Stormbot - Ein Botnetzwerk im Detail

Steve Walter
Betreuer: Matthias Wachs
Seminar Innovative Internet-Technologien und Mobilkommunikation WS2010/11
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: walteste@in.tum.de

ABSTRACT

In dieser Ausarbeitung wird das Peer-to-Peer-Bot-Netzwerk Stormnet basierend auf dem Stormbot detailiert behandelt. Es werden allgemeine Eigenschaften des Bots herausgearbeitet und analysiert. Die geschichtliche Entwicklung von Bots wird erläutert und Stormbot in einen zeitlichen Rahmen eingeordnet. Zum besseren Verständnis der Bot-internen Abläufe wird auf das Kademlia Peer-to-Peer-Protokoll eingegangen, auf dem Stormbot basiert. Auf dieser Grundlage wird die Art und Weise der Verbreitung und die Kommunikation innerhalb des Botnetzwerkes genau beschrieben. Darauf basierend werden die Schwachstellen der derzeitigen Implementierung aufgezeigt und auf mögliche Verbesserungen hingewiesen. Des Weiteren wird auf Methoden eingegangen, mit denen das Netzwerk übernommen und zerschlagen werden kann, sowie auf die Möglichkeiten die zur Verfügung stehen dies zu unterbinden.

Keywords

Stormbot, Peer-to-Peer-Netzwerke, Kademlia, Sybil-Angriff

1. EINLEITUNG

Ein Bot ist die Software, welche auf einen einzelnen Rechner installiert ist und diesen Rechner fernsteuert. Kommunizieren und organisieren sich mehrere Bots untereinander und führen somit bestimmte Aufgaben aus, spricht man von einem Botnetzwerk. Dabei bekommen die einzelnen Bots über unterschiedliche Verfahren Befehle vom Botnetzbetreiber mitgeteilt. Botnetzwerke stellen eine zunehmende Bedrohung für die Sicherheit im Internet dar. Durch eine Masse ferngesteuerter Rechner ist es möglich eine Vielzahl von Angriffen durchzuführen, unbefugte Daten zu sammeln oder Spam zu versenden. Da es sich dabei in der Regel um illegale Aktivitäten handelt und hauptsächlich privaten Rechner mit Bots infiziert werden, geschieht dies meist ohne das Wissen des jeweiligen Nutzers. Der Großteil der bestehenden Botnetzwerke arbeiten zentralisiert indem sich die einzelnen Bots zu einem bestimmten Server verbinden von dem sie ihre Aufträge bekommen. Diese Botnetzwerke können durch das Aufspüren und Abschalten des Zentralservers unschädlich gemacht werden. Eine wesentlich robustere Variante ist ein dezentralisiertes Botnetzwerk, bei dem kein zentraler Server existiert, sondern die Kommunikation zwischen den Bots basierend auf einem Peer-to-Peer-Netzwerk stattfindet. Bots die auf einer solchen Technologie basieren nennt man Peer-to-Peer-Bots (P2P-Bots). Im Zuge dieser Arbeit wird Stormbot behandelt, einer der ersten P2P-Bots der zwischen 2006 und 2007 aktiv war.

Im Ersten Abschnitt dieser Arbeit werden allgemeine Eigenschaften des Stormbots behandelt. Darunter fällt die globale Verbreitung sowie die Ziele und Möglichkeiten des Botnetzwerks. Im zweiten Abschnitt wird die Geschichte von Botnetzwerken, begonnen beim ersten IRC-Bot bis hin zum Stormbot, kurz beschrieben. Es folgt ein Abschnitt über das Kademlia-Protokoll, auf dessen Basis Stormbot konstruiert ist. Unter Anderem wird auf den Beitritt in das Netzwerk sowie die Suche innerhalb des Netzwerks eingegangen. Der folgende zentrale Abschnitt dieser Arbeit beschäftigt sich mit Stormbot im Speziellen. Es wird auf die Verbreitung des Bots, die Infektion des Rechners sowie die Befehlsstruktur des Botnetzwerks näher eingegangen. In den letzten Abschnitten wird auf die Schwachstellen des Netzwerkes hingewiesen und ein erfolgreicher Angriff, sowie entsprechende Gegenmaßnahmen geschildert. Am Ende befindet sich eine kurze Zusammenfassung der zentralen Eigenschaften des Stormbots und die finalen Erkenntnisse die aus dem Botnetzwerk gewonnen werden können.

2. ALLGEMEINE INFORMATIONEN

Stormbot verbreitet sich von Rechner zu Rechner in Form eines Trojaners, der den offiziellen Namen “Trojan.Peacomm” trägt. Er befällt ausschließlich die Betriebssysteme Windows 95, 98, ME, 2000, NT und XP. Die Größe des gesamten Botnetzwerks wird auf minimal 5.000 bis 6.000 und maximal 45.000 bis 80.000 infizierte System geschätzt [2]. Infizierte Rechner befinden sich weltweit in mehr als 200 Ländern. Die USA ist mit 31% infizierten Maschinen am stärksten betroffen. Gefolgt von Russland 15% und Indien 9,2%. Deutschland fällt mit 5,1% ins Gewicht. Eine aktuelle Statistik über die globale Verbreitung des Strombots kann in [6] gefunden werden.

Stormbot wurde hauptsächlich mit dem Ziel entwickelt Spam-Mails zu versenden und “Distributed Denial of Service”-Attacken auszuführen. Dies hat zur Folge, dass das Netzwerk ebenso im Stande ist sich durch illegale Informationsbeschaffung, meist in Form von E-Mail-Adressen, zu vergrößern. Des Weiteren kann das Netzwerk in verschiedene Teilnetze gegliedert werden, sodass davon auszugehen ist, dass das Botnetzwerk in Teilen auch vermietet hätte werden können. Somit diente es auch der Kapitalbeschaffung des Betreiber. Es wird davon ausgegangen, dass die Betreiber des Botnetzes entweder durch die Vermietung des Botnetzes zum Versenden von Spam Geld machen oder selbst ein pharmazeutisches Unternehmen führen für dessen Medikamente per Botnetz Spam-Werbung versendet wird.

doi: 10.2313/NET-2011-05-2_12

Bekannte Ziele der DDoS-Attacken[1] des Stormbotnetzwerkes sind unter anderem Wirtschaftsseiten und Anti-Spam-Seiten. Des Weiteren wurden Seiten konkurrierender Botnetzwerke angegriffen um womöglich die Konkurrenz auszuschalten. Auch wurden Analysten die versuchten die ausführbare Datei, welche Stormbot enthält zu analysieren Opfer von DDoS-Attacken. Jedoch ist nicht bekannt, ob diese automatisch erfolgen, wenn der Bot bemerkt, dass er analysiert wird oder ob sie vom Betreiber ausgelöst wurden.

Die Kommunikation von Stormbot erfolgt über Overnet, welches eine konkrete Umsetzung von Kademlia darstellt. Kademlia ist ein Protokoll zur Kommunikation in Peer-to-Peer-Netzwerken, das auf verteilten Hashtabellen basiert. Overnet selbst wurde für die Verwendung von Filesharing-Programmen wie eDonkey2000 und BitTorrent entwickelt, welche nicht über einen zentralen Server kommunizieren, sondern Daten innerhalb des P2P-Netzes übertragen.

3. GESCHICHTLICHE EINORDNUNG

Die meisten zentralgesteuerten Bots basieren auf dem Internet Relay Chat (IRC), welches eine Chatsoftware ist, die leicht zu handhaben ist und sehr gut mit der Anzahl der Nutzer skaliert. Der erste Bot wurde im Dezember 1993 entwickelt und trug den Namen "EggDrop". Die ersten Bots waren ursprünglich dazu entwickelt die Handhabung von IRC weiter zu vereinfachen und zu automatisieren. Aus diesen Grunde hatten die zu dieser Zeit entwickelten Bots alle noch keinen schädlichen Charakter, sondern wirkten den Benutzer unterstützend. Erst in April im Jahr 1998 wurde der erste schädliche IRC-Bot mit dem Namen "Global Thread" in mehreren Varianten entwickelt. Dieser zeichnete sich durch eine modifizierte ausführbare Datei von IRC aus, welche in der Lage war Schadcode in Form von Scripten auszuführen. Im Mai 1999 wurde mit Napster der erste offizielle Peer-to-Peer-Filesharing-Service veröffentlicht. Napster benutzte allerdings einen zentralen Server um die Anfragen der einzelnen Peers zu koordinieren. Erst im März 2000 wurde mit Gnutella der erste P2P-Service veröffentlicht, der keine zentralen Server zur Koordination benötigte. Die Verbindungen in diesen Netzwerk wird von den Knoten selbst erstellt, sodass jeder Knoten bei seinen Nachbarknoten die Adressen anderer Knoten erfragen kann. Damit bekommt jeder Knoten eine Liste von möglichen Knoten bei denen er nach Informationen oder nach anderen Knoten suchen kann. November 2003 wurde dieses Konzept in Form von Kademlia auf verteilte Hashtabellen erweitert. Verteilte Hashtabellen bieten den Vorteil, dass die Suche im Vergleich zu normalen linearen Listen wesentlich effizienter stattfinden kann. Zuvor wurde im März 2003 mit WASTE ein VPN[2] ähnliches P2P-Netzwerk entwickelt, welches zur Verschlüsselung der Verbindung RSA[3] nutzte. WASTE enthält neben der Verschlüsselung noch weitere Funktionen unter anderem einen "Random Traffic Generator" und die Möglichkeit die Benutzung von WASTE zu verschleiern. Es bildet damit die erste Implementierung eines Peer-to-Peer-Netzwerkes welches starken Wert auf Sicherheit legt. Basierend auf WASTE entstand im März 2004 mit "Phatbot" der erste P2P-Bot. Dieser Bot zeichnete sich durch modulares Design aus, was zur Folge hat, dass er sich schnell an neue Sicherheitslücken anpassen kann. So verwendete er zum Beispiel die von Sasser genutzten Sicherheitslücken um sich zu verbreiten. Schließlich entstand am 29. Dezember 2006 die erste Variante des Stormbots.

[1]Distributed Denial of Service-Attacken sorgen dafür, dass ein bestimmter Service oder eine Infrastruktur aufgrund von Überlast nichtmehr verfügbar ist.

[2]Virtual Private Network dienen dazu kleinere, scheinbar unabhängige Netzwerke innerhalb eines großen Netzwerkes zu integrieren

[3]RSA ist ein asymmetrisches Verschlüsselungsverfahren zur Verschlüsselung von Daten

4. KADEMLIA-PROTOKOLL

Da Stormbot auf dem Kademlia-Protokoll basiert, sind die Kommunikationswege bis auf einige Ausnahmen gleich. Um die Funktionsweise von Stormbot zu verstehen wird deshalb ein Grundverständnis von Kademlia benötigt. Kademlia ist ein Service für P2P-Netzwerke, bei dem jeder Knoten der im Netzwerk teilnimmt sowohl Senden als auch Empfangen kann. Für die Kommunikation innerhalb des Kademlia-Netzwerkes wird UDP verwendet. Hauptmerkmal von Kademlia ist, dass die Knoten des Netzwerkes in verteilten Hashtabellen eingeordnet werden. Jeder Knoten besitzt dabei eine eindeutige "Distributed Hash Table ID" (DHT ID), welche durch einen 128-Bit MD4 Hashwert dargestellt wird. Dieser Hashwert wird beim ersten Eintritt in das Netzwerk automatisch generiert. Des Weiteren besitzt jeder Knoten eine Hashtabelle in der die Adressen anderer Knoten sowie deren Distanz gespeichert sind. Distanz meint in diesem Zusammenhang nicht die tatsächliche räumliche Distanz sondern die Distanz innerhalb des Netzwerks. Es kann zum Beispiel vorkommen, dass der direkte Nachbar eines infizierten Rechners in Deutschland ein infizierter Rechner in Australien ist. Die Distanz ergibt sich aus dem Modulo der beiden Hashwerte deren Distanz ermittelt werden soll. Ist zum Beispiel die DHT ID des Knoten A 1101 und die DHT ID des Knoten B 1001 so ergibt sich eine Distanz von $1101 \oplus 1001 = 0100$. Der Vorteil dieses Verfahrens und verteilter Hashtabellen besteht darin, dass die Suche innerhalb dieser Tabellen wesentlich schneller ist als bei linearen Listen. Dazu werden die DHT IDs in k-buckets sortiert, welche den Adressraum in entsprechende Teile spalten.In den k-buckets werden die Verbindungsdaten anderer Knoten gespeichert. Während der Kommunikation wird geprüft ob die in den k-buckets enthaltenen Verbindungsdaten aktuell sind und verfallene Verbindungsdaten werden entfernt. Dies garantiert, dass die Knoten nur eine Liste von aktiven Nachbarn verwalten und das Netzwerk nicht mittels einer DDoS-Attacke zerstört werden kann, bei welcher die Routingtabellen mit falschen Verbindungsdaten geflutet wird. Eine komplette Zusammenfassung über das Kademlia-Protokoll kann in [5] nachgelesen werden.

4.1 Bootstrapping

Um dem Netzwerk beizutreten, benutzt jeder Knoten eine als Bootstrapping bekannte Methode. Die Idee hinter diesem Verfahren ist, dass sich der Knoten von selbst in das Netzwerk einklinkt ohne auf einen speziellen Verbindungspunkt angewiesen zu sein, der ihn in das Netzwerk eingliedert. Dabei muss dem neuen Knoten, der sich mit dem Netzwerk verbinden will, allerdings ein Knoten der bereits Teil des Netzwerkes ist, bekannt sein. Dieser Einstiegsknoten kann jeder beliebige Knoten des Netzwerk sein und muss keine besonderen Eigenschaften erfüllen. Der neue Knoten verbindet sich zu diesem Einstiegsknoten und fordert eine Liste

doi: 10.2313/NET-2011-05-2_12

der dem Einstiegsknoten bekannten weiteren Netzwerkknoten an. Anhand dieser Liste kann der neue Knoten nun seine eigene Knotenliste erweitern und ist nun vom Einstiegsknoten unabhängig.

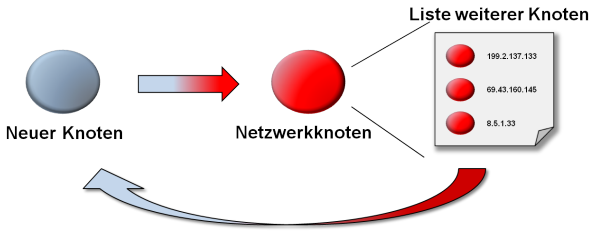

Figure 1: Bootstrapping

4.2 Suche innerhalb des Netzwerks

Um bestimmte Inhalte innerhalb eines DHT basierten Peer-to-Peer-Netzwerks zu finden wird zuerst der Hashwert des gesuchten Inhalts ermittelt. Im Anschluß werden die Knoten in der Hashtabelle mit der geringsten Distanz zum errechneten Hashwert befragt. Die befragten Knoten haben entweder die gesuchte Information, worauf dann eine direkte Verbindung zum Zielknoten aufgebaut werden kann oder sie berechnen ihrerseits die Knoten ihrer Hashtabelle mit der geringsten Distanz zum Hashwert des gesuchten Inhalts. Diese neu gefundenen Knoten werden dann zum suchenden Knoten übermittelt, welcher daraufhin diese Knoten nach den gesuchten Inhalten befragt. Besitzt der neu gefundene Knoten die gesuchten Informationen wird eine separate Verbindung zwischen den beiden Knoten aufgebaut und die Information übertragen. Andernfalls übersendet der neu gefundene Knoten ebenfalls einen Knoten aus seiner Hashtabelle der näher an dem vom suchenden Knoten benötigten Inhalten ist. Mit dieser Methode wird auf iterative Art und Weise der Knoten mit dem gesuchten Inhalt ermittelt.

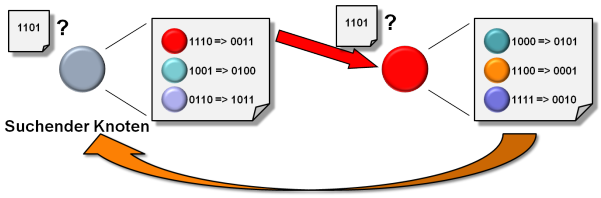

Figure 2: Suche nach dem Hashwert 1101

Ein verdeutlichendes Beispiel zeigt Figur 2. Der suchende, hellblaue Knoten will den Inhalt mit dem Hashwert 1101 erhalten. Dazu überprüft er die Distanz der Knoten seiner Hashtabelle. Die Distanz des roten Knoten zum suchendem Knoten beträgt 1110. Durch den Modulo-Operator wird die Distanz des roten Knoten zum gesuchten Inhalt ermittelt: $1110 \oplus 1101 = 0011$. Des weiteren ermittelt der suchende Knoten auch die Distanz der Anderen Knoten in seiner Hashtabelle und stellt fest, dass der rote Knoten näher an der gesuchten Information ist als alle anderen Knoten, die ihm bekannt sind. Folglich fragt der suchende Knoten den roten Knoten nach dem Inhalt mit den Hashwert 1101. Der rote Knoten wiederholt den Prozess der Distanzermittlung innerhalb seiner Hashtabelle und übersendet dem fragenden Knoten den orangen Knoten.

4.3 Nachrichten zwischen den Bots

Während des Kommunikationsprozess werden folgende Nachrichten verwendet:

hello wird verwendet um zu überprüfen ob ein Knoten existiert oder die umliegenden Knoten über die eigene Existenz zu informieren.

route request wird verwendet um Knoten die näher am gesuchten Hashwert sind von einem Nachbarknoten zu erfragen.

route response ist die Antwort auf einen route request und beinhaltet die empfohlenen Knoten die näher am gesuchten Hashwert sind.

publish request/response werden zum Veröffentlichen bestimmter Inhalte verwendet.

search request/response kommen zum Einsatz bei Suchen nach bestimmten Inhalten.

5. STORMBOT IM DETAIL

5.1 Verbreitung

Der Stormbot-Trojaner verbreitet sich ausschließlich über E-Mails mittels Social Engineering. Aus diesem Grund handeln die Inhalte der E-Mails meist von sozialen Ereignissen wie zum Beispiel Weihnachten, Neujahr oder Halloween. Des Weiteren gibt es Mails, die auf derzeitig aktuelle Ereignisse hinweisen, wie Tag der Arbeit, Beginn der Ferien oder der NFL Saison. So verdankt der Stormbot seinen Namen einer Verbreitungsmail in der vor dem Orkan Kyrill, der 2007 in der USA wütete, gewarnt wird. Auch sind Varianten im Umlauf in denen Spiele angeboten werden. In den Mails befindet sich neben einem Text entweder eine ausführbare Datei im Anhang die den Trojaner installiert oder ein Link von dem eine entsprechende Kopie des Trojaners heruntergeladen wird. Eine Liste der möglichen Anhänge kann in Tabelle 1 eingesehen werden. In Tabelle 2 sind einige Betreffzeilen mit denen Stormbot sich per Mail verbreitete, eingetragen. Es wurden keinerlei Sicherheitslücken im System ausgenutzt und die Infektion fand somit durch den Benutzer, wenn auch unter Vorspiegelung falscher Tatsachen, statt. Die Spam-Mails wurden hauptsächlich in der USA verbreitet. Mithilfe von *spamtraps* konnte die Anzahl der versendeten Spam-Mail, welche die ausführbare Datei des Stormbots enthielten festgestellt werden. Eine *spamtrap* ist ein für den Erhalt von Spam erstelltes E-Mail-Postfach anhand dessen analysiert werden kann, wie viel und welche Art Spam weltweit versendet wird. Vom September 2006 bis September 2007 wurden zwischen 2.200 und 23.900 Spam-Mails pro Tag vom Stormbotnetzwerk verschickt. Das entspricht 8.500 Spam-Mails im Durchschnitt und machte damit 10% aller von den *spamtraps* täglich gesammelten Spam-Mails aus [2].

FullVideo.exe	FullStory.exe	Video.exe
ReadMore.exe	FullClip.exe	GreetingPostCard.exe
MoreHere.exe	FlashPostCard.exe	GreetingCard.exe
ClickHere.exe	FullNews.exe	

Table 1: Namen der Anhänge der Stormbot-Spam-Mails

doi: 10.2313/NET-2011-05-2_12

Naked teens attack home director.
230 dead as storm batters Europe.
Radical Muslim drinking enemies's blood.
Chinese missile shot down Russian satellite.
Saddam Hussein alive!
Venezuelan leader: "Let's the War beginning".
Fidel Castro dead.

Table 2: Betreffzeilen der Stormbot-Spam-Mails

5.2 Infektion

Die Infektion erfolgt über 2 Phasen, die Erste und zweite Injektion genannt werden. Mit dem Ausführen des Anhangs wird die erste Injektion initiiert. In dieser Phase verankert sich der Bot im System. Dazu wird der Systemtreiber "wincom32.sys" im Windows-Stammverzeichnis erzeugt, welcher zur Verbindung mit dem Netzwerk benötigt wird. Dieser Treiber wird in die "service.exe" injiziert, wodurch dieser Service wie ein P2P-Client fungiert und die zweite Injektion herunterladen kann. Die Uhr des infizierten Rechner wird mit dem "Network Time Protocol" synchronisiert, da zur korrekten Kommunikation der Bots untereinander eine zeitliche Synchronisation vorausgesetzt wird (siehe Abschnitt 5.4). Die Synchronisation der Uhr mittels "Network Time Protocol" wird zwar von Windows XP ab Service Pack 2 automatisch vollzogen, jedoch ist dies auf den anderen Zielsystemen (Win 95/98/NT/2000/ME) nicht automatisch der Fall. Dies ist nötig um die Kommunikation zwischen den Bots zu gewährleisten. Des Weiteren werden der Windows Firewall und die Einstellungen der Internetsicherheit deaktiviert. Da es vorkommen kann, dass neben dem Windows Firewall noch weitere Firewalls auf dem infizierten Rechner installiert sind oder sich der Rechner hinter einem NAT[4] verbirgt, werden wie in [1] beschrieben eine Reihe von Ports zusätzlich geöffnet:

TCP:	139, 12474
UDP:	123, 137, 138, 1034, 1035, 7871, 8705, 19013, 40519

Nachdem eine DHT ID generiert wurde, tritt der neue Bot dem Netzwerk wie in 4.1 beschrieben bei. Im Stormbot-Netzwerk sucht der neue Bot nach einem bestimmten Hashwert, der ihm eine verschlüsselte URL zur zweiten Injektion übermittelt. Die URL wird entschlüsselt und die zweite Injektion heruntergeladen und installiert.

Bei der zweiten Injektion handelt es sich um sechs zusätzlicher Komponenten. Diese Komponenten sind optional und müssen nicht alle installiert werden. Sie tragen die Namen "game0.exe" bis "game5.exe". Die zusätzlichen Funktionalitäten sind wie folgt [7]:

- Erweiterung zum Herunterladen weiterer Module oder Befehle
- Rootkit um den Bot vor dem System zu verbergen
- SMTP- und Spam-Client zum versenden von Spam-Mails
- E-Mail-Adressensammler zur Verbreitung des Bots per Mail
- DDoS-Attacken-Komponenten

Nach dem Abschluss der zweiten Injektion beginnt der Bot das Netzwerk nach Befehlen zu durchsuchen.

Zu Forschungszwecken wurde die zweite Injektion von einer Gruppe von Analysten mehrfach von der vom Bot gefundenen URL heruntergeladen um festzustellen, ob sich die der Inhalt oder die Zusammensetzung der zweite Injektion verändert. Dabei stellte sich heraus, dass wenn man die zweite Injektion in einem bestimmten kurzen Zeitintervall (innerhalb weniger Minuten) mehrfach mit der gleichen IP-Addresse herunterläd, nach einem undefinierten Zeitintervall ein DDoS-Angriff auf diese IP-Addresse erfolgt. Dadurch gerieten die Analysten selbst ins Visir des Botnetzwerk. Es ist allerdings nicht bekannt, ob diese Angriffe automatisch erfolgten oder vom Betreiber des Botnetzwerk ausgingen, da die Zeitintervalle die zwischen dem Herunterladen und den DDoS-Angriffen lagen stark varierten und von 10 Minuten bis zu einem halben Tag reichten.

5.3 Verbindung zum Netzwerk

Um den Netzwerk beizutreten, benötigt es wie in im Abschnitt 4.1 beschrieben einen Einstiegsknoten. Zu diesem Zweck wird mit der ersten Injektion im Windows-Stammverzeichnis die Datei "%windir%\system\wincom32.ini" angelegt. Diese enthält eine Liste von über 140 Einstiegsknoten in das Netzwerk. Diese Liste ist in der ersten ausführbaren Datei des Trojaners hartkodiert. Es ist nicht bekannt, wie diese Liste zusammengestellt ist und ob sie unter Umständen aktualisiert wird. Sie hat den folgenden Aufbau:

```
<DHT ID>=<IP><Port>00
<DHT ID>=<IP><Port>00
...
```

Zum Beispiel:

```
D943283AB63746B8E62436682728DDD4=5511238154BD00
D6E46BF02E64D940E37EECCC982584A8=573349B6124A00
...
```

wobei auf der linke Teil die DHT ID darstellt und der rechte Teil die IP und den Port in Hexadezimal:

```
Knoten DHT ID: D943283AB63746B8E62436682728DDD4
0x55.0x11.0x23.0x81:0x54BD = 85.17.35.129:21693
```

Anhand dieser Einstiegsknoten verbindet sich der neue Bot zum Netzwerk. Nach dem ersten Verbinden hat dieser dann eine eigene Liste von Kontaktknoten von den entsprechenden Einstiegsknoten erhalten, sodass er danach von dieser Liste unabhängig ist.

[4]Network Address Translation verändern die Addressdaten in einem Datenpaket. Tritt zum Beispiel bei Routern auf.

doi: 10.2313/NET-2011-05-2_12

5.4 Steuerung des Botnetzes

Damit Befehle an die Bots übermittelt werden können, benötigt es bestimmte Kontrollknoten zu denen sich die einzelnen Bots verbinden und Befehle entgegen nehmen können. Diese Kontrollknoten werden anhand eines bestimmten Suchhashwerts ausfindig gemacht. Dieser wird durch einen Algorithmus berechnet, der in jedem Bot verankert ist. Der Suchhashwert errechnet sich aus dem aktuellem Datum und einen zufälligen Wert im Intervall von $[0 - 32]$. Aus diesem Grund muss während der ersten Injektion die Uhr des infizierten Systems synchronisiert werden. Somit sind täglich 32 unterschiedliche Schlüssel im Umlauf. Diese Schlüssel fungieren als Treffpunkte an denen sich die Bots und die die Kontrollknoten treffen beziehungsweise anhand deren sie sich finden können. Da der Algorithmus sowohl dem Bots als auch den Kontrollknoten bekannt ist, veröffentlichen die Kontrollknoten ihre Inhalte unter dem Hashwerten nach den die Bots suchen oder in naher Zukunft suchen werden. Nach der Berechnung des Hashwerts innerhalb eines Bots wird eine Suche mittels "routing request" gestartet an deren Ende der gesuchte Kontrollknoten gefunden wird. Ist die IP-Adresse und das Port schließlich bekannt, wird eine TCP/IP-Verbindung zwischen Bot und Kontrollknoten hergestellt. Es erfolgt eine kurze Authentifizierung bei welcher der Kontrollknoten eine zufällige Zahl mit einem in jedem Bot hartkodierten Schlüssel XOR-verknüpft und dem verbundenen Knoten übersendet. Dieser errechnet mittels des hartkodierten Schlüssels "`0x3ED9F146`" die vom Kontrollknoten generierte, zufällige Zahl und sendet sie dem Kontrollknoten als Antwort zurück. Damit ist die Authentifizierung abgeschlossen. Der Kontrollknoten übermittelt im Anschluß weitere Befehle wie zum Beispiel Angriffsdaten für deine DDoS-Angriff oder den Inhalt einer Spam-Mail, die verbreitet werden soll. Durch die Kontrollknoten und die Suche der normalen Botknoten nach deren verbreiteten Daten lässt sich das Botnetzwerk direkt steuern und kann so vom Netzwerkbetreiber entsprechend verwendet werden.

In den frühen Versionen des Stormbotnetzwerkes wurde Overnet zur Kommunikation verwendet, welches keinerlei Verschlüsselung unterstützt. Mit fortschreitender Entwicklung und zunehmender Größe des Netzwerkes wurde das Netzwerk erweitert und eine 40-Bit XOR-Verschlüsselung integriert. Dieser 40-Bit-Schlüssel ist in jeden Bot hartkodiert. Alle Nachrichten innerhalb des Netzwerkes werden in neueren Versionen mit diesen Schlüssel verschlüsselt. Die Funktionalität bleibt aber die selbe wie in Overnet/Kademlia.

6. SCHWACHSTELLEN UND ÜBERNAHME

6.1 Schwachstellen

Die gewählte Funktionsweise und der Aufbau des Bots zeigen einige Schwachstellen die unter Anderem dazu führten, dass das große Teile des Botnetzwerk übernommen und zerschlagen werden konnten. Eine der offensichtlichsten Schwachstellen ist die Liste mit den Einstiegsknoten insofern sie nicht aktualisiert wird oder veränderlich ist. Sollte es gelingen alle in dieser Liste angegebenen Startknoten zu lokalisieren und vom Netz zunehmen, findet ein neuer Bot keinen Zugang mehr zum bestehenden Botnetzwerk. Des Weiteren bietet die Suche des Kontrollknotens eine weitere Schwachstelle, da durch ein Abfangen und fehlleiten der Routing-Anfrage die Suche auf einen falschen oder gefälschten Knoten umgeleitet werden kann. Es fehlt weiterhin ein sicheres kryptographisches Verfahren, das die Kommunikation zwischen den Bots verschleiert.

Eine gravierende weitere Schwachstelle stellen die hartkodierten Daten und Algorithmen dar. So konnte durch Reverse Engineering der ausführbaren Datei der Algorithmus zur Berechnung des Suchhashwertes und damit des Treffpunktes des Stormbots gewonnen werden. Die Auswirkungen dieser Erkenntnis werden im folgendem Abschnitt 6.2 behandelt. Ebenso wurde der 40-Bit-Schlüssel für die XOR-Verschlüsselung gefunden und der Hashwert unter dem die verschlüsselte URL für die zweite Injektion angeboten wird. Dadurch ist die Kommunikation zwischen den einzelnen Knoten nicht mehr sicher und kann abgehört werden. Auch der Schlüssel der zur Authentifizierung des Bots beim Kontrollknoten verwendet wird, ist extrahiert wurden.

6.2 Sybil-Angriff

Große Teile des Strombotnetzwerk wurden mithilfe des Sybil-Angriffs erfolgreich infiltriert und übernommen [2, 3]. Dazu werden künstliche Bots in das Netzwerk eingespeist. Künstliche Bots sind Bots, die von den Angreifern des Botnetzwerkes entwickelt wurden um das Botnetzwerk zu infiltrieren. Alle anderen Bots, die von den Entwicklern des Botnetzwerkes entworfen wurden, werden als echte Bots bezeichnet. Diese künstlichen Bots verbreiten falsche Routingeinträge mittels Hello-Nachrichten im Netzwerk. Dazu wird den echten Botnetzknoten in den Hello-Nachrichten vorgetäuscht, dass sich die künstlichen Knoten näher zu den tatsächlichen Botnetzknoten befinden als die echten Botnetzknoten. Durch diesen Umstand übernimmt der echte Botnetzknoten den künstlichen in seine Routingtabelle und entfernt im Besten Fall die Einträge der echten Botnetzknoten. So werden die echten Botnetzknoten sämtliche Anfragen in erster Linie an die künstlichen Bots senden. Da der Algorithmus mit denen die Bots ihren nächsten Treffpunkt berechnen gefunden wurde, können die gesuchten Schlüssel vor berechnet werden. Die künstlichen Bots leiten dann die echten Bots auf einen künstlichen Kontrollknoten, statt auf den Kontrollknoten des Betreibers, um. Dadurch ist die Übernahme gelungen und die Bots können mithilfe des künstlichen Kontrollknotens unschädlich gemacht oder manipuliert werden. Der Vorteil dieser Methode ist neben einer Übernahme des Netzwerk auch die Effizienz. Es werden keine Unmengen an Rechner benötigt um die Vielzahl an künstlichen Bots bereit zustellen. Sondern Dank der geringen Daten, die ein einzelner Knoten benötigt, ist es möglich an einen einzelnen Rechner das Netzwerk mit hunderten von künstlichen Bots zu überschwemmen.

Den Sybil-Angriff kann allerdings durch entsprechende Vorkehrungen verhindert werden[4]. Die sicherste Methode besteht darin eine vertrauenswürdige Autorität in das Netzwerk zu integrieren, die sicher stellt, dass es sich bei allen Teilnehmern des Netzwerkes um vollwertige Mitglieder handelt. Nicht validierte Knoten hätten somit keine Chance das Netzwerk zu zerschlagen. Ein weitere Methode wäre die Knoten einen Ressourcen-Test zu unterziehen. Dazu wird geprüft ob die Knoten die entsprechenden Ressourcen eines normalen Rechners aufweist und zum Beispiel die Rechenleistung oder die Speicherkapazität getestet. Zum Beispiel könnte dem Bot, der sich in das Netzwerk einklinken will, eine sehr rechenintensive Aufgabe gestellt werden, welche

doi: 10.2313/NET-2011-05-2_12

den Prozessor des Rechners auf dem der Bot läuft stark beansprucht. Der Bot müßte diese Aufgabe in einer vertretbaren Zeit lösen können. Wird nun versucht der Sybil-Angriff durch die Simulation mehrer Botknoten auf einen Rechner anzuwenden, wird der Prozesser des Rechners überlastet und die Aufgaben, die jede Simulation eines Bots gestellt bekommt, könnten nicht in vertretbarer Zeit gelöst werden. Die Variante des Sybil-Angriffs die auf einen Rechner ausgeführt werden kann, würde damit enttarnt werden, da ein Rechner nicht die Rechen- oder Speicherkapazität von hunderten Knoten aufbringen kann.

7. ZUSAMMENFASSUNG

Stormbot hat durch seine Peer-to-Peer-Struktur und erste primitive Sicherheitsverfahren in Form einer 40-Bit XOR-Verschlüsselung die Problematik der Botnetzwerke auf eine neue Ebene verlagert. Das Stormbotnetzwerk konnte sich über ein Jahr behaupten und in dieser Zeit Unmengen an Spam-Mails versenden und einige DDoS-Angriffe starten. Obgleich die gewählte Methode der Verbreitung über Spam-Mails keinerlei Sicherheitslücken ausnutzte und nur auf Social Engineering basierte, erwies sie sich dennoch als effektiv. Die Zerschlagung des Botnetzwerkes stellte seine Gegner vor ein größeres Problem aufgrund fehlender Zentralisierung. Da allerdings das verwendete Protokoll des Stormbotnetzwerkes identifiziert und dessen Schwachstellen lokalisiert werden konnten, war es möglich das Botnetz durch die Einspeisung künstlicher Bots zu infiltrieren und die Kontrollstrukturen des Botnetzwerkes zu übernehmen. Durch Reverse Engineering war es möglich die im Stormbotnetzwerk verwendeten Sicherheitsalgorithmen zu umgehen und die Treffpunkte der Knoten untereinander im Voraus zu berechnen. Heute hat Stormbot keinen nennenswerten Einfluss mehr. Dennoch bietet es durch die geführte Pionierarbeit in dem Bereich der Peer-to-Peer-Botnetzwerke eine ausreichende Basis für weitere Generationen von Botnetzwerken.

Peer-to-Peer-Botnetzwerke sind ernst zunehmende neue Herausforderungen im Bereich der Internetsicherheit. Das größte Problem besteht in der Dezentralisierung, welche dazu führt, dass alle Bots unabhängig sind und ein Loch im Netzwerk keine größeren Schäden anrichtet. Die Problematik besteht darin eine Masse von Bots einzeln auszuschalten oder die steuernden Knoten zu übernehmen. Auch gibt es keinen zentralen Server, durch dessen Abschaltung das Netzwerk zerschlagen werden könnte. Zur Verbreitung von Befehlen wird immer ein Treffpunkt für die Netzteilnehmer benötigt. Dieser Treffpunkt könnte die potentielle Schwachstelle aller Peer-to-Peer-Netzwerke darstellen. Ein Reverse Engineering ist zu diesem Zweck sehr hilfreich, kann aber bei intelligenten Bots zu unerwünschten Nebenwirkungen führen, wie Angriffe auf die Analysten. Ein ausführliches und intensives Reverse Engineering benötigt allerdings viel Zeit. Das System des Botnetzwerke konnte in diesem Fall zwar durch eine Sybil-Attacke gebrochen werden, jedoch gibt es bereits Methoden um Sybil-Attacken vorzubeugen, wodurch ein P2P-Netzwerk wesentlich schwieriger zu infiltrieren ist.

Die Struktur von Peer-to-Peer-Netzwerken könnte durch erhöhtes modulares Design und mehr Flexibilität in der Art der Kommunikation verbessert werden. Durch die Verwendung bekannter und bewährter kryptographischer Verfahren kann die Sicherheit eines P2P-Netzwerkes weiter erhöht werden. Dabei können bereits einfache und viel erprobte Sicherheitsverfahren wie ein Diffie-Hellmann-Schlüsselaustausch, synchrone und asynchrone Verschlüsselungsverfahren mittels bekannter Kryptographieverfahren wie AES[5] oder RSA einen wesentlichen Beitrag zur Sicherung des Peer-to-Peer-Netzwerkes beitragen.

8. REFERENCES

[1] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon. Peer-to-peer botnets: Overview and case study. Technical report, The Johns Hopkins University, Georgia Institute of Technology, University of Noth Carolinat at Charlotte, 2007.

[2] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freiling. Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm. Technical report, Universität Mannheim, Institut Eurècom, 2007.

[3] F. Leder, G. Wicherski, T. Werner, and M. Schlösser. Stormfucker: Owning the storm botnet. 25th Chaos Communication Congress, Dezember 2008.

[4] B. N. Levine, C. Shields, and N. B. Margolin. A Survey of Solutions to the Sybil Attack. Tech report 2006-052, University of Massachusetts Amherst, October 2006.

[5] P. Maymounkovand and D. Mazi'eres. Kademlia: A peer-to-peer information system based on the xor metric. Technical report, New York University, 2002.

[6] McAfee and T. S. tm. Storm tracker, 2010.

[7] SecureWorks. Storm worm ddos attack, Februar 2007.

[5]Advanced Encryption Standard ist ein symmetrischer Verschlüsselungsalgorithmus zur Verschlüsselung von Daten

doi: 10.2313/NET-2011-05-2_12