Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Seminarausarbeitung - Master Semester 2

André Harms

Ausbreitungssimulation von Schadsoftware - Simulation of malware propagation

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

1 Einführung

Kritische Infrastrukturen von Ländern oder wichtige wirtschaftliche Einrichtungen werden mittlerweile immer häufiger Opfer von zielgerichteten Cyber-Angriffen. Die wohl bekanntesten Beispiele für ein solches Vorgehen sind ein politisch motivierter Angriff auf Estlands digitale Infrastruktur im Jahre 2007 (Arquilla, 2011) (John J. Kelly, 2008) und der Einsatz von Stuxnet, der das iranische Atomprogramm zum Angriffsziel hatte (Symantec Corporation, 2011).
Die Angriffsmöglichkeiten sind dabei vielfältig. Um eine Attacke zu realisieren, werden häufig mehrere Angriffsvektoren verwendet. Generell ist die Verbreitung von Schadsoftware und das Angreifen von IT-Systemen ein facettenreicher Prozess. Für das Opfer kommt dabei der Angriff meist unvorhergesehen und ist im Vorwege und auch im Nachhinein nicht oder nur schwer nachzuvollziehen. Wie die prominenten Fälle Stuxnet, Doqu und Flame gezeigt haben, ist neben Netzwerkangriffen vor allem das allgemeine sowie das Ausbreitungsverhalten von Schadsoftware von Interesse.
Möchte man bestehende Systeme auf ihre Anfälligkeit von Angriffen testen, besteht die Möglichkeit diese mit Penetration- und Vulnerability-Tests zu ermitteln. Solche Tests kommen echten - aber abgesprochenen - Angriffen gleich. Somit besteht auch die Gefahr, ein System unbeabsichtigt bei so einem Test zu beschädigen und einen Ausfall hervorzurufen (Messner, 2011). Diese Tatsache birgt vor allem beim Testen von kritischer Infrastruktur - aufgrund ihrer Bedeutung - Gefahren. Daher bieten sich Simulationen von Angriffen an, um Erkenntnisse über die Anfälligkeit zu erlangen.

1.1 Simulationsarten im Bereich Schadsoftware

In der Vergangenheit wurden verschiedene Ansätze zur Simulation von Schadsoftware realisiert. Sie reichen von Simulationen, die zur Lehre gedacht sind, bis hin zu Simulatoren zum Testen von Anti-Viren Lösungen. Diese Unterteilung wird auch in der betreffenden Literatur schon gemacht (vgl. (Gordon und Ford, 1996)).

Simulationen für die Lehre haben verschiedene Ziele. Frühe Beispiele für Implementierungen dieser Art sind Virlab, welches die Infektionsmöglichkeiten eines einzelnen Systems visualisiert und die Virus Simulation Suite, die audiovisuelles Verhalten einiger ausgewählter älterer Viren nachahmt, um die Effekte zu veranschaulichen. Beide erwähnten Arbeiten sind veraltet, da moderne Schadsoftware andere Ausbreitungswege kennt und sich die Virus Simulation Suite auf DOS Viren beschränkt. Neben den schon erwähnten Ansätzen existieren Arbeiten, die es ermöglichen echten Schadcode in einer isolierten Umgebung (z.B. virtuelle Maschinen) auszuführen, um sein Verhalten dort zu beobachten.

Simulatoren zum Testen von Anti-Viren Lösungen dienen dazu, die verwendete Anti-Viren Software zu testen, ohne echte, schadhafte Viren verwenden zu müssen. Die einzige bekannte Implementierung dieser Art ist der Rosenthal Virus Simulator, der Dateien mit der Signatur einiger Viren erstellen kann, ohne dass diese schadhafte Routinen enthält. Da die letzte Version dieser Implementierung 1996 veröffentlicht wurde, besteht keine aktuelle Relevanz mehr.

Weiterhin gibt es Simulationsvarianten, die sich mit der Ausbreitung von Schadsoftware beschäftigen. Hier existieren rein mathematische Simulationen auf Basis von Modellen zur biologischen Infektionsausbreitung, um eine makroskopische Betrachtung zu erlauben (vgl. (Lora Billings, 2002)). Eine detaillierte Betrachtung der Ausbreitung erlauben unter anderem Multi-Agenten-Simulationen, wie noch in Abschnitt 2.3 zu sehen sein wird.

2 Verwandte Arbeiten

2.1 Botnet Evaluation Environment

Die Botnet Evaluation Environment (BEE) (Barford und Blodgett, 2007) wurde von der University of Wisconsin-Madison entwickelt, um ein tieferes Verständnis für die Funktionsweise von Botnetzen zu schaffen. Die hieraus abgeleiteten Erkenntnisse sollen es ermöglichen, geeignete Gegenmaßnahmen zu treffen. Dabei schafft BEE eine skalierbare Laborumgebung, die es ermöglicht, kleine aber auch große Botnetze mit tausenden von Bots zu testen. Die verwendeten Bots sind dabei echte Bot-Clients und werden nicht simuliert; es wird also echter Code ausgeführt.

2.1.1 Details

BEE stellt ein Framework für den Einsatz in Emulab[1] - einer Netzwerk Emulations-Umgebung - dar. Die Emulation geschieht mittels eines Rechenzentrums und virtueller Maschinen, die hier gestartet und entsprechend virtuell vernetzt werden können. Das BEE-Framework besteht aus VM Images, die in Emulab eingesetzt werden können. Dabei ist es möglich, mit Emulab eine beliebige Topologie simulieren zu lassen, in der das Botnetz agieren kann. Zudem ist es möglich, in sich abgeschlossene Experimente durchzuführen. BEEs vorgefertigte VM Images, welche aus einer Kombination eines Betriebssystems und eines Bots bestehen, werden für ein Experiment in beliebiger Häufigkeit gestartet, so dass mehrere autarke Bot-Instanzen zur Verfügung stehen.

[1] http://www.emulab.net/

Neben den verwendeten Instanzen von virtuellen Maschinen kommen grundlegende Dienste im emulierten Netzwerk zum Einsatz, die benötigt werden, um ein Botnetz zu betreiben. Hierzu gehören DHCP-, DNS- sowie IRC-Server, die vor allem für die Kommunikation bei vielen Botnetzen notwendig sind. Hierüber werden die Clients mit Befehlen zu Aufgaben versorgt werden (Command and Control Nachrichten). Ein Beobachten dieser Nachrichten und dadurch auch des Verhaltens von Bots innerhalb der Umgebung ist somit möglich.

Eine Schwierigkeit, die sich beim Verwenden von virtuellen Maschinen ergibt, ist die in manchen Bots eingebaute VM-Detection, die ein Ausführen in einer virtuellen Maschine verhindert. Dieser Mechanismus soll Reverse-Engineering erschweren. Die gewählte Lösung, dieses Problem zu umgehen, sieht zwei Möglichkeiten vor. Falls der Botnet-Client als Quellcode vorliegt, wird die VM-Erkennung auskommentiert und der modifizierte Client für das Image verwendet. Sollte der Quelltext allerdings nicht vorliegen, besteht die Lösung darin, einen dedizierten Computer einzurichten, der den unmodifizierten Bot-Client ausführt und in das Emulab-Netzwerk eingebunden wird. Darunter leidet allerdings die Skalierbarkeit; der Vorteil einer vollständig virtualisierten Umgebung geht verloren.

2.2 Drive-by-Download Simulator

Um Studenten die Möglichkeit zu bieten, Angriffe nachzustellen oder selber zu entwickeln - also mit ihnen reale Erfahrung zu sammeln -, wurde an der University of Calgary der Lightweight Drive-by Download Simulator (McDonald u. a., 2010) entwickelt. Dieser stellt eine geschlossene Umgebung sicher, in der Experimente und Beobachtungen gemacht werden können, ohne unbeteiligte Netze und Computer zu gefährden. Die Beschränkung auf Drive-by-Downloads[2] ergibt sich aus dem Wunsch, beliebigen Schadcode bei seiner Verbreitung zu beobachten, dies allerdings nur an einzelnen Computern. Da aber für ein realitätsnahes Verbreitungsverhalten - unter Einsatz verschiedener Verbreitungswege - mehrere Computer oder virtuelle Maschinen nötig wären, entschloss man sich, einen Ausbreitungsweg beliebigen Codes mit einem bekannten Infektionsvektor zu untersuchen. Da Drive-by-Downloads eine relevante Ausbreitungsvariante darstellen, fiel die Entscheidung auf diesen Ausbreitungsweg.

[2] Drive-by-Downloads sind unter Ausnutzen von Sicherheitslücken des verwendeten Browsers realisierbar, durch die es möglich ist, eine Datei ohne Willen des Nutzers herunter zu laden und auszuführen; ein Ausbrechen aus der Sandbox des Browsers ist hierzu im Normalfall nötig. Hierdurch kann ungewollt Schadsoftware installiert werden.

2.2.1 Details

Der Lightweight Drive-by Download Simulator basiert auf einer früheren Arbeit der Autoren, dem Spamulator. Dieser war für die Simulation von Spamverteilung vorgesehen und stellt dem Benutzer eine transparente Simulation des Internets zur Verfügung. Hierzu werden an Internet-Dienste gestellte Anfragen an lokal installierte Dienste umgeleitet. Somit sind umfassende Experimente an einem Rechnerplatz möglich, ohne zusätzliche Infrastruktur bereit stellen zu müssen.

Die Studenten nehmen bei der Versuchsdurchführung (siehe Abb. 1) die Rolle des Angreifers ein, der den Exploit entwickelt und den schadhaften Code verbreiten will. Dabei wird sich darauf beschränkt, Exploits für Firefox zu entwickeln und auszunutzen.

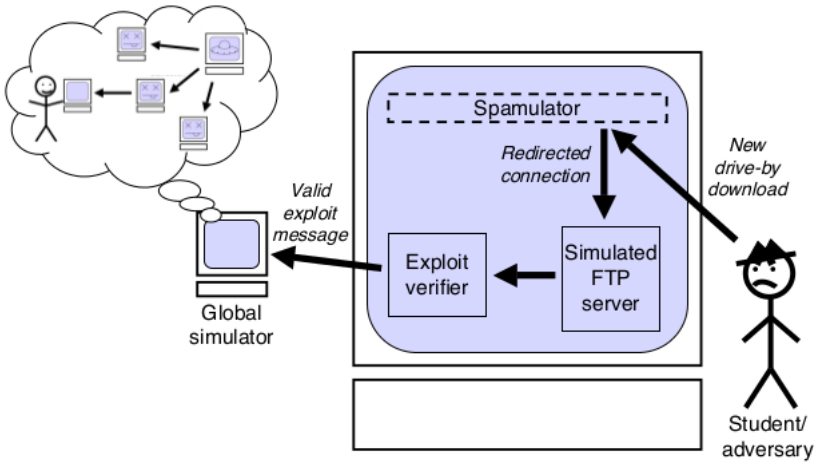

Abbildung 1: Versuchsdurchführung (schematisch) (McDonald u. a., 2010)

Der Student lädt seinen Drive-by-Download auf den Infektionsherd, das sog. Mutterschiff, welches durch einen FTP-Server repräsentiert wird. Anschließend surft der Student die von ihm präparierte Seite innerhalb des simulierten Internets an. Geschieht ein Download vom FTP-Server, wird ein Prozess - der Exploitverifier - dazu veranlasst, eine Überprüfung des Browsers vorzunehmen. Bei dieser Überprüfung wird festgestellt, ob das Ausnutzen eines Exploits und der Drive-by Download tatsächlich funktioniert haben. Wenn ein erfolgreicher Angriff attestiert wird, wird der *global simulator*, welcher die Angriffsmeldungen im Labor zentralisiert visualisieren kann, benachrichtigt. Somit ist eine zusammenfassende Visualisierung mehrere Testsysteme möglich. Das Verifizieren durch den Exploitverifier geschieht durch Prozessmonitoring, bei dem das Verhalten der Browserinstanz beobachtet wird. Hierzu werden

Systemcalls, welche vom Browser getätigt werden, verfolgt. Werden Systemaufrufe gemacht, die als *bad* eingestuft sind, gilt der Browser als kompromittiert und der *global simulator* wird informiert. Somit ist das Testen von selbst erstellten oder auch fremden Drive-by-Downloads möglich und kann innerhalb einer minimalen Laborumgebung durchgeführt werden.

2.3 Mobile Agent Malware Simulator

Der Mobile Agent Malware Simulator (Leszczyna u. a., 2008a) - kurz MALSim - wurde am Joint Research Centre der Europäischen Kommission entwickelt, um Angriffe durch Schadsoftware auf kritische Infrastruktur zu simulieren. MALSim stellt ein Framework dar, welches zur Simulation von verschiedenartiger Malware genutzt werden kann. Es lässt sich das Verhalten von Viren, Würmern und anderen Schädlingen, sowie ihre Eigenschaften (z.B. polymorph, metamorph, usw.) in einer definierten Umgebung simulieren.

2.3.1 Details

MALSim basiert auf dem Multi-Agenten Framework JADE [3] und modelliert Schadsoftware als mobile Agenten, die sich bei einem Versuch innerhalb der Experimentierumgebung bewegen können. Um ein Experiment durchzuführen, wird die zu testende Umgebung - zum Beispiel das Computernetz eines Kraftwerkes - im Labor physisch nachgebaut.

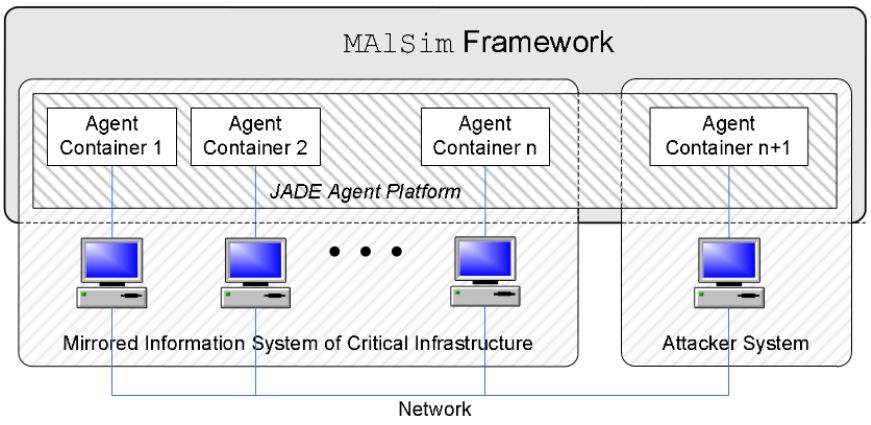

Abbildung 2: Schematischer Versuchsaufbau (Leszczyna u. a., 2008a)

Auf den einzelnen Rechnern wird das MALSim Framework installiert. Als Bestandteil des Frameworks dient JADE als Ausführungsumgebung für die einzelnen Instanzen von Agenten (vgl. Abb. 2). Ein Agent kann sich dabei von einem System der Simulationsumgebung auf

[3] http://jade.tilab.com/

ein anderes transferieren und dort weiter ausgeführt werden (mobile Agenten). Dies stellt die Propagierung innerhalb des zu testenden Systems dar. Es besteht also die Möglichkeit, auch innerhalb der Simulationsumgebung genau zu bestimmen, auf welcher physikalischen Rechnereinheit sich ein Schädling zur Zeit befindet. Das Verhalten eines Agenten bestimmt letzten Endes, wie sich dieser in der Simulationsumgebung repliziert oder Schaden verursacht. Dieses wird bei der Simulation von bekannten Schädlingen aus Virendokumentationen (z.B. von diversen Anti-Viren-Software Herstellern) abgeleitet und in Templates codiert (Leszczyna u. a., 2008b).

Konzeptionell besteht ein Template aus drei Teilen:

- **Initiales Ereignis**: Ereignis, das den Beginn des Lebenszyklus einer Schadsoftware beschreibt (Geburt).
- **Auslöser**: Bedingungen, die erfüllt sein müssen, damit die Schadsoftware funktionieren kann.
- **Aktionen**: Funktionen, die den schadhaften Teil eines Virus, Wurms, etc. ausmachen und zur Verbreitung beitragen.

```
Initial event: Sending e-mail with file called LIST.DOC, which contains passwords
for X-rated websites.

Trigger: Opening the file LIST.DOC in Microsoft Word.

Action 1: Propagating to other computers.

 1. CONNECT(MAlSim)
 2. IF "HKEY_CURRENT_USER\Software\Microsoft\Office\"→"Melissa?" EQUALS "...by
    Kwyjibo" THEN END
    // checking if the routine has been executed previously on the current
    machine
 3. OPEN(MS Outlook)
 4. MAPI_GET(userProfile)
    // getting user profile to use MS Outlook
 5. CREATE(eMailMessage)
 6. FOR {c=0; c⩽50; eMailMessage.addresse = msOutlook.addressBook.contact[c]};
    // setting the message with up to 50 addresses from MS Outlook Address
    Book
 7. eMailMessage.subject = "Important Message From msWord.document.author"
 8. eMailMessage.body = "Here is that document you asked for ... don't show
    anyone else ;-)"
 9. eMailMessage.attachments[0] = msWord.document.this
    // attaching the active WORD document to the email message
10. SEND(eMailMessage)

Action 2: Modifying Word documents.
 1. IF system.time.minutes EQUALS system.date.day AND (msWord.event EQUALS
```

Abbildung 3: MALSim Templatedefinition (Melissa Wurm) (Leszczyna u. a., 2008b)

In Abbildung 3 ist die Definition der elementaren Eigenschaften des Wurms Melissa aus dem Jahre 1999 zu sehen. Dieser verbreitete sich via E-Mail, wie anhand der Definition zu

erkennen ist (Propagation to other computers: Action 1). Auch geht aus der Definition hervor, dass das initiale Ereignis zum Verbreiten des Schädlings das Versenden einer E-Mail mit einer Datei namens LIST.DOC und der Auslöser einer Infektion das Öffnen dieses Anhangs ist. Aus solchen Definitionen lassen sich konkrete Implementierungen erstellen, welche als Teil eines Agenten in einem Experiment zur Ausführung kommen. Schadhafte Aktionen und Änderungen am System, wie zum Beispiel das Verändern von Registrierungseinträgen, geschehen dabei tatsächlich auf dem infizierten System und werden nicht simuliert.

3 Zusammenfassung

Es existieren mehrere verschiedenartige Ansätze zur Simulation von Schadsoftware. Eine komplexe Variante ist die Simulation des Ausbreitungsverhaltens, da hier neben der konkreten Ausbreitung auch das sonstige Verhalten (Kommunikation, Destruktivität, etc.) berücksichtigt werden muss. Die vorgestellten Arbeiten unterscheiden sich vor allem in dem, was simuliert werden soll. Daher ergeben sich auch unterschiedliche Herangehensweisen.

Die Botnet Evaluation Environment ist zum Beobachten von Botnetzen geeignet und kann durch das Verwenden von Emulab gut skalieren. Sie ermöglicht einen genauen Einblick in die Kommunikation und das Verhalten eines Botnetzes, berücksichtigt die Ausbreitung eines Botnetzes allerdings nicht.

Der Lightweight Drive-by Download Simulator hingegen, kann dazu genutzt werden, Exploits zu verifizieren, die einen Drive-by-Download ermöglichen. Hierdurch kann beobachtet werden, wie eine Schadsoftware durch einen bestimmten Ausbreitungsvektor propagiert. Durch den komplexen Vorgang des Monitorings des Firefox-Prozesses - zum Erkennen eines erfolgreichen Angriffs -, kann allerdings nur dieser als Angriffsziel genutzt werden. Ein anderer Browser würde ein anderes erwartetes Verhalten aufweisen und das Monitoring müsste angepasst werden.

MALSim eignet sich zur Ausbreitungssimulation von verschiedenartiger Malware, die durchaus verschiedene Ausbreitungsvektoren verwenden können. Dies kann innerhalb einer Nachbildung einer beliebigen - vornehmlich aber kritischen - Infrastruktur geschehen. Die Verwendung von mobilen Agenten, die die schadhafte Software simulieren, ermöglicht eine genaue Beobachtung des Ausbreitungsverhaltens, da sich ein Schädling auch tatsächlich auf der physischen Einheit befindet, die infiziert wurde. Durch die Notwendigkeit eine eins zu eins Abbildung der zu testenden Infrastruktur bilden zu müssen, skaliert MALSim allerdings nicht. Des Weiteren findet keine Betrachtung des Einflusses des Benutzerverhaltens auf die Ausbreitung, und der Ausbreitung via mobiler Geräte oder Wechselmedien statt.

4 Ausblick

Im weiteren Verlauf des Masterstudiums soll die Möglichkeit geschaffen werden, die Ausbreitung von Schadsoftware über mehrere Ausbreitungswege simulieren zu können. Dabei soll es zum Beispiel möglich sein, Angriffe auf kritische Infrastruktur mit geringem Hardwareaufwand nachzuahmen. Zudem sollen neben der Propagierung über feste Kommunikationsnetze auch Ausbreitungswege über mobile Geräte sowie der Einfluss des Nutzerverhaltens betrachtet werden können.

5 Literatur

[Arquilla 2011] Arquilla, John: From blitzkrieg to bitskrieg: the military encounter with computers. In: *Commun. ACM* 54 (2011), Oktober, S. 58–65. – URL http://doi.acm.org/10.1145/2001269.2001287. – ISSN 0001-0782

[Barford und Blodgett 2007] Barford, Paul ; Blodgett, Mike: Toward botnet mesocosms. In: *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*. Berkeley, CA, USA : USENIX Association, 2007 (HotBots'07), S. 6–6. – URL http://dl.acm.org/citation.cfm?id=1323128.1323134

[Gordon und Ford 1996] Gordon, Sarah ; Ford, Richard: Real world anti-virus product reviews and evaluations - Part 1. In: *Netw. Secur.* 1996 (1996), Dezember, Nr. 12, S. 14–18. – URL http://dx.doi.org/10.1016/S1353-4858(00)90009-6. – ISSN 1353-4858

[John J. Kelly 2008] John J. Kelly, Lauri A.: eWMDs. In: *Policy Review No. 152* (2008). – URL http://www.hoover.org/publications/policy-review/article/5662. – abgerufen 10.01.2012

[Leszczyna u. a. 2008a] Leszczyna, Rafał ; Fovino, Igor N. ; Masera, Marcelo: MAISim: mobile agent malware simulator. In: *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST, Brussels, Belgium, Belgium : ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008 (Simutools '08), S. 35:1–35:6. – URL http://dl.acm.org/citation.cfm?id=1416222.1416262. – ISBN 978-963-9799-20-2

[Leszczyna u. a. 2008b] Leszczyna, Rafał ; Fovino, Igor N. ; Masera, Marcelo: Malware Templates for MAlSim / European Commission Joint Research Centre, Institute for the Protection and Security of the Citizen. 2008. – Forschungsbericht

[Lora Billings 2002] Lora Billings, Ira B. S.: A unified prediction of computer virus spread in connected networks. In: *Physics Letters A* 297 (2002), S. 261–266

[McDonald u. a. 2010] McDonald, Matthew ; Ong, James ; Aycock, John ; Friess, Nathan: *A Lightweight Drive-by Download Simulator*. 2010

[Messner 2011] Messner, Michael: *Metasploit: Das Handbuch zum Penetration-Testing-Framework*. 1. Auflage. d.punkt Verlag, 2011. – 13–58 S

[Symantec Corporation 2011] Symantec Corporation: W32.Stuxnet Dossier / Symantec Corporation. 2011. – Forschungsbericht. abgerufen 01.04.2011