Effektives Sammeln von Malware mit Honeypots

Thorsten Holz und Georg Wicherski
Laboratory for Dependable Distributed Systems
German Honeynet Project
Universität Mannheim
holz@informatik.uni-mannheim.de
georg.wicherski@mwcollect.org

Zusammenfassung

Ein Großteil der sich heutzutage autonom verbreitenden Malware infiziert weitere Opfer über bereits bekannte Schwachstellen in Netzwerkdiensten, die sich automatisiert exploiten lassen. Darüber hinaus tauchen immer mehr Bots auf, die auf der gleichen Quellcode-Familie basieren, jedoch oft mit unterschiedlichen und teilweise modifizierten Packern gepackt sind. Daher ist es wichtig, solche Malware automatisiert sammeln zu können, um effektiv neue Signaturen für Virenscanner zu erstellen oder das Verhalten von Botnetzen zu studieren.

Da es sich um bekannte Schwachstellen handelt, lassen sich reaktiv Pattern für diese Schwachstellen erstellen und ein Daemon kann implementiert werden, der verwundbare Services gegenüber sich autonom verbreitender Malware simuliert. Dabei ist es nicht nötig, diese Services vollständig und korrekt nachzubilden, sondern es ist ausreichend, eine vereinfachten Emulation der Dienste zu implementieren.

Einen solchen Daemon stellt das seit März 2005 vom Honeynet Project entwickelte Projekt mwcollect (`http://www.mwcollect.org/`) bereit.

1 Einleitung

Als *Honeypots* bezeichnet man Netzwerkressourcen (beispielsweise normale PCs, Router oder Switches), deren Wert darin besteht, angegriffen und kompromittiert zu werden. Oft sind dies Computer, die keine spezielle Aufgabe im Netzwerk haben, aber ansonsten nicht von regulären Rechnern zu unterscheiden sind. Sie sind mit spezieller Software (beispielsweise das abgewandelte Rootkit *Sebek*) ausgestattet, die die Forensik nach dem erfolgreichem Einbruch eines Angreifers erleichtert. Durch die Vielfalt an mitgeschnittenen Daten kann man deutlich mehr über das Verhalten von Angreifern in Netzwerken lernen als mit herkömmlichen forensischen Methoden. Honeypots können auch zu Netzen zusammengeschlossen werden, so genannte *Honeynets*. Dadurch kann man noch mehr über das Verhaltensmu-

ster von Angreifern lernen und gleichzeitig Angriffe auf verschiedene Plattformen beobachten [McC03b, McC03a, The05a, The05b].

Ziel der Honeypot-basierten Technologie ist es, einen Angreifer zu studieren und mehr über ihn zu lernen. Die Voraussetzung ist somit ein erfolgreicher Einbruch mit anschließenden Aktivitäten auf der für den Beobachter völlig transparenten Maschine. So kann der Betreiber des Honeypots mehr über die Vorgehensweise und die Tools von Angreifern lernen und daraus resultierend hoffentlich bessere Abwehrmechanismen erstellen. Die meisten Resultate wurden bisher mit Honeypots auf Basis von Linux erreicht. Dazu wird auf einem gewöhnlichen Computer ein (meistens älteres) Linux-System und einige Zusatzsoftware installiert. Diese Zusatzsoftware hilft vor allem, Informationen über das System zu sammeln. Außerdem werden auf dem Honeypot einige Dienste aktiviert, mit denen ein Angreifer interagieren kann. Diese Dienste haben häufig wohl bekannte Schwachstellen, so dass ein Angreifer auch (relativ) einfach den Honeypot kompromittieren kann.

Heutzutage sind allerdings viele der Angriffe auf Endsysteme automatisiert. Ein Honeypot auf Basis von Windows hat deshalb ein deutliches Problem: Installiert man Windows 2000 oder XP ohne jegliche Service Packs, so dauert unserer Erfahrung nach eine Kompromittierung im Mittel weniger als 15 Minuten. Nach einer so kurzen Zeitspanne wird der Honeypot von einer automatisierten Malware infiziert und ein *Wurm* oder ein *Bot* haben die Kontrolle über das System erlangt. Ein *Wurm* ist ein Computerprogramm, dass sich autonom durch das Ausnutzen von Schwachstellen in entfernten Systemen weiterverbreiten kann. Dazu benutzt der Wurm die Schwachstelle und kopiert sich selbst auf das entfernte System. Von dort aus wird die Verbreitung weitergeführt. Zu den bekanntesten Würmern der letzten Jahre zählen *Code Red*, *Slammer* [MPS+03], *Mydoom* und *Blaster*. Im Gegensatz dazu erlaubt ein *Bot* (auch *Drone* oder *Zombie* genannt) die Fernsteuerung des infizierten Systems durch den Angreifer. Typische Kanäle zur Fernsteuerung sind IRC oder HTTP und typische Vertreter dieser Klasse von Malware *Agobot*, *SDbot* oder *Mybot* [Hol05]. Auch Bots haben meistens eine Möglichkeit implementiert, sich innerhalb eines Netzes weiterzuverbreiten. Dazu scannen sie ebenfalls entfernte Computer auf bekannte Schwachstellen, nutzen diese aus, und laden sich danach selbst auf den entfernten Computer nach. Bots können zu *Botnetzen* zusammengeschlossen werden – einer der größten Gefahren des gegenwärtigen Internets. Mit Hilfe von Botnetzen können selbst unerfahrene Angreifer erheblichen Schaden anrichten. Eine ausführliche Einleitung zum Thema Botnetze liefert der Artikel “Know Your Enemy: Tracking Botnets” des Honeynet Projects [The05b].

Benutzt man also einen Honeypot mit Windows als Betriebssystem und schließt ihn an das Internet an, so hätte man sehr schnell die erste Malware gesammelt, aber auch die Kontrolle über den Computer verloren. Der Aufwand steigt gewaltig: Der Honeypot muss häufig überprüft, manuell gereinigt und die Malware gesammelt und analysiert werden. Ein weiterer Nachteil liegt in der Natur der Exploits: Es kann natürlich auch passieren, dass die Offsets des Exploits falsch sind (beispielsweise falsche Sprachversion oder der Exploit war für ein Windows 2000 System, hat allerdings ein Windows XP getroffen). Dies führt dann typischerweise zum Absturz oder automatischen Reboot des Systems. Selbst wenn anstatt eines ungepatchten Systems ein System mit einem Service Pack oder diversen Hotfixes verwendet wird, ändert sich das grundlegende Problem nicht: Entweder gibt es bereits Malware, die die offen gelassenen Schwachstellen automatisiert ausnutzt und nach relativ kurzer Zeit den Honeypot kompromittiert. Oder bisher existiert keine bekannte Schwachstelle und wir müssen lange warten, bis wir die erste erfolgreiche Kompromittierung beobachten können

Um nun mehr über Würmer, Bots und andere Malware zu lernen, gibt es spezialisierte Honeypots. Ein Beispiel für einen solchen Honeypot ist das Programm “mwcollect”. Es simuliert in nicht-nativer Umgebung Services, die typischerweise von automatisierter Malware zur Verbreitung genutzt werden. Diese Services lauschen auf bestimmten Ports und sind typisch für ungepatchte Windows-Installationen oder bereits infizierte Rechner. Im Moment sind beispielsweise Module verfügbar, die die Schwachstellen von DCOM (TCP Port 135) und LSASS (TCP Port 445) emulieren. Diese verhalten sich größtenteils wie ihre echten Vertreter, emulieren allerdings nur die Teile des Services, die von sich weiterverbreitender Malware ausgenutzt werden. Dies trägt maßgeblich zur Effektivität dieses Ansatzes bei.

Durch eine automatisierte Analyse des empfangenen Shellcodes an diesen Ports kann man bestimmen, wie sich die Malware weiterverbreiten will. Man kann also automatisiert ermitteln, wie sich die Malware nachladen möchte. Diese Information kann dazu genutzt werden, das Binary von einem entfernten Rechner herunterzuladen. Somit wird ein automatisiertes “Sammeln” von Malware mit Hilfe dieses Ansatzes ermöglicht.

Der Artikel ist folgendermaßen gegliedert: In Abschnitt 2 stellen wir die Idee von mwcollect detaillierter vor und gehen auf verwandte Arbeiten ein. Abschnitt 3 beschreibt die technische Implementation von mwcollect und ein konkretes Beispiel wird in Abschnitt 4 erläutert. Auf die Resultate wird in Abschnitt 5 eingegangen, bevor der Artikel dann mit einem Überblick über zukünftige Arbeiten und einer Zusammenfassung schließt.

2 Sammeln von Malware

Die Idee von mwcollect wurde bereits im vorhergehende Abschnitt kurz erläutert: Durch Emulation der notwendigen Funktionen eines Dienstes können wir eine effektive und effiziente Emulation eines Windowsrechners erzeugen, mit deren Hilfe dann automatisiert Malware gesammelt werden kann. In diesem Abschnitt werden wir dann auf die einzelnen Bausteine von mwcollect eingehen, bevor wir im nächsten Abschnitt die technische Realisierung vorstellen.

mwcollect simuliert auf rudimentärster Ebene die unter weiter Verbreitung ausgenutzten Schwachstellen für die Windows Services auf den TCP Ports 135, 445 und 1025. Spezifisch werden derzeit die Schwachstellen unterstützt, die in den folgenden Microsoft Security Bulletins dokumentiert sind:

- MS03-26 (“Buffer Overrun In RPC Interface Could Allow Code Execution”)
- MS04-11 (“Lsasrv.dll RPC buffer overflow remote exploit”)
- MS05-39 (“Vulnerability in Plug and Play Could Allow Remote Code Execution and Elevation of Privilege”)
- MS05-51 (“Vulnerabilities in MSDTC and COM+ Could Allow Remote Code Execution”)

Die Emulation der Schwachstellen funktioniert nach dem folgenden Prinzip: Zunächst werden die entsprechenden Ports geöffnet und eingehende Verbindungen werden auf spezifische,

für die Exploits charakteristische, Pattern untersucht. Falls für eine erfolgreiche Ausnutzung der Schwachstelle bestimmte Information zum Angreifer geschickt werden müssen (beispielsweise ein Verbindungsaufbau auf Anwendungsebene), so antwortet mwcollect mit einfachen Antworten, die größtenteils aus zufälligen Byte-Sequenzen und nur an den kritischen Offsets aussagekräftigen Daten bestehen. Wird ein Exploit erfolgreich erkannt, wird mittels der bekannten Pattern die Payload – der eigentliche Shellcode – extrahiert und analysiert. Die daraus erhaltenen Informationen werden genutzt, um ein Binary der Malware von einem infizierten Host herunterzuladen.

Der entscheidende Vorteil eines solchen Honeypots gegenüber einem klassischen high-interaction Honeypot zum Sammeln von Malware ist die wesentlich höhere Stabilität und Performance. Ein klassischer, sich in einer virtuellen Maschine befindender Honeypot, kann leicht durch einen Exploit mit einem falschem Offset zu einem Neustart gezwungen werden. Da mwcollect die Exploits lediglich abstrakt betrachtet, besteht hier dieses Risiko nicht. mwcollect kann weiterhin nativ ausgeführt werden und benötigt keine Prozessorvirtualisierung wie eine virtuelle Maschine. Es ist aufgrund des Designs schlicht unmöglich für die Malware, aus mwcollect “auszubrechen” und ausgeführt zu werden. Mwcollect läuft unter Linux und weder Malware noch Shellcode werden jemals ausgeführt. Ein mwcollect Host kann also nicht durch klassische Malware kompromittiert werden. Desweiteren hat ein spezialisierter Honeypot wie mwcollect noch andere Vorteile gegenüber einem nativen Honeypot:

- Die Skalierbarkeit ist besser und es können viele IP-Adressen parallel zum Sammeln von Malware benutzt werden. In unseren Tests haben wir in einem /18 Netzwerk mit mehr als 16.000 IP-Adressen keine Performance-Probleme.
- Ein zentrales Speichern von Malware wird einfach ermöglicht. Momentan ist eine zentrale Datenbank im Aufbau.

Der offensichtliche Nachteil von mwcollect ist der reaktive Ansatz: 0day Exploits, also neue Schwachstellen, für die es noch keinen Patch gibt, können nur erkannt werden, wenn sie durch die bereits bestehenden Pattern erkennbar sind, was höchst unwahrscheinlich ist. Für jeden neuen Exploit muss ein neues Modul entwickelt oder ein bestehendes erweitert werden. Jedoch verbreitet sich ein Großteil der sich heute im Umlauf befindlichen Malware über seit Monaten bekannte Schwachstellen. Die meisten Angreifer bauen eher auf einen Vorteil mangels Wissen und Beachtung der End-User als durch die neuesten Attacken. Deshalb greifen sie gerne auf als stabil geltende Exploits wie beispielsweise den LSASRV Exploit (dokumentiert in der Microsoft Security Bulletin MS04-11) zurück. Dieser Exploits ist immer noch einer derjenigen, der unseren Beobachtungen nach am häufigsten benutzt wird.

2.1 Ähnliche Projekte

Es gibt noch zwei Projekte, die einen ähnlichen Ansatz wie mwcollect verfolgen:

- nepenthes (`http://nepenthes.it`) ist eine Realisierung der gleichen Idee. Beide Projekte sind sehr ähnlich. Momentan verfügt nepenthes über mehr Schwachstellenmodule

und bietet einige interessante Möglichkeiten wie beispielsweise die geographische Visualisierung der IP von Angreifern. Allerdings ist kein zentraler Server zum Sammeln von Daten oder eine generische Shellcode Analyse geplant.

- Multipot (`http://labs.idefense.com`) ist eine Windowsapplikation, die ebenfalls ähnliche Ideen wie mwcollect implementiert. Allerdings ist die Schwachstellenemulation nicht sehr fortschrittlich und durch die Fokussierung auf die GUI kann keine gute Skalierbarkeit erreicht werden.

3 Implementation von mwcollect

Um seine Aufgabe effektiv zu erledigen und den Code wartbar zu halten, ist mwcollect in der Version 3.x in eine Vielzahl unterschiedlicher Module und einen Core aufgeteilt, die allesamt als C++ Shared Libraries von dem in C++ gehaltenem Core Daemon geladen werden. Der Core übernimmt dabei die Funktion der Koordination sowie der Netzwerkinteraktion und der Inter-Modul-Kommunikation. Die Schnittstelle zur Netzwerk API selbst ist wiederum in ein Modul gekapselt, dass die entsprechenden Funktionsaufrufe weiterleitet. Der Core assoziiert dann eingehende Verbindungen und Daten mit den entsprechenden Modulen, und so können auch mehrere Module auf dem gleichen Port lauschen und auf Aktivierung aufgrund dem Entdecken der charakteristischen Pattern warten. Dazu kann jedes Modul nach dem Eintreffen von Daten entscheiden, ob es weiter über Verkehr auf dieser Verbindung informiert werden will und ob andere Module weiter informiert werden sollen.

Insgesamt stehen die folgenden Module momentan zur Verfügung:

- Module zur *Emulation von Schwachstellen* emulieren auf bestimmten TCP Ports Schwachstellen, die häufig automatisiert ausgenutzt werden.
- Module zur *Shellcode Analyse* untersuchen die gefangene Payload der Exploits und versuchen, möglichst viele Informationen über den Nachlademechanismus herauszufinden.
- *Download Module* laden die eigentlich Malware mit Hilfe der durch die Shellcode Analyse Module gewonnenen Informationen herunter.
- *Submission Module* sorgen dafür, dass die heruntergeladene Malware abgespeichert wird, beispielsweise auf dem lokalen Filesystem oder einem entfernten Server.

Das Zusammenspiel der einzelnen Module ist in Abbildung 1 dargestellt und wird im Folgenden näher erläutert.

Kommunizieren können die Module untereinander auf zwei Ebenen, die beide über einen zentralen Event-Manager im Core abgebildet werden. Einerseits liefert der Core mehrere so genannte *Dispatcher*. Dies sind Pools, in die sich ein Modul eintragen kann, um über ausstehende Aufgaben informiert zu werden. Ein solcher Dispatcher steht jeweils für zu analysierende Shellcode, herunterzuladende URLs und zu speichernde Binaries bereit. Zusätzlich zu den

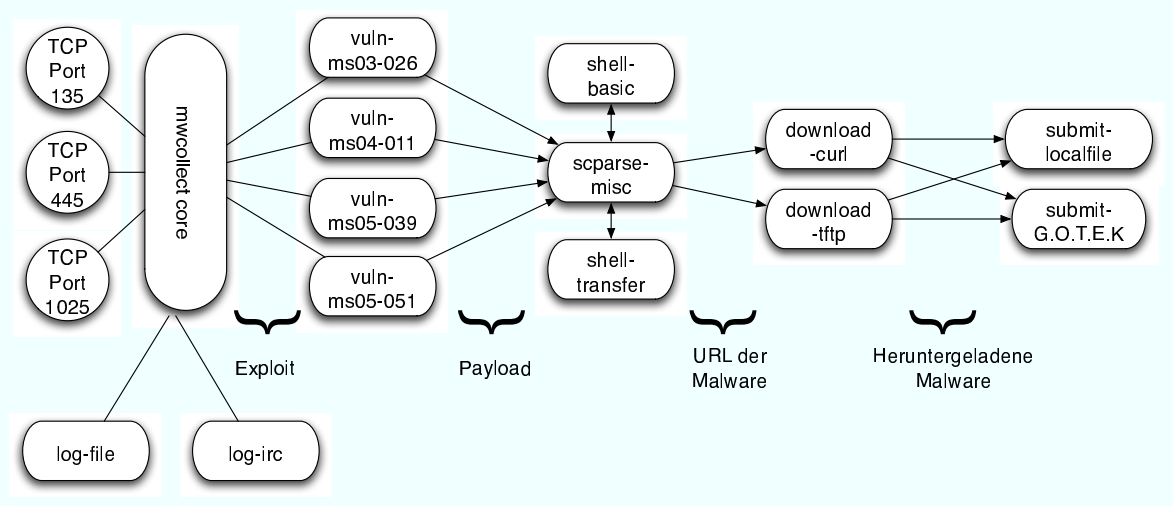

Abbildung 1: Schematischer Überblick über mwcollect

Dispatchern bietet der *Event Manager* den Modulen die Möglichkeit, selber Events bereitzustellen oder zu abonnieren, wodurch prinzipiell jedes Modul mit jeder Komponente jede Art von Daten austauschen kann.

Wie oben bereits erläutert, implementieren die *Module zur Schwachstellenemulation* eine hinreichende Abstraktion von verwundbaren Netzwerkdiensten. Als ein Beispiel wollen wir kurz die Funktionsweise der LSASS Emulation erläutern. Typischerweise wird TCP Port 445 von Windows 2000/XP genutzt, um das SMB (Server Message Block) Protokoll direkt über TCP/IP zu versenden. In der Microsoft Security Bulletin MS04-011 wurde eine kritische Schwachstelle zu diesem Dienst bekanntgegeben. Kurze Zeit nach dem Advisory wurde schon der erste Exploit veröffentlicht und der bekannteste Exploit für diesen Dienst wurde von houseofdabus als `HOD-ms04011-lsasrv-expl.c` erstellt. Prinzipiell läuft der Exploit in mehreren Stufen ab, in denen zunächst protokollspezifische Informationen ausgetauscht werden. In jeder diesen Stufen sendet der Exploit zunächst einen bestimmten Request und wartet dann auf eine Antwort, die allerdings nicht interpretiert wird, wie der Code-Ausschnitt im folgenden Diagramm zeigt:

```
if (send(sockfd, req1, sizeof(req1)-1, 0) == -1) {
        printf("[-] Send failed\n");
        exit(1);
}

len = recv(sockfd, recvbuf, 1600, 0);

if (send(sockfd, req2, sizeof(req2)-1, 0) == -1) {
        printf("[-] Send failed\n");
        exit(1);
}

len = recv(sockfd, recvbuf, 1600, 0);
```

Nachdem die ersten sechs Stufen durchlaufen sind, sendet der Exploit seine eigentliche Payload. Um an diese Payload zu gelangen, müssen also jeweils nur die ersten sechs Anfragen mit beliebigen Daten beantwortet werden, um so die nächste Stufe zu triggern. Als Code sieht das dann folgendmaßen aus:

```
case RPCS_GOT_LSASS_STAGE3:
case RPCS_GOT_LSASS_STAGE4:
case RPCS_GOT_LSASS_STAGE5:
  {
    unsigned char szBuffer[256];

    for(unsigned int i = 0; i < sizeof(szBuffer); ++i)
      szBuffer[i] = rand() % 0xFF;

    m_pCollector->getNetworkInterface()->sendData(iHandle,
                                szBuffer, sizeof(szBuffer));
    m_dsState = (rpc_state_t) ((unsigned int) m_dsState + 1);
  }
```

Das Vorgehen eines Moduls zur Schwachstellenemulation ist also relativ einfach: Der jeweilige Dienst wird nur insoweit emuliert, wie der Exploit es erwartet. Die einzelnen Stages des Exploits werden also solange getriggert, bis der Exploit seine eigentliche Payload sendet.

Die *Shellcode Analyse Module* arbeiten derzeit auf der Basis von Perl Compatible Regular Expressions"(PCRE) (`http://www.pcre.org`) Pattern. Auch hier wird also reaktiv vorgegangen, wobei die verwendeten Pattern so generisch wie möglich gehalten werden: Shellcodes werden bei der manuellen Analyse und Pattern-Erstellung in Decoder und eigentliche Payload separiert, so dass mit den Pattern beliebige Kombinationen bekannter Decoder und Payloads erkannt werden können. Weiterhin werden spezielle Bytesequenzen durch generische PCRE Bestandteile substituiert und damit Pattern erstellt, die unabhängig von den Assembler Spezifitäten wie beispielsweise verwendeten Registern sind.

Ein wesentlich generischerer Ansatz wird derzeit von Ryan Smith und anderen entwickelt. Dabei werden die Shellcodes in einer von qemu (`http://fabrice.bellard.free.fr/qemu/`) abgeleiteten virtuellen Maschine ausgeführt, jedoch derart isoliert und virtualisiert, dass sie für das Host-System nicht schädlich sein können. Aus dem Laufzeitverhalten der Shellcodes kann dann automatisiert und während der Laufzeit geschlossen werden, wie das entsprechende Malware Binary herunterzuladen ist.

Die Shellcode Analyse Module setzen dann für jedes Binary eine charakteristische URL zusammen, die über den *Download Dispatcher* an ein passendes *Download Modul* weitergeleitet werden. Dabei sind die URLs keinesfalls auf die gängigen Protokolle `http(s)://` und `ftp://` beschränkt, sondern es werden auch Protokolle wie beispielsweise `tftp://` unterstützt. Letztendlich ist der Protokollteil der URL nur wichtig, um das passende Download Modul zu identifizieren. Der Rest der URL ist charakteristisch für das jeweilige Protokoll und muss nicht zwingend der Form `$(host)/$(path)` entsprechen.

Wurde ein Binary erfolgreich heruntergeladen, so wird es an den *Submission Dispatcher* weitergereicht, der alle bei ihm registrierten Module über das neue Sample informiert. Derzeit

existieren zwei *Submission Module*. Eins speichert die Samples lediglich im lokalem Filesystem und dient damit in Verbindung mit den anderen Modulen als einfacher Downloadmanager von Malware auf dem lokalen Rechner. Die gespeicherten Samples werden jedoch nie ausgeführt, was in der UNIX/Linux Umgebung ohne Emulator unmöglich ist. Somit besteht keine Gefahr, durch die gesammelte Malware das Computersystem, auf dem mwcollect ausgeführt wird, zu infizieren oder zu beschädigen.

Das andere Submission Modul überträgt die gesammelten Samples über eine TCP Netzwerkverbindung an einen *G.O.T.E.K. Server*. Dieser ermöglicht das zentralisierte Sammeln der Binaries unter Verhinderung von überflüssigem Netzwerkverkehr mittels SHA-512 Hashing der Binaries. Weiterhin können mit den Samples *Prelude Correlation Alerts* assoziiert werden, was es unter zusätzlicher Verwendung des Prelude Logging Moduls erlaubt, neben dem eigentlichem Binary bereits vor der manuellen Analyse detaillierte Informationen über Herkunft und Verbreitungsmechanismen der Malware zu verfügen.

G.O.T.E.K. wurde von der mwcollect Alliance (`https://alliance.mwcollect.org/`) entwickelt, einer freie non-profit Organisation, die AV-Herstellern, anderen Entwicklern und Forschern aktuelle Malware Binaries kostenfrei zur Verfügung stellt. Diese werden mit überall auf der Welt verteilten mwcollect Sensoren gesammelt und auf einem zentralen Server gespeichert. Um einen Missbrauch der gesammelten Malware zu verhindern, ist eine vorherige Registrierung und Identifikation nötig. Beitreten kann der mwcollect Alliance grundsätzlich jede per Kopie des Personalausweises identifizierbare Privatperson oder über entsprechende Dokumente identifizierbare Institution. Die einzige Voraussetzung ist, dass der Betreibende mwcollect Sensoren bereitstellt, die die Alliance regelmäßig mit neuen Samples versorgen.

Neben den Submission Modulen gibt es – wie bereits oben erläutert – Module für das Simulieren der Schwachstellen, das Interpretieren der Shellcodes, das Logging, das Herunterladen der Binaries und als Schnittstelle zur Netzwerk API. Das Kapseln der eigentlichen Netzwerkkommunikation in eigene Module ermöglicht theoretisch das Integrieren von mwcollect in beliebige Umgebungen und in vielen Szenarien. So ließe sich beispielsweise ohne größeren Aufwand hinsichtlich der Schnittstelle zu mwcollect ein Modul schreiben, dass nachträglich tcpdump Capture Files einliest und wieder abspielt, wodurch sich eine forensische Analyse erleichtern lassen könnte. Derzeit ist das einzige Netzwerkmodul allerdings lediglich ein Wrapper Modul für die Berkley Socket UNIX Calls.

Durch das Modularisieren der *Logging Module* lassen sich einfach über den *Log Manager* des Cores Nachrichten ausgeben, dennoch könne diese dann vielfältig wiedergegeben werden. So unterstützt mwcollect neben Logging in Dateien und auf die Konsole per log-file Module auch mittels des log-syslog Module Logging an einen POSIX 1003.1-2001 kompatiblen Syslog Daemon und über log-irc das Logging in ein beliebiges IRC Netzwerk. Letzteres erleichtert das Beobachten mehrerer Sensoren in Echtzeit erheblich. Eine Livedemo von mwcollect ist auf dem IRC-Server `irc.freenode.org` im Channel `#mwcollect-demo` verfügbar.

4 Anwendungsbeispiel

Betrachten wir also als konkretes Fallbeispiel folgendes Szenario: eine rxBot will einen unseren Sensoren, den er für einen normalen Windows XP SP1 Host hält, mit sich selbst infizieren.

Dazu versucht er die Schwachstelle MS04-11 in lsass.exe auszunutzen. Der in den rxBot integrierte Public Exploit verbindet sich dazu zunächst auf TCP Port 445 unseres Sensor und sendet den ersten Teil seiner Payload (einen SMB Bind Request). Unser Sensor antwortet darauf mit einigen zufälligen Bytes, da der Public Exploit die empfangenen Daten nicht näher überprüft. Nach dem Austausch einiger weiterer Datenpakete, in denen mwcollect unter anderem dann seine Windows Version als XP angibt, wird die Payload des Exploits (der eigentliche Shellcode) übertragen.

Da der Autor dieser rxBot Variante kein besonders guter Programmierer, sondern vermutlich ein Scriptkiddie ist, das den Sourcecode des Public Exploits einfach übernommen hat, handelt es sich hierbei um einen einfachen Reverse Shell Shellcode. Dies bedeutet, dass der kompromittierte Host (in diesem Fall unsere Honeypot) sich zu der angreifenden Maschine verbindet und dann von dort weitere Befehle erhält. Der Shellcode wird von dem Vulnerability Module dann über den Dispatcher an das zuständige Shellcode Parsing Module ausgehändigt. Dieses bestimmt anhand einer der PCRE Pattern auch den Host und Port, zu dem sich der mwcollect Sensor verbunden werden soll und reicht diese Informationen an die Shell Simulation weiter, die sich prompt verbindet.

Sobald die Verbindung hergestellt wurde und die Shell Simulation den Shell Banner übertragen hat, sendet der rxBot einige `echo` Kommandos, die eine ftp.exe Befehlsdatei auf der Festplatte erstellen sollen. Dies Kommandos werden dann von der Shell Emulation in einem virtualisiertem Dateisystem interpretiert und somit kann die ftp.exe Simulation genau nachvollziehen, von welchem Server welche Datei heruntergeladen wird. Zu guter letzt versucht der rxBot, Windows ftp.exe direkt aufzurufen, was den Start der ftp.exe Simulation auslöst. Diese wiederum wandelt die Befehlsdatei in eine oder mehrere URLs um und reicht diese über den Download Dispatcher an das libCURL Download Module, dass HTTP und FTP URLs herunter lädt, weiter und startet somit den Downloadprozeß.

Wurde die Malware in einem separatem Thread vollständig heruntergeladen, wird das Binary über den Submission Dispatcher an alle Submission Module ausgehändigt. Das Modul submit-localfile speichert daraufhin die Malware als Datei im lokalem Filesystem, während das submit-gotek das neue Sample an die mwcollect Alliance oder einen anderen zentralen Sammelserver versendet.

5 Resultate

Der grundsätzliche Ansatz von mwcollect hat sich bewährt. So konnte beispielsweise mwcollect als erstes den Bot Dasher.B fangen. Dasher benutzt eine Schwachstelle von Windows auf TCP Port 1025, die in der Microsoft Security Bulletin MS05-051 dokumentiert ist. Da der zugehörige Exploit schon als Proof-of-Concept veröffentlicht war, konnten sehr schnell ein zugehöriges Modul für mwcollect erstellt werden. Mit Hilfe dieses Moduls konnten dann schnell der bösartige Netzwerkverkehr aufgezeichnet und ein Sample der Malware gefangen werden. Dies führte beispielsweise dazu, den zentralen Server, über den sich der Bot nachlädt, zu schließen und somit den Bot zu stoppen.

Innerhalb von etwa zwei Wochen konnten wir mit nur einem mwcollect Sensor 130 einzigartige Dateien sammeln. Einzigartigkeit wird in diesem Zusammenhang definiert als unterschied-

	Antivirussoftware 1	Antivirussoftware 2
Anzahl infizierter Dateien	46	129
Anzahl nicht entdeckter Dateien	84	1
Erkennungsrate	35,4%	99,2%

Tabelle 1: Erkennungsraten verschiedener Antivirussoftware

liche md5sum. Haben zwei gesammelte Binaries also die gleiche md5sum, so gelten sie als nicht unterschiedlich. Die gesammelten Binaries erlauben es uns, einen qualitative und quantitative Aussage über die Erkennungsraten von Antivirussoftware zu erstellen. Da wir davon ausgehen können, dass alle gesammelten Dateien bösartig sind (alle versuchten sich autonom weiterzuverbreiten), müsste eine perfekte Antivirussoftware also alle 130 Dateien als bösartig markieren. Tabelle 5 zeigt die Ergebnisse von zwei getesteten Programmen.

Wir sehen einen deutlichen Unterschied zwischen diesen beiden Antivirenprogrammen. Das Erste erkennt nur sehr wenige Samples und bietet deshalb nur einen sehr beschränkten Schutz. Das zweite hingegen bietet einen sehr guten Schutz, allerdings erkennt es auch nicht alle Samples. Dies unterstützt die These, dass signaturbasierte Antivirussoftware keinen zuverlässigen Schutz bietet. Ein Angreifer muss nur so lange seine Tools verändern, bis sie nicht mehr erkannt werden, und dann das Netzwerk angreifen.

Tabelle 5 listet die verschiedenen Bots auf, die mit Hilfe von mwcollect in diesen beiden Wochen gefangen wurden. Die meisten gefangenen Binaries sind verschiedene Varianten des Rbots. Dies ist nicht verwunderlich, da der Sourcecode von Rbot schon seit einiger Zeit frei im Internet verfügbar ist und Angreifer ihn einfach verändern können. Typischerweise fügen sie neue Exploits hinzu oder ändern den Sourcecode leicht ab. Durch die Verwendung von verschiedenen Compilern und Packern können sie häufig die Antivirussoftware umgehen und dann doch Schaden anrichten.

6 Zusammenfassung und Ausblick

Da sich mwcollect immer noch in Weiterentwicklung befindet, ist der derzeitige technische Standard keinesfalls als vollendet anzusehen. Neben dem offensichtlich notwendigem Hinzufügen von weiteren Shellcode Handlern und Schwachstellen Simulationen ist für die Zukunft ein generisches Python Interface zum einfacheren Erweitern sowie ein Modul für die Integration als Intrusion Detection System (IDS) geplant. Ein mwcollect Sensor hat naturgemäß eine definitionsbedingte Nullrate an False Positives – jede erfolgreiche Infektion ist ein Anzeichen für einen bösartigen Rechner innerhalb des Netzwerks. Somit kann diese Art von Honeypots vermutlich auch als IDS betrieben werden.

Insgesamt ist der Ansatz von mwcollect effektiv und erlaubt ein automatisiertes Sammeln von Malware. Im nächsten Schritt muss nun ein Weg gefunden werden, diese Malware auch effektiv und automatisiert zu analysieren. Dies ist ein schwieriges Problem, allerdings kann dynamische Programmanalyse vermutlich helfen, wenigstens grob die Funktionsweise der gesammelten Binaries zu verstehen.

Name der Malware	Anzahl
W32/Rbot-Fam	82
W32/Rbot-AMB	14
Exp/MS04011-A	5
W32/Rbot-BAH	2
W32/Rbot-ARS	2
W32/Korgo-T	2
W32/Codbot-AA	2
W32/Tpbot-A	1
W32/Rbot-CUG	1
W32/Rbot-BFR	1
W32/Rbot-BAP	1
W32/Rbot-AXQ	1
W32/Rbot-AOK	1
W32/Rbot-AAZ	1
W32/ParaDrop-B	1
W32/Korgo-U	1
W32/Korgo-S	1
W32/Francette-Y	1
W32/Doxpar-C	1
W32/Dopbot-B	1
W32/Codbot-Y	1
W32/Codbot-X	1
W32/Codbot-W	1
W32/Codbot-P	1
W32/Codbot-AC	1
W32/Codbot-AB	1
Troj/DownLdr-IMM	1

Tabelle 2: Vielfalt der mit mwcollect gefangenen Binaries

Literatur

[Hol05] Thorsten Holz. A short visit to the bot zoo. *IEEE Security & Privacy*, 3(3):76–79, 2005.

[McC03a] Bill McCarty. Automated identity theft. *IEEE Security & Privacy*, 1(5):89–92, 2003.

[McC03b] Bill McCarty. Botnets: Big and bigger. *IEEE Security & Privacy*, 1(4):87–90, 2003.

[MPS+03] David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, and Nicholas Weaver. Inside the slammer worm. *IEEE Security and Privacy*, 1(4):33–39, July–August 2003.

[The05a] The Honeynet Project. Know Your Enemy: Phishing, May 2005. `http://www.honeynet.org/papers/phishing/`.

[The05b] The Honeynet Project. Know Your Enemy: Tracking Botnets, March 2005. `http://www.honeynet.org/papers/bots/`.