprints“ der Malware (siehe IV-B.2) auf einem realen und einem virtuellen System erfolgen.

B. Analyse der Aktionen im Netz

Für diese Analyse müssen alle Nachrichten, die die Malware über das Netzwerk empfängt und sendet, protokolliert werden. Dafür genügt es, einen Paketsniffer in einer Man-in-the-Middle Position zwischen dem Analysesystem und dem Netzwerk zu installieren.

1) Analyse der ausgehenden Verbindungen: Über die ausgehenden Verbindungen eines Bots erhält man direkte Informationen über die Aktionen des Bots. Oft wird anhand der verwendeten Protokolle und des Traffic-Aufkommens bereits ersichtlich, ob ein Bot an einer DDos-Attacke teilnimmt oder Spam versendet. Andere Aktivitäten erfordern oft eine genauere Analyse des Traffics. Ein Bot verbindet sich jedoch unweigerlich mit seinem C&C-Server um Instruktionen zu erhalten. Wenn diese Kommunikation mitgeschnitten wurde, kennt man einen C&C-Server zu dem jeweiligen Bot-Netz. Durch gezielte Firewallregeln ist es auch möglich, den Bot an der Kommunikation mit seinem C&C Server zu hindern, um so herauszufinden, nach welchem Algorithmus der Bot versucht, einen neuen Server zu finden. Umgekehrt kann man so einem Bot, der inaktiv wurde, weil er keinen C&C Server finden konnte, die Existenz eines solchen Servers vorspielen um ihn in Aktion zu erleben. Jedoch muss dazu das verwendete C&C-Protokoll bekannt sein.

2) Klassifizierung der Malware: Viele Würmer und Bots existieren in vielen Ausführungen und können deshalb meistens nicht automatisch von Virenscannern als Duplikate erkannt werden, da ein Virenscanner nur versucht herauszufinden *ob* es sich bei einer Datei um eine Malware handelt, nicht *um Welche*. Die Universität von Washington nutzt deshalb in ihren Experimenten mit Botlab [4] einen sogenannten „network fingerprint“. Dieser wird wie folgt gebildet: Sei X die Menge aller ausgehenden Verbindungen eines Programms, wobei jede Verbindung durch das Tupel

`<Protokoll, IP Adresse, DNS Adresse, Port>`

dargestellt wird. Damit zufällige Verbindungen ausgeschlossen werden, wird die Datei mehrfach ausgeführt und der network fingerprint aus der Schnittmenge über alle X gebildet. Experimentell konnte für spamorientierte Bot-Netze festgestellt werden, dass zwei Bots mit ihren network fingerprints N_1 und N_2 zu einer sehr großen Wahrscheinlichkeit identisch sind, falls gilt:

$$\frac{N_1 \bigcap N_2}{N_1 \bigcup N_2} > 0.5$$

3) Analyse von Spam: Ein großer Teil aller weltweit versandten Emails besteht aus Spamnachrichten, welche von Bot-Netzen erzeugt wurden. Um diese zu analysieren, bietet es sich an, den Spam, der auf den eigenen Mailservern eingeht, sowie der Spam, der durch die lokal untersuchte Malware versendet wird, in Beziehung zu setzen. So lässt sich Spam sehr akkurat einem Bot-Netz zuordnen und dadurch lassen sich Rückschlüsse auf seine Größe ziehen. Es sei jedoch vermerkt, dass es keine Seltenheit ist, dass mehrere Bot-Netze an der gleichen Spamkampagne teilnehmen [4]. Dies ist ein Indiz für die starke Kommerzialisierung der Internetkriminalität.

4) Verhinderung von Blacklisting: Um zu verhindern, dass die eigene IP Adresse über kurz oder lang von den Bot-Netz-Operatoren auf die schwarze Liste gesetzt wird, ist es ratsam, diese nicht preiszugeben. Dies kann erreicht werden, indem man jeden Traffic, der ins Internet gelassen wird, über ein Anonymisierungsnetzwerk wie TOR [17] leitet.

5) Gefahren der dynamischen Analyse: Da bei der dynamischen Analyse die Schadsoftware tatsächlich ausgeführt wird, besteht eine Gefahr für alle an das das Netzwerk angeschlossenen Computer. Es gilt deshalb das Risiko für angrenzende Systeme zu minimieren und dabei trotzdem gute Analyseergebnisse zu erzielen. Eine Möglichkeit besteht darin, den Rechner komplett vom Netz zu trennen, worunter die Analyseergebnisse jedoch stark leiden. Viele Malware testet ihre Konnektivität bevor sie aktiv wird, die meisten Bots stellen sogar alle Aktivitäten ein, falls sie sich nicht zu ihrem C&C Server verbinden können.

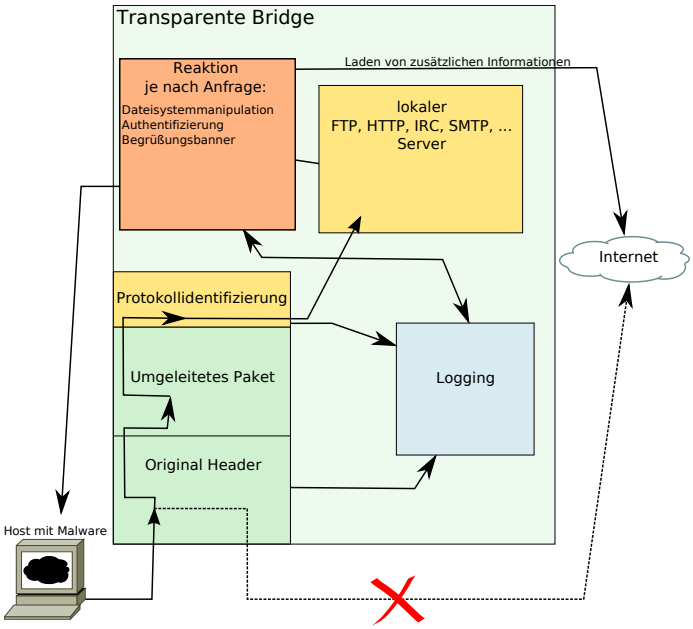

Abbildung 4. TrumanBox Schema

Einen erfolgreichen Ansatz um der Malware so viel Freiraum wie nötig und so wenig Zugang zum Netzwerk wie möglich zuzugestehen, bieten Systeme wie Honeywalls oder die TrumanBox [18]. Die TrumanBox wird wie eine transparente Bridge – und damit für die Malware unsichtbar – zwischen dem Host und dem restlichen Netzwerk eingerichtet. Sie kann Verbindungsversuche der Malware auf generische, lokale Dienste transparent umleiten und spielt der Malware so eine echte Internetumgebung vor, ohne einen fremden Rechner zu gefährden. Sie arbeitet auf allen ISO/OSI Schichten um so die Analyseergebnisse zu optimieren. Ein solches System muss in der Lage sein, eine Dienstanfrage auch zu erkennen, wenn sie auf einem nicht standardisierten Port gestellt wird. Außerdem muss der emulierte Dienst bei Protokollen, bei denen der Server dem Client zuerst eine Nachricht schickt, die richtige Nachricht ausliefern (beispielsweise die Begrüßungsbanner bei FTP oder SMTP). Des Weiteren sollte das System sich so verhalten, wie es die Malware erwartet, dies beinhaltet unter anderem die Dateisystemstruktur auf dem Server bei Diensten wie FTP und HTTP. Abbildung 4 zeigt diesen schematischen Aufbau. Alle Anfragen der Malware an das angrenzende Netzwerk (hier: Internet) werden transparent auf lokale Dienste umgeleitet. Es werden alle wichtigen Daten protokolliert, sodass der lokale Dienst den Erwartungen der Malware entsprechend reagieren kann. Falls nötig, können sogar Informationen von dem real angefragten Server im Internet nachgeladen werden.

Da sich das hier beschriebene System in einer Man-in-the-Middle Position befindet, können alle relevanten Informationen mitgeloggt werden.

C. Vorteile der dynamische Analyse

Nur durch die dynamische Analyse ist es möglich, in relativ kurzer Zeit, automatisiert und mit relativ geringem Aufwand, sehr viel Malware sehr effizient zu analysieren.

D. Nachteile der dynamische Analyse

Die gewonnenen Ergebnisse müssen nicht vollständig sein, wie Kapitel III-C bereits gezeigt hat. Der bekannte Wurm Conficker aus den Jahren 2008/09 enthielt beispielsweise ein Feature, welches bis zum 1. April 2009 im Wurm deaktiviert war und so durch die dynamische Analyse nicht vor diesem Datum erkannt werden konnte. Noch schwieriger gestaltet sich die Analyse, wenn ein Bot verschlüsselt kommuniziert. Aus dem Mitschnitt der versendeten Nachrichten wird nicht klar, welche Informationen der Bot gesendet und empfangen hat. Da Malware sehr individuell arbeitet, ist ein automatisches System zum Schutz der angrenzenden Systeme allein nicht ausreichend. Teilweise ist eine manuelle Filterung der ausgehenden Verbindungen nötig, um bessere Analyseergebnisse zu erreichen oder den ganzen Funktionsumfang der Malware auszulösen.

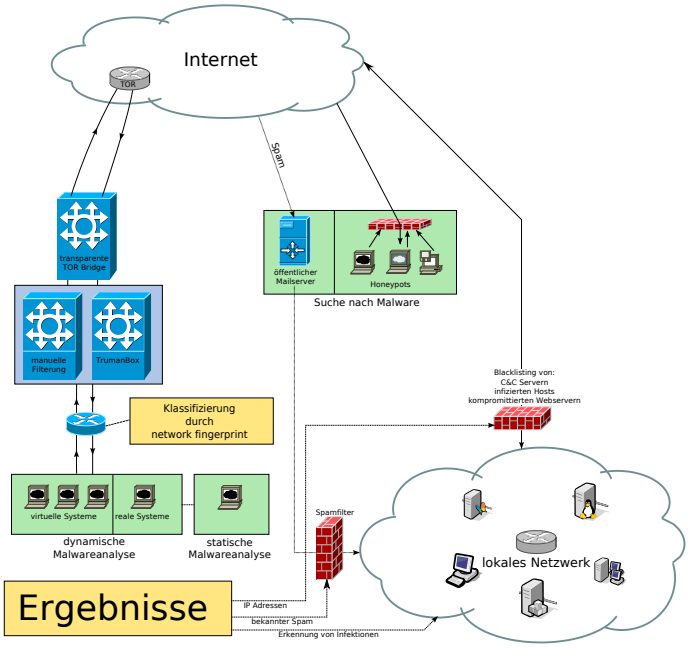

Abbildung 5. Darstellung einer kompletten Analyseumgebung.

V. Verwendung der Ergebnisse

Die gewonnenen Ergebnisse können vielseitig genutzt werden. Einerseits erhält man wichtige Erkenntnisse über die Arbeits- und Funktionsweise von Würmern und Bots und über die Vorgehensweise von Bot-Netz-Operatoren. Andererseits erhält man aktuelle Informationen über neuartige Bedrohungen. Aus der Analyse lässt sich sehr oft der C&C Server eines Bots bestimmen, oft lassen sich auch kompromittierte Webserver aufdecken, die weitere Infektionen verbreiten. Teilweise können sogar infizierte Clients durch bestimmte

Charakteristika entdeckt werden. Mit diesen Informationen lässt sich in der Firewall des lokalen Netzes (beispielsweise Firmennetz/Campusnetz) eine Blacklist mit allen bedrohlichen IP Adressen einrichten. Die dynamische Analyse eines Bots zeigt bereits, ob er allein durch Blacklisting seiner C&C Server deaktiviert werden kann. Des Weiteren lässt sich durch die Untersuchung von Spam die Trefferquote des lokalen Spamfilters verbessern. Außerdem ist es sogar möglich, infizierte Hosts im lokalen Netzwerk zu erkennen. Aufgrund des Wissens über die Infektion, durch die Analyse der Aktionen auf dem Host, lässt sich der betroffene Rechner schnell bereinigen.

Abbildung 5 zeigt ein Beispiel einer kompletten Analyseumgebung, wie sie in dieser Arbeit theoretisch beschrieben wurde. Diese Analyseumgebung wurde nach dem Beispiel von Botlab [4] entworfen. Durch verschiedene Arten von Honeypots wird Malware gesammelt. Auch eingehender Spam wird auf Links mit Malware untersucht. Die Analyse erfolgt bevorzugt auf virtuellen Systemen, da diese leichter Wiederherzustellen sind, im Einzelfall werden jedoch auch echte Systeme benutzt. Die dynamische Analyse wird bevorzugt, jedoch ist manchmal eine statische Analyse nötig. Ein Schaden an anderen an das Internet angeschlossenen Systemen, wird automatisch durch die TrumanBox verhindert, allerdings ist auch eine manuelle Filterung – falls nötig – vorgesehen. Der Traffic der Malware wird durch TOR geleitet, um ein Blacklisting der eigenen IP auf Seiten der Bot-Netz-Operatoren zu verhindern. Mit den Ergebnissen werden die Firewalls und Spamfilter eines lokalen, zu schützenden Netzwerks verbessert.

VI. Zusammenfassung und Ausblick

Wir haben die grundlegenden Funktionen und Vorgehensweisen von Würmern und Bots kennengelernt. Mit diesem Wissen haben wir uns einen Überblick verschafft, wie man unbekannte Bot- und Wurm-Dateien analysieren kann. Die Möglichkeiten, aber auch die Grenzen von automatischer Analysesoftware wurden vorgestellt. Die Qualität und Benutzerfreundlichkeit dieser Software kann sich in Zukunft noch verbessern, genauso wie die Bedrohung durch Würmer und Bots nicht geringer werden wird. Der Bedarf nach einem menschlichen Operator wird jedoch stets vorhanden sein.

Literatur

[1] S. McIntyre, "Teh Internetz are pwned - How I learned to stop worrying and love teh Internetz," in *Hacking at Random*, 2009. [Online]. Available: https://har2009.org/program/events/171.en.html

[2] Symantec, "Virendefinitionen und Sicherheits-Updates," September 2009. [Online]. Available: http://www.symantec.com/de/de/norton/security_response/definitions.jsp

[3] A. Kohring and C. Schafmeister, "Viren und Würmer: Mit welchen Mitteln gelangen Viren und Würmer in den Computer, und wie kann man sich gegen sie schützen?" 2009.

[4] J. P. John, A. Moshchuk, S. D. Gribble, and A. Krishnamurthy, "Studying Spamming Botnets Using Botlab," *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI'09)*, 2009.

[5] P. Bacher, T. Holz, M. Kotter, and G. Wicherski, "Know Your Enemy: Tracking Botnets," *The Honeynet Project*, 2005. [Online]. Available: http://www.honeynet.org/papers/bots

[6] Nomenumbra, "Countering behavior-based malware analysis through TimeLock Puzzles," in *Hacking at Random*, 2009. [Online]. Available: https://har2009.org/program/events/57.en.html

[7] P. Kleissner, "Stoned Bootkit - The Rise of MBR Rootkits & Bootkits in the Wild," in *Hacking at Random*, 2009. [Online]. Available: https://har2009.org/program/events/86.en.html

[8] D. Bachfeld, "Mit guten Karten - Sicher bezahlen im Internet," *c't*, vol. 19, pp. 92–95, 2009.

[9] C. Kurtenbach, "The ZeuS evolution - A malware case study.." in *Hacking at Random*, 2009. [Online]. Available: https://har2009.org/program/events/40.en.html

[10] BBC, "Click - Gaining access to a hacker's world," TV, Intenet, 03 2009. [Online]. Available: http://news.bbc.co.uk/2/hi/programmes/click_online/7938201.stm

[11] D. Dittrich and S. Dietrich, "command and control structures in malware," *USENIX ;login:*, vol. 32, 2007.

[12] C. Kruegel, W. Robertson, F. Valeur, and G. Vigna, "Static Disassembly of Obfuscated Binaries," 2004.

[13] "IDA Pro," 08 2009. [Online]. Available: http://www.hex-rays.com/idapro/

[14] K. Chiang and L. Lloyd, "A Case Study of the Rustock Rootkit and Spam Bot," in *The First Workshop in Understanding Botnets*, 2007.

[15] C. Eagle, "The x86 Emulator plugin for IDAPro," 2008. [Online]. Available: http://ida-x86emu.sourceforge.net

[16] C. Willems, T. Holz, and F. Freiling, "Toward Automated Dynamic Malware Analysis Using CWSandbox," *IEEE Security and Privacy*, vol. 5, no. 2, pp. 32–39, 2007.

[17] [Online]. Available: http://www.torproject.org

[18] C. Gorecki, "TrumanBox-Transparente Emulation von Internetdiensten."