FAKULTÄT FÜR !NFORMATIK

Einsatz von Honeyclient-Technologien zur Steigerung der Sicherheit im universitären Umfeld

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur/in

im Rahmen des Studiums

Software and Information Engineering & Internet Computing

eingereicht von

Christian Kekeiss
Matrikelnummer 0025971

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuer: Univ.-Prof. Dipl.-Ing. Dr. techn. Thomas Grechenig

Wien, 08.10.2008

(Unterschrift Verfasser/in) (Unterschrift Betreuer/in)

Einsatz von Honeyclient-Technologien zur Steigerung der Sicherheit im universitären Umfeld

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software and Information Engineering & Internet Computing

eingereicht von

Christian Kekeiss

0025971

ausgeführt am

Institut für Rechnergestützte Automation

Forschungsgruppe Industrial Software

der Fakultät für Informatik der Technischen Universität Wien

Betreuung:

Betreuer: Univ.-Prof. Dipl.-Ing. Dr. techn. Thomas Grechenig

Mitwirkung: Florian Fankhauser

Wien, 08.10.2008

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Wien, am _____

Name

Danksagung

Zunächst möchte ich Dipl.-Ing. Florian Fankhauser für seine ausgezeichnete Betreuung und die vielen wissenschaftlichen Hinweise beim Erstellen der Arbeit danken. Weiters danke ich Prof. Grechenig für die Betreuung und Unterstützung bei der Arbeit.

Während meines Studiums haben mich viele Personen begleitet und unterstützt, vor allem möchte ich an dieser Stelle Thomas, Linda und Gerhard danken.

Weiters bedanke ich mich bei meinem Onkel Robert für das unermüdliche Korrekturlesen und die vielen Anmerkungen bei der vorliegenden Arbeit, die mich oft auch aufgemuntert und zum Lachen gebracht haben.

Meinen Eltern und Geschwistern danke ich für ihre Unterstützung während des gesamten Studiums und auf meinem Lebensweg.

Ganz besonderer Dank gebührt meiner Freundin Steffi für die Geduld und Ausdauer, die sie mir beim Erstellen der Arbeit entgegengebracht hat - you ´re the best!

Zusammenfassung

Angriffe auf IT-Systeme sind heute weit verbreitet und stellen eine große Gefahr für Computernetzwerke dar. Bisher wurden dabei vor allem Server angegriffen, welche allerdings bereits relativ gut erforscht sind, wobei sich im Forschungsbereich Honeypots und Honeynets als Analysetool etabliert haben.

In letzter Zeit tritt in Netzwerken jedoch immer häufiger eine neue und weitgehend unerforschte Angriffstechnik auf, die eine ernstzunehmende Gefahr für IT-Systeme darstellen. Dabei handelt es sich um client-seitige Angriffe, bei denen Fehler in der Clientsoftware für Attacken genutzt werden. Die Clientrechner können dabei vom User völlig unbemerkt angegriffen werden. Ein Detektionstool für client-seitige Angriffe ist daher von großer Bedeutung für die Sicherheit von Computersystemen, da bisherige Techniken dafür nicht genutzt werden können.

Einen Lösungsansatz bietet jedoch die neu entwickelte Methodik der Honeyclients, welche für die Detektion solcher Angriffe entwickelt wurden. Dadurch stellen sie einen sehr wichtigen Faktor in der Erhöhung der Sicherheit für IT-Systeme dar.

Honeyclients sind ein spannendes, neues Forschungsgebiet mit vielen unterschiedlichen Umsetzungsvarianten. Ein Vergleich der Leistungsfähigkeit und der Eignung für spezifische Problemstellungen der einzelnen Systeme liegt jedoch bisher kaum vor. Die vorliegende Arbeit leistet einen Beitrag zu einem übersichtlichen Vergleich zwischen verschiedenen Honeyclients. Es wird zunächst eine Einführung in das Konzept der Honeyclients gegeben. Anschließend werden unterschiedliche Implementierungen von Honeyclients vorgestellt und in Hinblick auf eine Anwendung in einem universitären Netzwerk evaluiert. Abschließend werden die Daten daraufhin analysiert, ob die theoretischen Unterschiede zwischen verschiedenen Honeyclients, die aus der Literatur bekannt sind, auch in der Praxis bestehen.

Keywords:*client-seitige Angriffe; Honeyclient; Malware; Computersicherheit; universitäre Netzwerke*

Abstract

Attacks on IT systems are widespread these days and pose an enormous danger to computer networks. For a long time, mainly servers were attacked, but these attacks are relatively well investigated. In the area of research honeypots and honeynets have been established in order to investigate this kind of attacks.

Recently a new and generally unknown type of attacks, which pose a serious threat to IT systems are monitored more frequently in networks. These are so-called client-side attacks, which use faults in the client-software as initial point. Thereby the client computers are attacked by resources, which are hidden in the site that is accessed by the user. This is extremely malicious, as the user does not take any notice of this attack. Therefore a tool for the detection of client-side attacks is of extreme importance for the security of computer systems. Tools that are well established for other kind of attacks can not be used for that purpose.

A possible resolution is provided by the newly developed method of honeyclients, which were tailored for the detection of client-side attacks. Therefore, honeyclients are an important factor in order to increase security for the users of IT systems.

Honeyclients are an exciting new field of research with several different ways of implementation. However, a comparison of the efficiency and suitability of the different systems for specific tasks is still hardly available. This thesis contributes to a clearly arranged comparison of the different implementations of honeyclients. First, an introduction into the idea of honeyclients is given. Subsequently, it presents several implementations of honeyclients and evaluates them in view of their suitability for a university network. Finally, the gained data is analyzed in order to find out whether the theoretical differences that are reported for honeyclients in literature are also existing in practical use.

Keywords: *client-side attacks; honeyclient; malware; IT security; university networks*

Inhaltsverzeichnis

1 Einleitung

Die Nutzung von Computern für den wirtschaftlichen und privaten Bereich hat sich in den vergangenen Jahren enorm vergrößert. Mit dem Anwachsen der Computerbenutzung ist auch die Anzahl der Schadprogramme, die Computersysteme angreifen können, kontinuierlich gewachsen.

Vor allem seit der Einführung des Internets ist die Verbreitung von Malware über Netzwerke stark gestiegen. Zahlreiche Studien belegen, dass viele dieser Programme im Internet zu finden sind und eine große Gefahr für IT-Systeme darstellen Im Jahr 2005 etwa haben America Online (AOL) und das National Cyber Security Alliance (NCSA) [3] 329 Computer von Kunden untersucht und dabei festgestellt, dass 80% davon mit Malware infiziert waren.

Vor allem für Firmennetzwerke stellt Malware, die über das Internet verbreitet wird, eine bedeutende Quelle wirtschaftlichen Schadens dar. Dies gilt auch für universitäre Netzwerke, da diese von einer sehr großen Anzahl von Usern für viele unterschiedliche Zwecke genutzt werden. Eine Studie von Saroiu et al. [54] befasst sich mit der Verbreitung von Spyware in einem universitären Netzwerk. Dabei untersuchten sie das Verhalten von vier derartigen Programmen (Gator, Cydoor, SaveNow und eZula) und erstellten daraus Signaturen über den Datenverkehr. Mit Hilfe dieser Signaturen analysierten sie die Verbreitung dieser Programme im Netzwerk der Universität von Washington. Dabei zeigte sich, dass auf 5,1% der Computer mehr als ein Spywareprogramm unwissentlich installiert worden war.

Die Entwicklung von Detektions- und Sicherheitstools ist deshalb von größter Bedeutung für die Computersicherheit. Bisher hat man sich dabei hauptsächlich auf server-seitige Angriffe konzentriert, die bereits gut erforscht sind. Dabei haben sich Honeypots und Honeynets als Werkzeug etabliert. Auch Computerwürmer oder Trojanische Pferde, können durch Tools wie Firewalls und Antivirensoftware meist erfolgreich abgewehrt werden.

Immer häufiger tritt jedoch in letzter Zeit eine neue Art von Angriffen auf. Bei diesen so genannten client-seitigen Angriffen werden Fehler in der Clientsoftware ausgenutzt um Malware auf einem Rechner zu installieren. Dies kann vom User völlig unbemerkt geschehen und diese Art von Angriffen kann mit herkömmlichen Tools nicht detektiert oder abgewehrt werden.

Wie aktuell das Problem von client-seitigen Angriffen ist, belegt eine Studie, die das Honeynet Project [85] 2007 über derartige Attacken im Internet durchgeführt hat. Dabei wurden 300.000 Internetseiten untersucht und in Kategorien eingeteilt. Hier zeigte sich, dass Adult Seiten die meisten bösartigen Adressen enthalten (0,57% der untersuchten Adressen).

Anfang 2007 untersuchten Provos et al. [50] 66 Millionen Adressen. Die Internetseiten wurden auch in Gruppen eingeteilt. In der Kategorie Adult wurden die meisten Seiten (0,6%) mit Malware, welche sich automatisch auf dem System installiert, entdeckt. Dieses Ergebnis deckt sich mit denen des Honeynet Project. Außerdem stellten die Forscher fest, dass bei 1,3% der Suchanfragen an die Suchmaschine Google mindestens ein Link zu einer gefährlichen Seite im Ergebnis enthalten ist. Weiters konnten sie feststellen, dass 67% der bösartigen Internetseiten auf Servern in China gespeichert sind. Auf dem zweiten Platz liegen die Vereinigten Staaten mit 15%

In der Forschung stellen Honeyclients ein neues Tool dar, um client-seitige Angriffe zu detektieren. Da dieser Bereich, jedoch noch relativ unerforscht ist, fehlte bisher ein Vergleich der einzelnen Systeme und Daten über deren Eignung für spezifische Problemstellungen. Die vorliegende Arbeit versucht daher die Frage zu klären, welche Gemeinsamkeiten und Unterschiede die verschiedenen Systeme aufweisen. Von besonderem Interesse ist dabei, welche Honeyclients sich für die Nutzung in einem universitären Netzwerk eignen, da in derartigen Netzwerken ein großes Gefahrenpotential für client-seitige Angriffe über das Internet vorhanden ist.

Aufgrund der bisher aus der Literatur bekannten Systeme, ist zu erwarten, dass sich nicht alle Honeyclients gleichermaßen für die Nutzung in einem universitären Netzwerk eignen. Vor allem für die beiden verschiedenen Typen von Honeyclients, high-interaction und low-interaction, sind dabei unterschiedliche Ergebnisse zu erwarten. Die Daten aus der Literatur deuten etwa darauf hin, dass high-interaction

Honeyclients in der Lage sind auch unbekannte Angriffe zu erkennen, während low-interaction Honeyclients nur bereits bekannte Angriffe detektieren. Allerdings ist auch zu erwarten, dass low-interaction Honeyclients wesentlich schneller in der Verarbeitungsgeschwindigkeit sind.

Anhand von ausgewählten Honeyclients soll in der Arbeit evaluiert werden, ob die theoretischen Hypothesen aus den wissenschaftlichen Publikationen auch in der Praxis Relevanz haben. Weiters wird untersucht, welche Unterschiede sich für die einzelnen Honeyclients in Hinblick auf ihre Eignung für ein universitäres Netzwerk ergeben.

1.1 Gliederung der Arbeit

Das zweite Kapitel dieser Arbeit beschäftigt sich zunächst mit den Grundbegriffen der Thematik und gibt einen Überblick über die derzeit bekannten Angriffstechniken.

Anschließend wird in Kapitel 3 die Technik der Honeypots und Honeynets näher erläutert, da diese die Grundlage für die Entwicklung von Honeyclients darstellen.

In Kapitel 4 schließlich wird die Technologie der Honeyclients vorgestellt. Zunächst wird der Unterschied zwischen server-seitigen und client-seitigen Angriffen erklärt. Weiters werden die verschiedenen Typen von Honeyclients und die jeweiligen Implementierungen vorgestellt. Dabei werden unterschiedliche Systeme präsentiert und im Detail erläutert, auch auf die Möglichkeiten und Grenzen dieser Technologie wird eingegangen.

Danach werden in Kapitel 5 universitäre Netzwerke im Allgemeinen und das Netzwerk der TU Wien im Besonderen vorgestellt. Weiters wird dabei auf die Sicherheit in derartigen Netzwerken eingegangen.

Kapitel 6 beschreibt die Evaluierungsumgebung der Analyse von Honeyclients, sowie die Angriffe, die im praktischen Teil der Arbeit verwendet werden.

Die Ergebnisse der Evaluierung werden in Kapitel 7 präsentiert und diskutiert.

Abschließend bietet Kapitel 8 eine Zusammenfassung der Arbeit, sowie einen Ausblick auf zukünftige Forschungsziele.

2 Grundlagen

In diesem Kapitel werden grundlegende Begriffe definiert, welche im späteren Verlauf der Diplomarbeit verwendet werden.

2.1 IT-Sicherheit

Eine Definition von Sicherheit gibt zum Beispiel Meyers Großes Lexikon [1] wie folgt:

> Zustand des Unbedrohtseins, der sich objektiv im Vorhandensein von Schutz[einrichtungen] bzw. im Fehlen von Gefahr[enquellen] darstellt und subjektiv als Gewissheit von Individuen oder sozialen Gebilden über die Zuverlässigkeit von Sicherungs- und Schutzeinrichtungen empfunden wird.

Sicherheit kann somit als Abwesenheit von Risiko definiert werden, wobei man Risiko definieren kann als:

> Das Risiko einer Anlage oder Tätigkeit ist die Summe über alle (gefährlichen) Ereignisse der Produkte von Eintrittswahrscheinlichkeit und Schadensausmaß und eventuell (subjektiven) Gewichtungsfaktoren [38].

Das Schadensausmaß kann man genauso wie das Risiko als Wahrscheinlichkeit definieren, also eine Zahl zwischen 0 und 1. Folglich kann man Sicherheit definieren als

$$Sicherheit = 1 - Risiko$$

Nun ist zu definieren, wann ein Szenario als "sicher" gilt, wobei sich hier der Begriff des Grenzrisikos anbietet. Das Grenzrisiko ist ein als vertretbar definiertes Risiko. Vertretbar ist eine individuelle Größe und je nach Fall festzulegen. Eine

Szenario wird “sicher” genannt, wenn das tatsächliche Risiko kleiner als das Grenzrisiko ist. “Unsicher” bedeutet dann, dass das Risiko größer ist als das Grenzrisiko.

> Unter Sicherheit in der Informationstechnologie (IT) verstehen wir den Schutz von Informationen und Informationssystemen gegen unbefugte Zugriffe und Manipulationen sowie die Sicherstellung der Verfügbarkeit der durch die Systeme bereitgestellten Dienste für legitime Benutzer, einschließlich aller Maßnahmen zur Verhinderung, Entdeckung oder Protokollierung von Bedrohungen. Der Schutz vor unbefugtem Zugriff muss ständig gewährleistet sein, insbesondere während der Speicherung, der Verarbeitung und der Übertragung [30, S. 2].

In den letzten Jahrzehnten hat sich IT immer mehr durchgesetzt. In sehr vielen Bereichen befinden sich mittlerweile Computer oder andere elektronische Hilfsmittel, welche für verschiedene Tätigkeiten genutzt werden. Aus sicherheitstechnischer Betrachtungsweise ist die verarbeitete und auf der Hardware gespeicherte Information relevant, woraus sich der Schutzbedarf der Hardware ableitet. Diese Informationen besitzen oft einen hohen Wert, da diese oftmals den Wert eines Unternehmens bestimmen. Für einen Onlineshop sind die Kundendaten für die Abwicklung des Geschäftes von hoher Wichtigkeit.

Diese Informationen müssen geschützt werden, damit sie nicht durch kriminelle Aktivitäten missbraucht werden können. Das Ziel von IT-Sicherheit ist, den Zugriff auf diese Informationen zu schützen.

IT-Sicherheit kann man in verschiedene Schutzziele unterteilen. Diese beschreiben, welcher Schutz für Informationen gewährleistet sein muss. Für die Umsetzung gibt es mehrere Möglichkeiten und je besser ein Ziel umgesetzt ist, umso höher ist die Sicherheit für die Informationen. Je nachdem welche Anforderungen ein System hat, sollen entsprechende Schutzziele umgesetzt werden. Die Umsetzung hängt davon ab, welche Risiken minimiert werden sollen und ob diese mit den gegebenen finanziellen Möglichkeiten umgesetzt werden können.

Im Allgemeinen werden am häufigsten die nachfolgend beschriebenen drei Schutzziele beachtet. Diese können auch in [55, S. 114] nachgelesen werden.

Vertraulichkeit: Informationen müssen vor unbefugten Zugriffen durch Dritte geschützt werden. Dies kann auf einem System, auf dem Daten gespeichert

sind, mittels Zugriffskontrollen geregelt werden. Durch solche Methoden ist es Unbefugten grundsätzlich nicht möglich, diese zu lesen.

Ein Beispiel für Zugriffskontrollen sind die Dateirechte in Unix Betriebssystemen. Jede Datei besitzt mehrere Attribute: das Lesen, das Schreiben und das Ausführen einer Datei. Außerdem wird gespeichert, welche Rechte für die Datei der Ersteller dieser hat, die Gruppe, in der er sich befindet, und jeder einzelne Benutzer des Systems.

Bei der Übertragung von Informationen über ein Netzwerk wird Vertraulichkeit meistens mittels kryptographischer Methoden bereitgestellt.

Integrität: Daten sollen nicht von Unbefugten verändert werden können. Hier können ähnliche Methoden wie bei der Vertraulichkeit verwendet werden, um dieses Ziel umzusetzen. Wenn man Vertraulichkeit betrachtet als das Lesen von Daten, kann man Integrität als das Schreiben betrachten. Weiters sollen die Daten nicht von jemandem ohne Berechtigung gelöscht werden.

Dieser Begriff wird auch im Softwarebereich verwendet und beschreibt, dass Programme nicht verändert werden. Dies könnte durch einen Fehler im Programm oder einen Virus durchgeführt werden.

Bei der Übertragung von Informationen über ein Netzwerk muss sichergestellt sein, dass die Daten vom tatsächlichen Absender stammen und unverändert an den Empfänger übertragen wurden. Dies kann auch mittels kryptographischer Methoden sichergestellt werden.

Verfügbarkeit: Daten und Dienste sollen den legitimen Benutzern immer zur Verfügung stehen, wenn diese sie benötigen. Das bedeutet, dass diese Dienste nicht immer aktiv sein müssen, allerdings dann, wenn der Benutzer sie benötigt. In Zeiten von Botnetzen und Distributed Denial of Service (DDoS) Angriffen kann die Verfügbarkeit von Einrichtungen gestört werden. Die dauerhafte Verfügbarkeit, zum Beispiel von Internetdiensten, kann für Unternehmen eine wirtschaftliche Rolle spielen.

Bei den genannten Zielen handelt es sich um die klassischen Säulen der IT-Sicherheit. Im Bedarfsfall lassen sich diese je nach Anwendungsfall auch erweitern, wie zum Beispiel um Anonymität.

2.1.1 IT-Sicherheit als Prozess

Die Schaffung von IT-Sicherheit in einem Projekt ist kein einmaliges Ereignis. Es können jederzeit neue Situationen auftreten, die eine bestehende Architektur nicht behandeln kann. Deshalb muss das System immer wieder an aktuelle Bedürfnisse angepasst und evaluiert werden, ob nicht Änderungen am System vorgenommen werden müssen.

Damit die Sicherheit eines Systems, zum Beispiel durch einen Virenscanner, ständig gegeben ist, reicht es nicht aus diesen einmalig zu installieren. Durch die ständige Entwicklung neuer Viren kann der Scanner aktuelle Viren nicht identifizieren und den Anwender ausreichend schützen. Daher gehören die Signaturen regelmäßig aktualisiert, um weiterhin einen gewissen Schutz garantieren zu können. Dieser Ansatz wird auch von Badhusha et. al. in [5] beschrieben.

Man erkennt, dass Sicherheit nicht ein einmaliges Vorhaben ist, sondern ein Prozess, der immer wieder ausgeführt gehört.

Außerdem müssen auch menschliche Ressourcen in diesem Prozess betrachtet werden. Ist bei einer Firma zum Beispiel nur ein Administrator für das gesamte Netzwerk zuständig und dieser fällt aus, hat dies auch Konsequenzen für die Sicherheit der Systeme. Es werden keine Sicherheitsupdates installiert oder Computerprobleme können nicht behoben werden. Für solche Fälle müssen Pläne existieren, welche das Vorgehen in solchen Situationen beschreiben.

Das Bundesamt für Sicherheit in der Informationstechnik (BSI) stellt für den Grundschutz einer IT-Infrastruktur ein Verfahren bereit, um diesen umzusetzen. Unter Grundschutz versteht man Maßnahmen, welche die Sicherheit von Standardsystemen erhöhen. Dabei wird auf eine ausführliche Risikoanalyse verzichtet. Dadurch ist es möglich IT-Sicherheitskonzepte einfach und ohne großen Aufwand zu erstellen. Anfangs wird dabei die aktuelle Situation des Unternehmens erfasst und danach werden Vorschläge für einen Sicherheitprozess und das weitere Vorgehen gegeben. Dieser Katalog kann unter [9] eingesehen werden.

2.1.2 Softwaresicherheit

Im Jahr 2004 wurde für die Deutsche Telekom ein Portal implementiert, welches die Verwaltung von Kundendaten vereinfachen soll. So soll es Kunden möglich sein,

ihre eigenen Stammdaten und bestellte Services betreuen zu können. Allerdings war es möglich über manipulierte Internetadressen Administratorenrechte auf dem System zu erlangen und Daten zu ändern. Das gesamte Portal musste abgeschaltet werden und neu implementiert werden [32].

Wie man erkennt, ist Sicherheit auch bei der Entwicklung von Software sehr wichtig. Allerdings wird aufgrund von Zeitdruck oder mangelnden Wissens auf eine genaue Spezifizierung und Analyse von Sicherheitsaspekten meistens verzichtet. Dies kann sich allerdings später als Fehler herausstellen. Je später ein Fehler im Softwareentwicklungsprozess gefunden wird, desto mehr Kosten entstehen für das Entfernen dieses Fehlers [7].

Ein weiterer wichtiger Aspekt ist die Frage, ob Fehler in der Software veröffentlicht werden sollen, oder ob die Ersteller von Software die genaue Spezifikation eines Fehlers geheim halten und dem Anwender nur Bugfixes zur Verfügung stellen sollen. Diese beiden Arten werden Full Disclosure und Security by Obscurity genannt [55, S 328]. Unter Full Disclosure versteht man, dass eine gefundene Sicherheitslücke veröffentlicht wird. Dies kann zum Beispiel über eigens erstellte Mailinglisten erfolgen. Dabei werden oft Details und Möglichkeiten publiziert, wie das System konfiguriert werden muss, damit der Fehler nicht mehr auftritt oder ausgenutzt werden kann. Security by Obscurity besagt, dass ein gefundener Fehler in der Software im Allgemeinen nur dem Hersteller gemeldet wird. Der Hersteller kann für den Fehler einen Patch programmieren oder diesen ignorieren, das weitere Vorgehen ist nicht geregelt.

2.2 Malware

Malware wird aus den Begriffen malicious (bösartig) und Software zusammengesetzt und bezeichnet Programme, die vom Benutzer unerwünscht sind oder unerwünschte Funktionen ausführen. Diese Programme laufen meist getarnt im Hintergrund ab. Malware wird nach der Methode ihrer Verbreitung unterteilt.

Computerviren: Dabei handelt es sich um Programme, die sich bei ihrer Ausführung selber reproduzieren. Dazu schreiben sie einen Teil ihres Codes in andere ausführbare Dateien oder Dokumente und sobald diese ausgeführt

werden beginnt der Prozess von vorne [12]. Häufig richten Viren auch Schaden an infizierten Computersystemen an. Die Verbreitung auf andere Computersysteme kann über Speichermedien wie Disketten oder Universal Serial Bus (USB) Sticks oder über Netzwerke, wie zum Beispiel als Email Anhang, erfolgen.

Viren werden unter anderem mittels Signaturen erkannt, die Anti Viren Hersteller benutzen um diese zu detektieren. Eine weitere Möglichkeiten sind Heuristiken, welche Abnormalitäten bei der Verwendung des Systems erkennen. Es ist zum Beispiel auffällig, wenn Dateien im Systemverzeichnis des Betriebssystems gelöscht werden. Dieses Verhalten kann auf einen Virus deuten.

Trojanische Pferde: "Ein Trojanisches Pferd ist ein in ein Programm eingebetteter Code, welcher ungewollte Aktionen durchführt. Ein Beispiel dafür ist eine Loginmaske, welche nicht nur die Anmeldung durchführt, sondern auch unautorisiert Benutzername und Passwort abspeichert."[11]

Wurden anfangs Zugangsdaten für Computer oder Internetverbindungen mit Trojanischen Pferden ausspioniert, versuchen diese Programme in letzter Zeit immer mehr private Daten zu ermitteln. Dies können Bankdaten oder Personendaten sein.

Würmer: Würmer nutzen eine oder mehrere Schwachstellen eines Systems aus, um sich darauf auszubreiten. Außerdem warten sie nicht passiv wie ein Virus auf ihre Ausführung, sondern verbreiten sich automatisch. Oftmals weiten sie sich über ein Netzwerk aus. Tritt eine Infektion eines Wurmes in einem Netzwerk auf, entsteht oft großer Datenverkehr. Dieser wird dadurch verursacht, dass immer mehr Computer in dem Netzwerk befallen sind und eine weitere Verbreitung stattfindet. Dadurch können ganze Systeme ausfallen, wodurch auch finanzielle Schäden für Firmen entstehen können [73]. Computerwürmer werden wie Viren mittels Signaturen oder Heuristiken erkannt. Diese können in einem Intrusion Detection System (IDS) gespeichert sein und so die Ausbreitung eines Wurmes im Netzwerk erkennen. Bekannte Würmer sind Code Red [46] und der Loveletter Wurm [80].

Weiters unterscheidet man Malware nach dem Schaden, den sie auf einem Computersystem anrichtet kann.

Spyware: Darunter versteht man Programme, die ohne das Wissen des Benutzers Informationen sammeln. Das können Verhaltensmuster oder bestimmte sensible Daten auf einem Computersystem sein [54]. Diese werden an Organisationen versendet, welche daraus einen finanziellen Nutzen ziehen können.

Teilweise wird Spyware auch in Programmen versteckt, welche ganz andere Funktionen haben. Ein Beispiel dafür ist Kazaa, ein Client von einem Peer-to-peer Netzwerk. In diesem Fall wurden Informationen über besuchte Internetseiten oder über den Benutzer an den Spywareserver gesendet [8].

Dialer: Ein Dialer ist normalerweise ein Programm, das die Verbindung zu einem entfernten Computer über ein Modem herstellt. Allerdings können diese Programme auch dazu genutzt werden, teure Verbindungen, wie etwa Auslandsleitungen, zu verwenden und dem Benutzer dadurch hohe Kosten zu verursachen. Jedoch sind diese Programme aufgrund der Verbreitung von nicht telefonbasierten Zugangsmethoden zum Internet nicht mehr sehr verbreitet [30, S. 94].

Backdoors: Ein Programm wird so installiert, dass ein Angreifer zu einem späteren Zeitpunkt Zugriff auf ein System bekommt, ohne dass dies von dem Benutzer bemerkt wird. Dadurch muss der Angreifer nicht mehr das System angreifen, sondern kann sich leicht wieder mit diesem verbinden und weiter Aktionen ausführen [77].

Zang et. al. stellen in ihrer Arbeit [77] eine Möglichkeit vor, Backdoors anhand ihres Netzwerkverkehrs zu erkennen. Dazu stellen sie Kriterien für Pakete auf (kleine Paketgröße, große untätige Perioden zwischen Paketen), die sie solchen Programmen zuordnen können.

Rootkit: Dabei werden Programme so modifiziert, dass sie die Handlungen eines Angreifers verbergen. Es können zum Beispiel Befehle des Betriebssystems so verändert werden, dass bestimmte Prozesse oder Verzeichnisse nicht angezeigt werden. Dadurch ist es schwieriger zu überprüfen, ob ein Angriff auf ein System stattgefunden hat.

Um das System zu überprüfen, ob ein Rootkit installiert ist, gibt es mehrere Möglichkeiten. Chkrootkit, zum Beispiel, erstellt von jedem ausführbaren Programm eine Prüfsumme und vergleicht diese zu einem späteren Zeitpunkt mit der aktuellen Prüfsumme. Wenn es einen Unterschied gibt, deutet dies auf eine ungewollte Modifikation des Programms hin. Natürlich müssen die Prüfsummen erstellt werden, wenn das System sich in einem sicheren Zustand befindet [82].

In den Arbeiten [6] und [52] werden Methoden vorgestellt, mit denen Malware analysiert und klassifiziert werden kann. Dadurch können bei ähnlichen Vorkommen wesentlich schneller Signaturen für Antiviren Programme entwickelt werden.

2.3 Ausgewählte Angriffsarten auf Clientsoftware

Es werden verschiedene ausgewählte Arten von Angriffen für Clientsoftware vorgestellt. Diese können von Angreifern benutzt werden, um in ein System einzudringen und dort Schaden anzurichten. Es werden speziell diese Angriffe vorgestellt, da sie im späteren Verlauf der Arbeit für die Analyse von Honeyclients herangezogen werden.

2.3.1 Buffer Overflow

Programme sind meistens im Assemblercode auf einem Rechner gespeichert. Dabei handelt es sich um Maschinenbefehle, welche für den Menschen schwer lesbar sind, wie in [30] erläutert wird. Allerdings werden Anwendungen meistens nicht mittels Maschinenbefehlen programmiert, sondern in einer höheren Programmiersprache wie C oder C++ und dann mit Hilfe eines Compilers in Maschinenbefehle umgewandelt.

Wird ein Programm gestartet, erhält es vom Betriebssystem einen Speicherbereich für Programmcode und Daten zur Verfügung. Beides sind normalerweise Folgen von Bits, wobei der Programmcode Maschinenbefehle enthält, welche ausgeführt werden. Daten werden nicht als Befehle verarbeitet, sondern gelesen und geschrieben. Ein Angreifer könnte nun bei einer Eingabe eines Programmes - anstatt von Daten - Maschinenbefehle eingeben und diese werden im Datenbereich

gespeichert. Gelingt es ihm weiters, den Rechner so zu manipulieren, dass die Befehle dieses Datenbereiches ausgeführt werden, war sein Angriff erfolgreich, da Befehle ausgeführt werden konnten, ohne dass es im Programm vorgesehen war. Solch ein Angriff wird Buffer Overflow genannt.

Damit die Befehle eines Angreifers ausgeführt werden können, muss es ihm gelingen, bestimmte Parameter des Programmes zu manipulieren.

Viele Programmiersprachen bieten die Möglichkeit an, so genannte Unterprogramme oder Funktionen zu verwenden. Dabei wird Programmcode gekapselt und kann an mehreren Stellen im Programm oder rekursiv ausgeführt werden. Bei der Ausführung einer Funktion werden die Befehle sowie die verwendeten Daten auf einen bestimmten Speicherbereich kopiert, auf dem die Funktion arbeiten kann. Damit nach der Abarbeitung der Funktion das Programm weiterarbeiten kann, wird die sogenannte Rücksprungadresse gespeichert, welche auf den nächsten Befehl nach der Funktion zeigt. Gelingt es einem Angreifer diese Rücksprungadresse mit einer modifizierten zu überschreiben hat er sein Ziel erreicht, da der Programmablauf geändert wurde. An der Stelle, auf die die Rücksprungadresse zeigt, befinden sich Befehle des Angreifers, die er vorher eingeschleust hat. Dies kann dadurch passieren, dass bei einer Eingabe von Daten nicht überprüft wird, wie viel Speicher die Eingabe benötigt und ob sie überhaupt in die vorgesehene Variable passt. Ist die Eingabe zu groß, wird der Speicherbereich nach der vorgesehenen Variable überschrieben. Hier müssen die restlichen Speicherbereiche so lange überschrieben werden, bis die Rücksprungadresse erreicht ist, welche mit einem gewünschten Wert überschrieben wird. Wenn nun die Funktion beendet wird, führt das Programm die Befehle des Angreifers aus.

Dies kann nur geschehen, wenn der Programmierer eine Funktion verwendet, welche die eingegebenen Daten nicht auf ihre Länge überprüft. Es gibt zum Beispiel in einigen Bibliotheken der Programmiersprache C solche Funktionen und es wird aus Sicherheitssicht davon abgeraten, diese zu verwenden.

Um sich vor einem Buffer Overflow zu schützen, kann zum Beispiel während der Laufzeit die Länge der eingegebenen Daten überprüft werden. Allerdings gibt es auch andere Methoden einen Buffer Overflow zu verhindern.

- Es wird vor jeder Rücksprungadresse ein zufällig erzeugter Wert gespei-

chert, welcher garantiert, dass die Adresse nicht manipuliert wurde. Wird dieser Wert nicht mehr vorgefunden, weil er durch einen Buffer Overflow überschrieben wurde, beendet sich das Programm und der Angriff kann nicht stattfinden [14].

- Wird der Speicherbereich strikt in Daten und Befehl Speicherbereiche unterteilt, können keine eingegebenen Daten ausgeführt werden. Dies kann durch die CPU erfolgen, welche ein Execute Disable Bit oder NoExecute (NX) Bit verwendet, um den Speicherbereich eindeutig zu markieren [75].

Diese Techniken stellen eine Möglichkeit dar, Buffer Overflows zu verhindern. Allerdings gibt es schon Methoden, diese zu überwinden. Eine dieser stellen Xu et. al. in [75] vor.

Eine praktische Einführung zu diesem Thema kann in [2] gefunden werden.

2.3.2 Code Injection

Bei Code Injection nutzt ein Angreifer einen Fehler in einem Programm aus, um ausführbaren Code einzuschleusen und diesen auszuführen. Bei einem Angriff über ein Netzwerk kann dieser Code Befehle enthalten, um weitere Programme nachzuladen, wie Ma et. al in [36] beschreiben.

2.4 Durchführung von Angriffen auf Clientsoftware

In den Studien [54], [47] und [50] wurden Methoden für client-seitige Angriffe vorgestellt. Für diese Angriffe gibt es verschiedene Vorgehensweisen. In diesem Kapitel werden diese näher beschrieben.

2.4.1 Zero-Day Attack

> A zero-day exploit is an exploit for a vulnerability that is unknown at that point - that is, a new und unreleased vulnerability. Consequently, there is usually no patch available for a zero-day vulnerability. This kind of attack is a severe threat, since this new attack vector cannot be mitigated efficiently [51, S. 253].

Bei einem zero-day Exploit gibt es noch keinen Sicherheitspatch, der eine Sicherheitslücke in einem Programm behebt. Wird ein solcher zero-day Exploit für einen Angriff verwendet, spricht man von einem zero-day Angriff. Diese sind besonders gefährlich für einen Hersteller, da dieser auf solche Angriffe nur schwer reagieren kann. Es muss erst die Schwachstelle gefunden werden, die ein solcher Angriff ausnutzt, um einen Sicherheitspatch schreiben zu können. Eine Möglichkeit zero-day Angriffe zu erkennen stellen high-interaction Honeyclients dar, welche in Kapitel 3.1 vorgestellt werden. In einer Studie von Microsoft [76] wird beschrieben, wie mit einem high-interaction Honeyclient ein noch unbekannter Angriff detektiert werden kann.

2.4.2 Driven by Download

Unter Driven by Download versteht man, dass Programme ohne das Wissen des Benutzers auf dessen Rechner gespeichert werden. Dabei handelt es sich meistens um Malware, welche nach einem erfolgreichen Angriff für weitere Aktivitäten verwendet wird.

Weiters gibt es den Begriff Driven by Install, welcher die unwissentliche Installation von Programmen auf einem System beschreibt.

In Kapitel 1 wurde auf mehrere Studien verwiesen, welche diese Möglichkeit der Durchführung eines Angriffes untersuchten.

2.4.3 Remote Code Execution

Dabei handelt es sich um die Möglichkeit eines Angreifers auf einem entfernten System nach einem Angriff Befehle auszuführen. Meistens bedient sich der Angreifer einer Schwachstelle, welche dieser über eine Netzwerkverbindung ausnutzen kann. Oftmals folgt darauf eine Driven by Download und beziehungsweise oder eine Driven by Install Aktion, damit spezielle Software für den Angreifer auf dem angegriffenen System zur Verfügung steht.

Das Gefährliche an dieser Methode ist, dass der Angreifer sie entfernt durchführen kann, also nicht direkt an dem jeweiligen System seine Eingaben tätigen muss. Da in der heutigen Zeit sehr viele Dienste über ein Netzwerk angeboten werden, hat auch ein Angreifer mehr Möglichkeiten in ein System einzudringen.

2.5 Verteilung von Malware

Um zu analysieren, wie bösartige Inhalte in Seiten gelangen, muss man die Komponenten einer Internetseite beachten. Heutzutage sind viele Seiten nicht mehr rein statisch, sondern verfügen auch über dynamische Inhalte. Viele Entwickler von Webseiten verwenden externe Programme, um zusätzliche Funktionen wie Besucherzähler oder Kalender in ihre Seiten einzubauen. Dazu müssen sie darauf vertrauen, sichere Inhalte einzubinden. Dies ist allerdings nicht immer garantiert.

In vielen Fällen werden Verweise zu Servern eingeschleust, die Skripte enthalten, bei deren Ausführung eine Schwachstelle im Browser ausgenutzt wird. Diese Art des Angriffes wird in Kapitel 4.1 genauer beschrieben.

Allgemein ist es bei client-seitigen Angriffen für den Angreifer wichtig, Inhalte in Internetseiten einzuschleusen, welche bei jedem Aufruf der Seite ausgeführt werden. Im Folgenden werden nun verschiedene Möglichkeiten aufgezählt, wie diese Inhalte in Internetseiten importiert werden können. Detailliertere Informationen können in [50] nachgelesen werden.

Webserver Sicherheit: Die Sicherheit von Inhalten einer Internetseite ist abhängig von der Sicherheit der Infrastruktur, auf der sie gespeichert ist. Es bieten jedoch nicht nur der Webserver, sondern auch verwendete Skiptsprachen (PHP, ASP) oder Komponenten, welche in diesen Sprachen programmiert wurden, wie Forensoftware, solche Einfallstore. Enthalten diese Sicherheitslücken, kann der Angreifer sie ausnutzen.

Benutzerdefinierte Inhalte: Heutzutage ist es dem Benutzer auf vielen Internetseiten möglich, den Inhalt mitzugestalten. Dies kann etwa durch Kommentare zu Artikeln oder durch Blogsysteme geschehen. Dadurch kann ein Angreifer Inhalte in die Seiten einschleusen, welche Sicherheitslücken im Browser ausnutzen. Diese werden beim Aufruf der Seite nachgeladen und ein Angriff findet statt.

Um dies zu verhindern, wird bei der Eingabe von Texten meistens nur ein kleiner Teil von Hypertext Markup Language (HTML) Befehlen weiter verarbeitet. Allerdings ist es manchmal möglich, die Überprüfung mit speziellen Eingaben zu überlisten, sodass alle Befehle verwendet werden können.

Oft werden Links zu Schadseiten nicht im Klartext gespeichert, sondern erst bei der Ausführung einer Javascript Funktion lesbar gemacht. Mit dieser Methode ist es möglich, Verweise auf bösartige Seiten zu tarnen.

Werbung: Die Mehrheit der Werbung im Internet wird gegenwärtig von speziellen Firmen zur Verfügung gestellt. Dazu wird ein Skript der Firma in die Seite eingebaut, welches die Werbung aufruft. Allerdings können über Werbung bösartige Inhalte importiert werden. Oft verkaufen Werbefirmen Anteile an Werbung weiter, wodurch immer unübersichtlicher wird, von welcher Quelle der Werbeinhalt tatsächlich bereit gestellt wird. Ein Angreifer kann solche Anteile kaufen und zusätzlich zur Werbung andere Inhalte in die Seiten einschleusen. So ist es möglich, viele Besucher zu erreichen und Schadprogramme zu verbreiten, ohne Inhalte direkt auf dem Webserver der Zielseite zu speichern.

Drittanbieter Programme: Werden Bibliotheken von Drittanbietern in eine Seite eingebunden, so können diese auch für die Verbreitung von Malware verantwortlich sein. Ändert der Anbieter die Funktion hinter der Bibliothek, ist es für den Betreiber der Seite schwierig, dies festzustellen. So können anstatt Besucherstatistiken, welche die eigentliche Funktion der Bibliothek sind oder waren, auch Schadprogramme verteilt werden.

Damit die eingeschleusten Befehle nicht sofort entdeckt werden, verwendet ein Angreifer oft unsichtbare Elemente wie IFrames. Dabei handelt es sich um ein HTML Konstrukt, welches erlaubt, beliebige Inhalte in eine Seite zu importieren. Manchmal wird eine Kaskade von Verweisen genutzt. Dabei wird vom ersten Server ein Skript heruntergeladen, welches eine Seite auf einem zweiten Server ausführt und von dort wieder auf einen anderen Server geführt wird. Erst nach einigen Stationen wird das endgültige Skript heruntergeladen, welches die Schwachstelle im Browser ausnutzt. Diese Methode ist in Abbildung 2.1 dargestellt und wird in [50] detaillierter beschrieben.

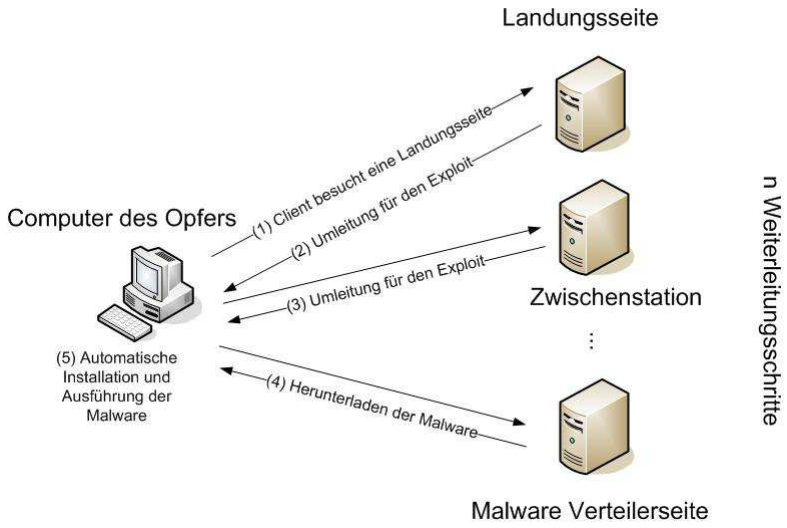

Abbildung 2.1: Verteilung von Malware mit Zwischenstufen: Vom ersten Server wird man auf den zweiten Server weitergeleitet und erst nach einigen Schritten wird die Malware von einem Rechner heruntergeladen. (nach [50])

2.6 Virtualisierungssoftware

> Eine virtuelle Maschine ist in der Lage verschiedene Prozesse oder ein gesamtes System unterstützen.[...] Wird ein System oder eine Komponente, wie etwa ein Prozessor, Speicher oder ein I/O Gerät, virtualisiert, kann die jeweilige Ressource der entsprechenden auf dem realen Computer zugeordnet werden. [59].

In diesem Kapitel wird die Virtualisierung auf Softwareebene vorgestellt. Es besteht auch noch die Möglichkeit der Virtualisierung auf Hardwareseite, welche aber in dieser Arbeit nicht benötigt wird.

Eine virtuelle Maschine ist ein Emulator, welcher softwaretechnisch die gesamte Architektur eine Systems emuliert. Es werden alle Geräte eines Systems emuliert und in einer virtuellen Umgebung zur Verfügung gestellt. In dieser läuft die zu virtualisierende Software oder ein gesamtes Betriebssystem, wie in [79] beschrieben. Eine virtuelle Maschine läuft getrennt von der physikalischen Hardware und es werden alle Befehle der virtuellen Maschine für die tatsächlichen Komponenten umgesetzt.

Der Kern für virtuelle Maschinen ist die Virtualisierungssoftware. Diese verwaltet die virtuellen Maschinen und die Ansteuerung der Hardware, welche auch als Virtual Machine Monitor (VMM) bezeichnet wird.

Bei der Virtualisierung unterscheidet man einerseits das Host System, auf dem die Virtualisierungssoftware ausgeführt wird. Andererseits gibt es die verwendeten virtuellen Maschinen, welche Gast Systeme genannt werden. Das Zusammenspiel von Host System, Virtualisierungssoftware und Gast System ist in Abbildung 2.2 dargestellt.

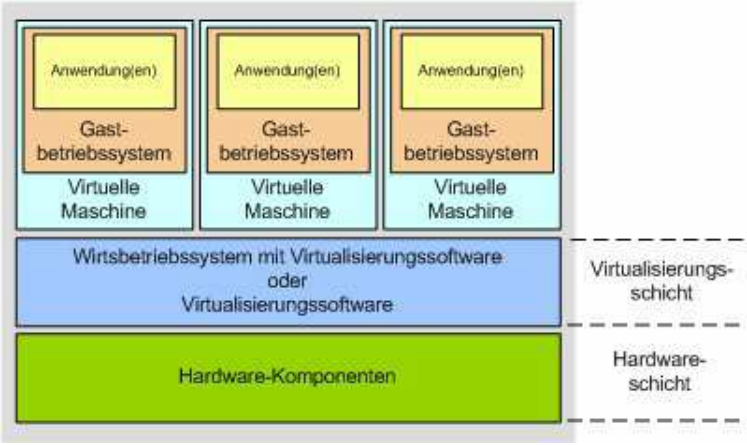

Abbildung 2.2: Das Hostsystem und mehrere Gast Systeme. Die Virtualisierungssoftware ist als Layer dazwischen. [79].

Es gibt verschiedene Möglichkeiten der Virtualisierung mit virtuellen Maschinen:

Native Virtualisierung: Es werden nur Teilbereiche der physischen Hardware dem Gast System zur Verfügung gestellt. Diese reichen allerdings aus, damit das Betriebssystem ausgeführt werden kann. Es muss allerdings das Gast System denselben Central Processing Unit (CPU) Typ aufweisen wie das Host System [79]. Beispiele für diesen Typ sind VMWare [104] oder Microsoft Virtual PC [91].

Paravirtualisierung: Bei dieser Art wird keine Hardware emuliert, sondern das virtuell gestartete Betriebssystem verwendet eine abstrakte Verwaltungsschicht, um auf Ressourcen des Host System zugreifen zu können. Bei dieser Art muss

allerdings das Betriebssystem portiert werden, um auf der virtuellen Maschine zu starten. Allerdings ist die Leistung bei dieser Methode der virtuellen Maschine höher. Ein Beispiel hierfür ist Xen 3.0 [19].

In dieser Arbeit wird die Native Virtualisierung von Honeyclients, welche in Kapitel 4 beschrieben werden, genutzt.

3 Honeypots und Honeynets

> "An approach to learn more about attacks and attack patterns is based on the idea of electronic decoys, called honeypots. A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource. It allows us to learn more about attacks in communication networks."[26]

Honeypots imitieren meistens ein Produktivsystem, um sich dadurch für einen Angreifer interessant zu machen. Sie enthalten allerdings keine wertvollen oder sensiblen Informationen, wodurch ein Angriff in Hinsicht darauf wenig Schaden anrichten kann. Es muss jedoch darauf hingewiesen werden, dass das System dennoch für andere Angriffe genutzt werden kann. Da der Honeypot kein Produktivsystem ist, gilt jeder Zugriff als potenzieller Angriff, wodurch die Rate der false positives gleich Null ist.

Honeypots werden allgemein in zwei Arten unterteilt: low-interaction und high-interaction Honeypots.

3.1 High-interaction Honeypots

Ein high-interaction Honeypot ist ein vollwertiges, voll funktionsfähiges System mit Daten, welche einen Angreifer zu Attacken verführen soll. Da er wie ein Produktivsystem aufgebaut ist, kann er auch als solches angegriffen werden. Allerdings werden alle Aktionen des Angreifers genau mitprotokolliert und dadurch ist es möglich, im Nachhinein alle Phasen des Angriffs zu analysieren. So kann man herausfinden, wie das System gefunden wurde, welche Schwachstellen der Angreifer ausgenutzt hat und welche Programme er verwendet hat. Es ist auch möglich

daraus Informationen zu beziehen, wie der Angreifer versucht hat, den Angriff zu verbergen, damit dieser nicht bemerkt wird.

Natürlich möchte ein Angreifer nicht entdeckt und überwacht werden, da dadurch Informationen über seine Angriffsmuster bekannt werden könnten. Zum Beispiel kann ein System über einen zero-day Angriff angegriffen werden. Wenn diese Sicherheitslücke bekannt wird, verliert das Wissen darüber viel an Wert, sobald ein Patch dafür vorhanden ist. Es gibt verschiedene Möglichkeiten für einen Angreifer herauszufinden, ob er sich auf einem Honeypot befindet, welche in [24] dargestellt werden.

Da es sich um ein vollwertiges System handelt, kann der Angreifer dieses, sobald er es unter seiner Kontrolle hat, auch als solches nutzen. Das bringt ein großes Risiko mit sich. So kann der Angreifer versuchen, andere Rechner im Netzwerk oder im Internet anzugreifen.

Es gibt Möglichkeiten auf einem System alle Inhalte mitzuprotokollieren. Unter anderem hat das Honeynet Project [85] das Programm Sebek [61] veröffentlicht, welches diese Aufgabe übernimmt. Bei Sebek handelt es sich um ein Kernel basiertes Rootkit, bei dem bestimmte System Calls mit eigenen Funktionen überschrieben werden, um so die ausgeführten Aktionen protokollieren zu können. Dadurch ist es auch möglich, eigentlich als abhörsicher geltende Kommunikationswege, etwa Secure Shell (SSH), genau zu protokollieren. Diese können mit einem Netzwerksniffer nicht abgehört werden, da die einzelnen Pakete verschlüsselt sind. Das System ist als Client-Server System aufgebaut, wobei der Client auf dem Honeypot läuft, der Server auf einem externen System, welches alle Daten mittels User Datagram Protocol (UDP) Paketen zugeschickt bekommt. Auch diese Kommunikation wird vom Kernel versteckt, damit der Angreifer sie nicht erkennt. Allerdings ist es für einen erfahrenen Angreifer möglich, Sebek, zum Beispiel durch Analyse von Antwortzeiten bei bestimmen Netzwerkpaketen, auf einem System zu erkennen und die Protokollierung abzuschalten [18]. Um das Programm zu deaktivieren müssen nur die ursprünglichen System Calls wiederhergestellt werden.

3.2 Low-interaction Honeypots

Low-interaction Honeypots sind im Gegensatz zu high-interaction Honeypots nur simulierte Systeme. Alle Services auf dem System sind nur emuliert und werden von bestimmten Programmen gesteuert. Dabei werden meist nur eine begrenzte Anzahl an Transmission Control Protocol (TCP) oder UDP Diensten emuliert und bei diesen auch nicht alle Befehle.

Das bedeutet zum Beispiel ein emulierter File Transfer Protocol (FTP) Server läuft auf Port 21 und kann nur FTP Logins behandeln; alle anderen Befehle des Protokolls sind nicht implementiert. Es können jedoch weitere Befehle des FTP Protokolls implementiert werden, um mehrere Arten von Angriffen zu ermöglichen.

Diese sind leicht auf einem System zu installieren und bieten mehr Sicherheit als bei einem high-interaction Honeypot.

Allerdings ist es durch die emulierten Services nicht möglich, zero-day Attacken zu erkennen und ein erfahrener Angreifer wird bald erkennen, dass er sich auf einem Honeypot befindet. Ein weiterer Nachteil ist, dass der Angreifer alle Rechte erlangen kann, falls es ihm trotz der emulierten Services gelingt, das System entsprechend auszunutzen.

Eine Art von low-interaction Honeypots sind Systeme, welche darauf hin entwickelt wurden Malware, die sich über das Internet verbreitet, zu sammeln und zu analysieren. Diese werden installiert, um neue Signaturen für Antiviren Programme oder IDS zu erstellen. Weiters kann man dadurch auch die Ausbreitung von Würmern erforschen, um möglichst rasch auf solche Situationen reagieren zu können. Es ist auch möglich, dass mehrere solcher Honeypots ein Netzwerk schützen, indem sie die Weiterverbreitung von Würmern verlangsamen. Dies wird dadurch möglich, dass die Honeypots zum Beispiel wesentlich langsamer auf Pakete antworten als normale Systeme. Mit den gewonnenen Informationen aus den Honeypots können neue Signaturen für die IDS erstellt werden. Dadurch können noch nicht befallene Systeme vor dem Angriff beschützt werden. Beispiele für solche Honeypots sind mwcollect [25], [31] und nepenthes [4]. Diese wurden so implementiert, dass es möglich ist, mehrere Tausend Instanzen auf einem einzigen Rechner zu installieren und auszuführen.

Die Möglichkeit mit einem high-interaction Honeypot, selbst verteilende Mal-

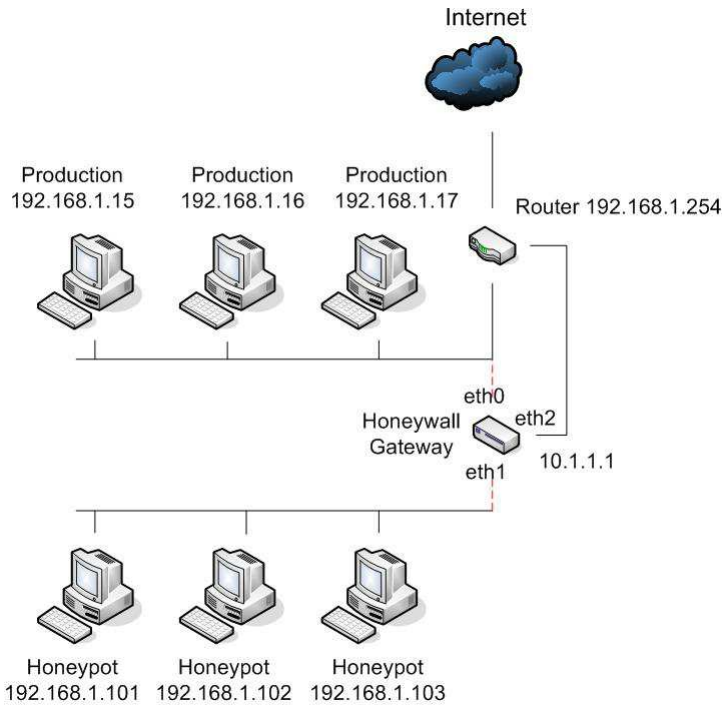

Abbildung 3.1: Die Architektur eines Honeynets. Die einzelnen Honeypots sind in einem extra Netzwerk angeordnet und über die Honeywall mit dem Internet verbunden (nach [63]).

ware zu analysieren und zu sammeln, wird in [78] vorgestellt.

3.3 Honeynet

Ein Honeynet ist ein Netzwerk, in dem nur high-interaction Honeypots existieren. Da Honeypots keine Produktivsysteme sind, hat auch das Honeynet keine produktiven Aktivitäten [63]. In einem normalen Netzwerk müssen die gesammelten Informationen von IDS und Firewall analysiert werden, um einen Angriff zu entdecken, in einem Honeynet ist jeder Zugriff ein potenzieller Angriff.

3.3.1 Architektur eines Honeynets

Eine mögliche Architektur eines Honeynets, welche vom Honeynet Project entworfen wurde, ist in Abbildung 3.1 dargestellt [63]. Die Honeypots sind in einem

eigenen Netzwerk und über die so genannte Honeywall mit dem Internet verbunden. Jede Verbindung von oder zu den Honeypots erfolgt über die Honeywall, welche normalerweise eine Layer 2 Bridge ist, die daher für den restlichen Netzwerkverkehr nicht sichtbar ist. Die Honeywall hat grundsätzlich die Funktion, die Daten des Netzwerkverkehrs im Honeynet mitzuprotokollieren und zu speichern. Diese Aufgaben sind in folgende Punkte unterteilt:

- Data Control bedeutet, dass die Aktivitäten eines Angreifers auf dem System kontrolliert werden können. Wenn ein Angriff stattfindet, werden Informationen mit diesem ausgetauscht, welche sowohl eingangs- als auch ausgangsseitig kontrolliert werden müssen. So lädt ein Angreifer oft eigene Programme aus dem Internet auf den Computer oder versucht, weitere Systeme anzugreifen. Diese Verbindungen werden von der Honeywall kontrolliert und entweder ganz blockiert oder nur teilweise zugelassen. Damit der Angreifer keine DDoS Angriffe von dem Honeypot aus durchführen kann, kann man mit der Honeywall die Anzahl der ausgehenden TCP Verbindungen pro Zeiteinheit limitieren. Es können auch UDP und TCP Pakete auf diese Weise gefiltert werden. Eine weitere Möglichkeit bietet die Honeywall, um bestimmte Angriffe von dem Honeypot aus zu verhindern, indem zum Beispiel bei ausgehenden Verbindungen der Dateninhalt der Pakete so verändert wird, dass der Angriff fehlschlägt. Dieser Vorgang wird "extrusion prevention" genannt, welcher durch modifizierte Snort Regeln implementiert ist [62].

- Data Capture bedeutet, dass alle Aktivitäten auf dem Honeypot und alle Verbindungen von und zu dem Rechner sorgfältig mitprotokolliert werden. Allerdings muss dies so geschehen, dass der Angreifer davon nichts bemerkt.

- Data Analysis bedeutet, dass die gesammelten Daten in verwertbare Informationen umgewandelt werden. Aus diesen Informationen können Entscheidungen für die Zukunft getroffen werden. So können zum Beispiel die Regeln der Firewall angepasst werden. Da jedes Unternehmen verschiedene Bedürfnisse hat, sind auch die Analysemethoden unterschiedlich.

Die Honeywall kann als komplettes ISO Image vom Honeynet Project heruntergeladen und auf einem Rechner installiert werden.

3.3.2 Virtuelles Honeynet

Eine weitere Ausführung eines Honeynets ist das virtuelle Honeynet. Dabei werden alle Honeypots des Honeynets auf einem Computer ausgeführt. Dies ist mittels Virtualisierungssoftware möglich, da diese erlaubt, dass mehrere Betriebssysteme auf einem Rechner gleichzeitig ausgeführt werden [51, S. 53]. Dazu sind als Virtualisierungssoftware sowohl VMWare [104] als auch UML [96] geeignet. So ist es zum Beispiel möglich, ein gesamtes Honeynet auf einem Laptop zu installieren und diesen überall mitzunehmen.

Allerdings hat ein virtuelles Honeynet auch einige Nachteile. So können nur Betriebssysteme installiert werden, welche die Virtualisierungssoftware und die CPU Architektur unterstützten. Weiters könnte es dem Angreifer gelingen, Kontrolle über die Virtualisierungssoftware zu erlangen und dadurch die einzelnen Honeypots zu übernehmen. Bemerkt der Angreifer, dass er sich auf einem virtuellem System befindet, kann dies für ihn ein Indiz sein, dass er sich auf einem Honeypot befindet. Allerdings werden heutzutage auch viele Produktivsysteme virtuell betrieben, um zum Beispiel Kosten zu sparen. Eine Möglichkeit festzustellen, ob ein System virtuell ist oder nicht, wird von Carpenter et. al. in [10] vorgestellt.

Virtuelle Honeynets können in zwei Gruppen unterschieden werden. Bei einem self contained virtual Honeynet befindet sich auch die Honeywall virtuell auf dem System. Bei einem hybrid virtuellen Honeynet sind die einzelnen Honeypots virtuell, die Honeywall läuft aber auf einem eigenen System. Jede Variante hat einige Vor- und Nachteile, welche in [60] aufgeführt sind.

4 Honeyclients

In diesem Kapitel werden die grundlegenden Ideen und Eigenschaften eines Honeyclients präsentiert und es wird auf die zur Zeit aktuellen Umsetzungen eingegangen.

4.1 Einführung

In letzter Zeit hat sich immer mehr gezeigt, dass neue Angriffsmuster gegen Computer auftreten. So kann man zwischen server-seitigen Angriffen und client-seitigen Angriffen unterscheiden.

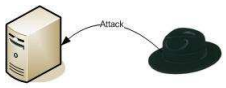

Abbildung 4.1: Angriff, bei dem ein Server attackiert wird [57].

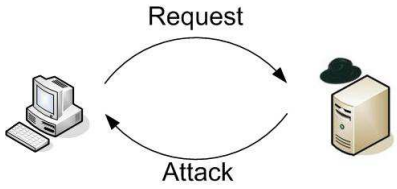

Abbildung 4.2: Ein client-seitiger Angriff. Der Client fordert Informationen von einem Server an und in der Antwort ist ein Angriff enthalten, der eine Sicherheitslücke in der Client Software ausnutzt.

Bei einem server-seitigen Angriff wird ein Fehler in einem Service ausgenutzt, welches entfernt auf einem Server ausgeführt wird. Jeder Server im Netzwerk oder Internet stellt Dienste, so genannte Services, zur Verfügung. Auf die Dienste können Benutzer mittels eines Clients zugreifen und Informationen austauschen.

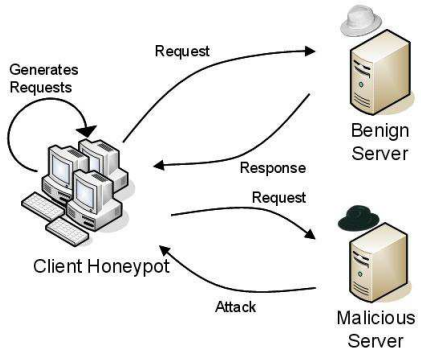

Abbildung 4.3: Die schematische Darstellung eines Honeyclients. Dieser erzeugt Anfragen an Server im Netzwerk und analysiert die Antworten. Die erhaltenen Antworten können Angriffe enthalten [57].

Jedes Service kann Fehler enthalten, die ein Angreifer ausnutzen kann. Da viele Dienste permanent von Benutzern abgefragt werden können, kann auch ein Angreifer diese jederzeit für seine Attacken nutzen. Er verbindet sich zu dem Server und schickt Pakete mit einem Inhalt ab, welche eine Sicherheitslücke des angebotenen Dienstes ausnutzen, um weiteren Zugriff auf den Server zu erhalten. Diese Methode des Angriffes ist zum Beispiel in Abbildung 4.1 dargestellt. War der Einbruch erfolgreich, kann der Angreifer nun Malware auf dem Server installieren und seine Spuren des Angriffes verstecken. Diese Methoden werden genauer in [58] vorgestellt.

Im Gegensatz dazu gibt es client-seitige Angriffe. Bei diesen Angriffen wird versucht, eine Schwachstelle in einer Client Applikation auszunutzen. Dazu muss sich allerdings erst der Client auf einen Server verbinden, welcher in den Antwortdaten die Sicherheitslücke ausnutzen kann. Stellt der Client keine Verbindung zu einem solchen Server her, kann dieser nicht angegriffen werden. Dieser Angriff ist in Abbildung 4.2 dargestellt.

Da es eine viel größere Anzahl an Clients gibt als Server, ist es für einen Angreifer interessant, diese anzugreifen. Die möglichen Ziele für einen Angriff können sein.

- Der Angreifer benötigt viele Clients für ein so genanntes Botnet. Ein Botnet ist ein Netzwerk von Computern, welche von einer zentralen Stelle aus gesteuert werden können, um Befehle entgegenzunehmen [21]. Allerdings geschieht dies meistens ohne das Wissen des Besitzers des Rechners. Über einen

Angriff wird ein Trojaner auf dem Rechner installiert, welcher sich mit dem Botnet verbindet und auf Befehle wartet. Diese Befehle können ein gemeinsamer Angriff aller Rechner des Botnets auf einen einzigen Server sein, ein so genannter DDoS Angriff, welcher den Server lahm legt und weitere Angriffe zulässt.

- Der Angreifer benötigt Rechner, welche als Mailverteiler für Spam Mails fungieren. So wird auf dem Rechner mit Hilfe des Angriffes ein Mailserver installiert, welcher Emails vom Angreifer aus verschickt. Dadurch besteht keine direkte Verbindung mehr zum Angreifer und dieser kann schwieriger zurückverfolgt werden.

- Gelingt es dem Angreifer jeden Tastenanschlag auf einem Computer mitzuprotokollieren, kann er sensible Informationen wie Kontodaten oder Anmeldedaten von Paypal oder eBay aufzeichnen.

- Installiert der Angreifer auf einem Rechner ein Programm, welches Werbung im Internet regelmäßig aufruft, bekommt er vom Werbebetreiber eine bestimmte Provision. Wird diese Anzeige von vielen manipulierten Computern aufgerufen, vervielfacht sich auch das Einkommen des Angreifers.

  Die Werbung muß ein Benutzer gar nicht tatsächlich zu sehen bekommen. Dies wäre ein Indiz dafür, dass sich Malware auf dem Rechner befindet.

- Der Angreifer kann gezielt Onlinebefragungen manipulieren, indem viele übernommene Rechner seine Auswahl treffen. Dadurch sind solche Befragungen nicht mehr repräsentativ. Dies wird aber nicht erkannt, da jede Stimme von einem scheinbar unabhängigen Rechner vergeben wird.

Um auf solche Angriffe zu reagieren, benötigt man neue Hilfsmittel. Der klassische Honeypot ist dafür nicht geeignet, da er passiv auf einen Angriff wartet und nicht von sich aus nach gefährlichen Quellen sucht. 2004 entstanden so genannte Client Honeypots oder Honeyclients, die solche Angriffe erkennen sollen. Ein Client Honeypot untersucht entfernte Ressourcen, ob sie auf einem Client für den Benutzer ungewollte Änderungen vornehmen. Dieser Vorgang ist in Abbil-

dung 4.3 dargestellt. Jeder Honeyclient besteht aus folgenden drei grundlegenden Elementen:

Queuer: Dieser Teil sucht die zu untersuchenden Inhalte des Internets. Da heutzutage mit Honeyclients meistens Internetseiten untersucht werden, kann dies eine Liste von Internetadressen sein. Eine weitere Möglichkeit ist eine Suchanfrage an eine Suchmaschine zu stellen, wobei in der Ergebnisliste jede Adresse überprüft wird.

Visitor: Der Visitor ruft das entfernte Objekt auf, indem er eine Anfrage an den entsprechenden Server abschickt. Dieser sendet normalerweise eine Antwort zurück, welche der Honeyclient entgegennimmt.

Analysis Engine: Dieser Teil des Honeyclient untersucht die Antwort oder das System, nachdem die Antwort erhalten wurde. Da man Ressourcen sucht, welche den Systemzustand ändern (Sicherheitslücken in der Client Software ausnutzen, um neue Dateien zu erstellen, Dateien zu löschen, Prozesse anzulegen etc.), kann diese Änderung nur nach Erhalt der Antwort erfolgt sein.

Bei der technischen Implementierung von Honeyclients gibt es, analog zu den bereits beschriebenen Honeypots, zwei verschiedene Arten:

Low-interaction: Ein low-interaction Honeyclient ist ein System, bei dem die Client Software emuliert wird und die Antwort des Servers mittels Signaturen analysiert wird. Dabei können natürlich nur Angriffe entdeckt werden, welche bereits bekannt sind und für die es Signaturen gibt; zero-day Angriffe werden nicht entdeckt. Auf der anderen Seite sind diese Clients sehr schnell und können daher sehr viele Adressen überprüfen. Außerdem sind sie nicht so komplex aufgebaut wie high-interaction Honeypots und leichter zu installieren sowie zu administrieren.

High-interaction: Bei diesem System handelt es sich um ein vollwertiges Betriebssystem mit Client Software, welches gesteuert wird, um entfernte Objekte zu untersuchen. Dieser Typ von Honeyclient kann auch Angriffe erkennen,

welche noch unbekannt sind und kann daher auch unter anderem für die Erstellung von Patches für Client Software benutzt werden. Allerdings ist diese Art wesentlich komplexer zu erstellen und zu administrieren, da ein gesamtes Betriebssystem dahinter steckt. Auch ist dieser Typ meistens langsamer als low-interaction Honeyclients, da das ganzes System nach Anomalien untersucht werden muss.

4.2 Low-interaction Honeyclients

Nachfolgend werden unterschiedliche low-interaction Honeyclients vorgestellt. Dabei werden auch Aktualisierungen von bestehenden Honeyclients erörtert und es wird auf die verschiedenen Implementierungen eingegangen.

4.2.1 Monkey Spider

Monkey Spider ist ein low-interaction Client Honeypot, der von Ali Ikinci an der Universität von Mannheim entwickelt wurde [27]. Die grundlegende Architektur ist modular aufgebaut und kann durch neue Module erweitert werden. Das Verbindungsstück zwischen den einzelnen Modulen ist in Python geschrieben und kann auf Grund dieser Tatsache auf verschiedenen Plattformen ausgeführt werden. Für die einzelnen Module wurden großteils frei erhältliche Programme verwendet. Für die Suche nach Adressen bietet der Honeyclient drei Möglichkeiten an:

- Es können Adressen über eine Suche mit Google oder Yahoo ermittelt werden. Dabei werden verschiedene Stichwörter benutzt und aus dem Ergebnis der Suchmaschine werden die einzelnen Links extrahiert.
- Es können Adressen von so genannten Blacklists verwendet werden. Dabei handelt es sich um Listen, welche auf Stellen im Internet hinweisen, die man nicht besuchen sollte, da diese schadhafte Inhalte aufweisen können. Diese Seiten enthalten mit hoher Wahrscheinlichkeit Malware.
- Es kann auch ein Email Account eingerichtet werden, welcher Spam Mails sammelt und in diesem können die enthaltenen Internetadressen verwendet werden.

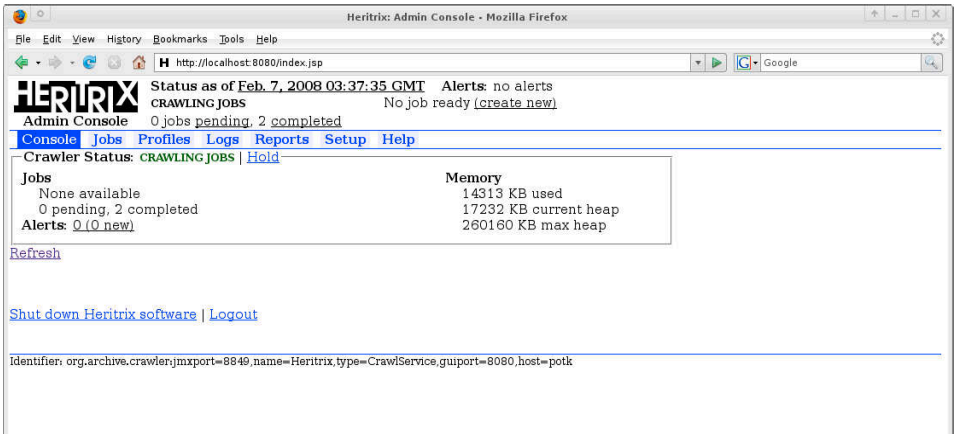

Abbildung 4.4: Das Webinterface des Heritrix Crawlers

Um die einzelnen Internetadressen herunterzuladen, wird der Heritrix Crawler [84] verwendet, welcher vom Internet Archive [87] entwickelt und eingesetzt wird. Das Internet Archive ist eine Non Profit Organisation, die versucht, eine historische Bestandsaufnahme des Internets zu erzeugen. Es wird dabei das ARC Dateiformat verwendet, um ganze Internetseiten performant und kompakt zu speichern. Heritrix kann auch verlinkte Seiten herunterladen, welche der Honeyclient später analysiert. Der Crawler wird über ein Webinterface gesteuert, welches in Abbildung 4.4 dargestellt ist.

Die einzelnen Internetseiten werden im nächsten Schritt aus dem ARC Format extrahiert und mittels ClamAV [83], einem kostenlosen Antiviren und Malwarescanner, auf Schadsoftware überprüft. Die dabei ermittelten Informationen werden im Dateisystem und in einer Datenbank abgelegt. Für zukünftige Versionen von Monkey Spider ist geplant, dass andere Scanner für die Überprüfung genutzt werden können.

4.2.2 HoneyC

HoneyC ist ein low-interaction Honeypot, der an der Universität von Wellington entwickelt wurde [56].

Das Zusammenspiel der einzelnen Komponenten des Honeyclients ist in Abbildung 4.5 zu sehen. Der erste Teil stellt die einzelnen Adressen für den Client zur

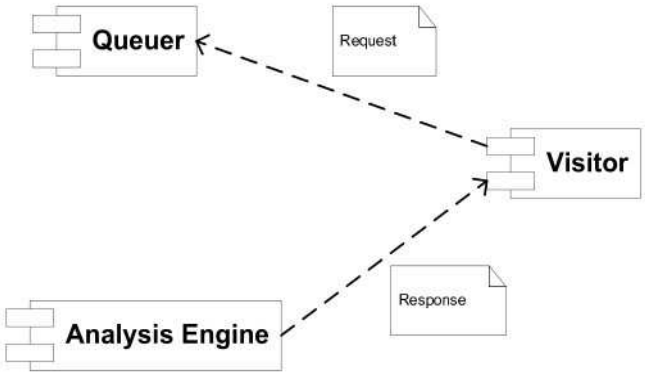

Abbildung 4.5: Die Komponenten des HoneyC. Der Informationsfluss zwischen den einzelnen Komponenten ist folgender: Der Queuer erzeugt Requests für den Visitor, welche der Visitor ausführt und die erhaltenen Informationen an die Analysis Engine weiterleitet [56].

Verfügung. In Version 1.0.0 gibt es dafür ein Modul, das die Yahoo Search API [99] verwendet, um Adressen aus der Suchmaschine Yahoo auszulesen. Ab der Version 1.1.2 kann man auch eine Liste von Adressen an den Client Honeypot übergeben.

Diese Liste wird an den Visitor weiter gegeben, welcher die Adressen besucht und die Antwortdaten lokal speichert. Dabei simuliert er einen normalen Internetbrowser, wobei die Browserkennung eingestellt werden kann sowie welche Teile einer Internetseite, wie etwa Bilder, heruntergeladen werden sollen.

Die Antwort wird mit der Analysis Engine bearbeitet, welche anhand von Snort Regeln [92] versucht, zu erkennen, ob es sich um eine bösartige Seite handelt oder nicht. Die einzelnen Programmteile werden von einem Hauptprogramm sequenziell aufgerufen. Es können eigene Routinen in den Ablauf eingefügt werden, um die Funktionen von HoneyC zu erweitern.

Damit die Geschwindigkeit des Clients erhöht wird, werden Threads verwendet.

4.2.3 SpyBye

SpyBye ist ein low-interaction Honeypot, welcher von Niels Provos implementiert wurde. Grundsätzlich ist dieser für Administratoren von Internetportalen entwickelt worden, um damit Internetseiten überprüfen zu können. Dabei wird untersucht, ob versteckte Links zu bösartigen Seiten vorkommen.

Für die Erkennung verwendet dieser Honeyclient verschiedene Regeln. Anhand dieser Regeln werden die Adressen aller Ressourcen auf einer Seite überprüft und klassifiziert. Dabei gibt es drei verschiedene Status:

harmless: Der Inhalt der Seite stimmt mit keiner Signatur des Honeyclients überein.

unknown: Adressen, die auf der Seite verlinkt sind, können gefährliche Inhalte enthalten.

dangerous: Diese Seite enthält gefährliche Inhalte. Der Administrator der Seite soll überprüfen, wie diese Inhalte in die Seite integriert werden konnten. Möglicherweise sind nicht die letzten Sicherheitsupdates installiert und es soll überprüft werden, ob der Angreifer nicht Backdoors auf dem Server installiert hat.

Der Client Honeypot wird als Proxy eingesetzt, um die Analyse der Seiten im Browser bereitzustellen. Ist im Browser der Proxyserver von SpyBye eingetragen, erscheint bei jedem Seitenaufruf ein Fenster, welches Informationen über den Status der Seite ausgibt. Dabei wird jeder Verweis auf eine Ressource mit dem jeweiligen Status angezeigt. Alternativ kann ein Modus verwendet werden, welcher nur bei einer bösartigen Seite eine Meldung an den Benutzer gibt. Die verschiedenen Möglichkeiten für die Anzeige des Analyseergebnisses sind in Abbildung 4.6 dargestellt.

4.3 High-interaction Honeyclients

In diesem Kapitel werden unterschiedliche high-interaction Honeyclients vorgestellt. Weiters wird auf die verschiedenen Methoden der Implementierungen eingegangen.

4.3.1 Honeyclient

Honeyclient war das erste Programm, welches die Idee eines high-interaction Honeyclients implementierte. Der Honeyclient wurde 2004 von Kathy Wang erstellt

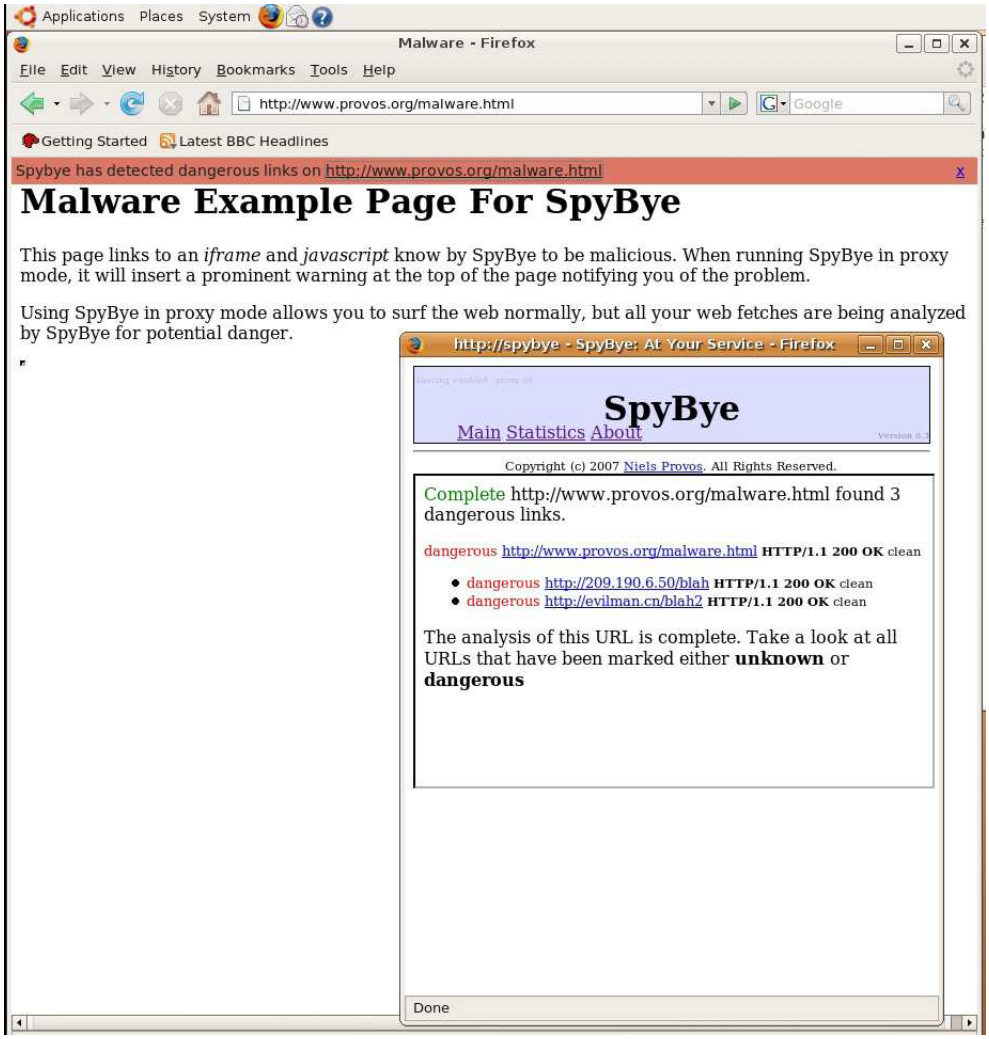

Abbildung 4.6: Die Analyse von SpyBye im Browserfenster. Der Status der einzelnen Links wird über dem Seiteninhalt angezeigt (kleines Fenster). Alternativ kann auch nur eine Meldung bei bösartigen Seiten angezeigt werden (großes Fenster) [93].

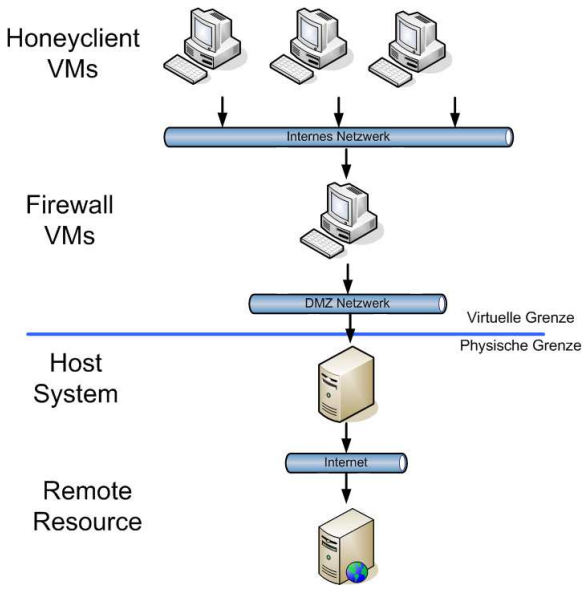

Abbildung 4.7: Der Informationsfluss des Honeyclients über die verschiedenen Komponenten des Systems. Ausgehend von den Honeyclients werden Anfragen an ein Netzwerk über die Firewall an das Host System gesendet. Dieses schickt sie weiter an die entsprechende entfernte Ressource. In der Abbildung erkennt man auch die Grenze zwischen virtuellen Systemen und physikalischen Rechnern. (Abb. aus [102])

und ist OpenSource. Die ursprüngliche Version war ein Perl Programm, welches in einer virtuellen Maschine das Dateisystem und die Registry auf Änderungen untersucht hat, nachdem der Internet Explorer eine Seite aufgerufen hatte.

Mittlerweile ist das Projekt in das MITRE Honeyclient Project umbenannt worden und besteht nun aus einem wesentlich komplexeren, weiterentwickelten Client. Es wird weiterhin in Perl programmiert, besteht aber nun aus folgenden Teilen (siehe Abbildung 4.7):

- Honeyclient VM ist eine virtuelle Maschine, die eine Applikation ausführt und eine Seite aus dem Internet anfordert. Danach wird die Integrität des Systems überprüft. Als Betriebssystem wird zur Zeit Microsoft Windows XP verwendet.

- Firewall VM ist eine virtuelle Maschine, welche den Datenverkehr zwischen

Honeyclient Netzwerk und dem Demilitarized Zone (DMZ) Netzwerk filtert. Wird einem Client der Zugriff auf eine entfernte Ressource ermöglicht, werden auch alle damit in Zusammenhang stehenden Ressourcen zugelassen. Als Firewall dient die Honeywall CD Rom Roo [86].

- Das Host System ist ein Linux basierter Server, der jeden Honeyclient verwaltet und die Verbindung zwischen DMZ Netzwerk und Internet herstellt. Bei diesem Systemaufbau gibt es zwei verschiedene Netzwerke. Einerseits das interne Netzwerk, welches den Verkehr zwischen Honeyclients und Firewall abbildet, andererseits das DMZ Netzwerk, welches den Verkehr zwischen DMZ und Internet darstellt.

4.3.2 Strider HoneyMonkey

Strider HoneyMonkey ist ein high-interaction Honeyclient und wurde 2005 von Microsoft Research entwickelt [76]. Für die Analyse werden mehrere virtuelle Maschinen verwendet, welche Windows XP in unterschiedlichen Patchstadien enthalten. Jede einzelne Maschine ist mit dem Strider Flight Data Recoder [71], welcher jeden Dateizugriff und Registryzugriff überwacht, dem Strider Gatekeeper, welcher Autostart Änderungen erkennt und dem Strider Ghost Buster, der Malware erkennt, ausgestattet.

Am Anfang steht eine vollkommen ungepatchte Windows Installation (State 1), bei der gleichzeitig mehrere Internetadressen überprüft werden. Wenn eine Anomalie aufgetreten ist, wird jede Adresse einzeln überprüft, um zu identifizieren, welche Adresse für den Angriff verantwortlich war. Wurde eine Adresse mit einem Angriff auf das System erkannt, wird diese an State 2 weitergegeben, welcher eine teilweise gepatchte Windows XP Maschine ist. Diese ist auf den Stand Anfang 2005 gepacht und soll überprüfen, welche Bedrohungen für Benutzer bestehen, die die Software auf ihrem Computer nur gelegentlich aktualisieren.

Wenn in State 2 ein Angriff erkannt wurde, wird die Adresse zum Schluss mit State 3 überprüft, welcher ein vollständig gepatchtes Windows XP enthält. Sollte auf dieser Maschine ein Einbruch erfolgreich sein, kann es sich dabei um einen zero-day Exploit handeln oder eine bereits bekannte Sicherheitslücke, zu der es noch keinen Patch gibt. Die einzelnen virtuellen Maschinen werden von einer zentralen

Kontrolleinheit gestartet und mit den jeweiligen Adressen versorgt.

Im Mai / Juni 2005 konnten die Wissenschaftler mit dieser Variante eines Honeyclients einen zero-day Exploit erkennen und nachweisen, dass diese Technologie auch für unbekannte Angriffe eingesetzt werden kann [76]. Bei ihren Testläufen lässt sich eindeutig feststellen, dass mit jedem State weniger Angriffe erfolgreich sind.

4.3.3 McAfee SiteAdvisor

McAfee SiteAdvisor ist ein high-interaction Honeyclient, der seine Ergebnisse im Internet publiziert und dadurch seine Resultate jedem Benutzer zur Verfügung stellt.

Site Advisor wurde 2005 von einer Gruppe des MIT gegründet. 2006 wurde das Projekt von McAfee gekauft und wird dort nun weiterentwickelt.

Genauere Informationen über die verwendete Technik des Honeyclients werden vom Betreiber nicht publiziert, nur in dem Blog der Seite wird darauf hingewiesen, dass es sich um einen Crawler basierten high-interaction Honeyclient handelt. Dabei werden Seiten nicht nur auf Exploits untersucht, sondern auch auf Spam Mails bei Eingabeformularen, auf PopUp Fenster und wie die Seite mit anderen geprüften Seiten zusammenhängt. Das Ergebnis wird im Internet publiziert und kann kommentiert sowie angezweifelt werden (zum Beispiel bei Sicherheitsseiten). Jede geprüfte Seite erhält eines von vier Zeichen:

- Ein grünes bedeutet, dass bei dem Test keine signifikanten Probleme aufgetreten sind.
- Ein gelbes Zeichen bedeutet, dass entweder geringe Sicherheitsprobleme gefunden worden sind oder die Seite mit einer gefährlichen Seite im Zusammenhang steht.
- Ein rotes Zeichen bedeutet, dass die Seite als gefährlich eingestuft wurde, weil entweder ein Angriff stattgefunden hat oder andere Sicherheitslücken aufgetreten sind.

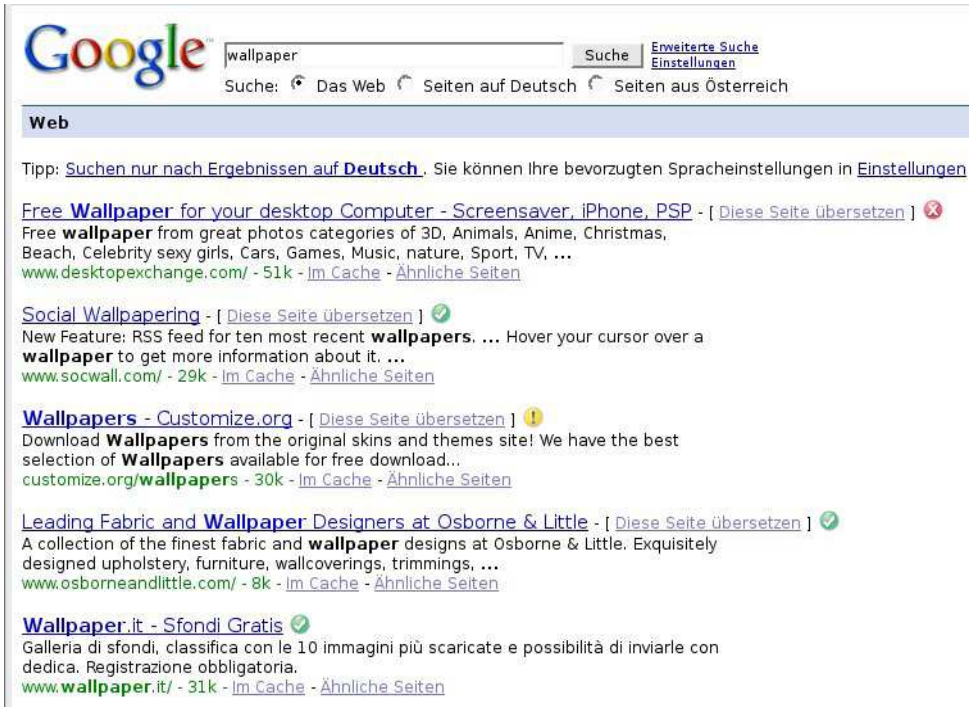

Abbildung 4.8: Die verschiedenen Zeichen des Site Advisors bei einer Internetsuche als Plugin für den Browser. Ein grünes Zeichen neben dem Suchergebnis bedeutet, dass kein Problem bei der Seite vorliegt, ein rotes, dass die Seite bei SiteAdvisor als gefährlich eingestuft wurde. Ein gelbes Zeichen bedeutet geringe Sicherheitsprobleme bei den Suchergebnissen mit Google.

- Ein graues Zeichen bedeutet, dass die Seite noch nicht geprüft worden ist. Auf der Hauptseite von SiteAdvisor kann man eine neue Adresse eingeben und diese von dem Programm prüfen lassen.

Die Ergebnisse können nicht nur im Internet aufgerufen werden, sondern auch über ein Browser Plugin für Internet Explorer und Firefox eingebunden werden, sodass bei einer Websuche mittels Google sofort neben dem Ergebnis die Klassifikation der Seite angezeigt wird (siehe Abbildung 4.8). Auch erscheint in der Statusleiste bei jeder Seite die Klassifikation auf. Weiters versucht das Team von SiteAdvisor Statistiken aus ihren Ergebnissen zu ziehen, welche veröffentlicht werden. So wurde untersucht, in welcher Suchmaschine unter den ersten Ergebnissen am häufigsten Links zu bösartigen Seiten sind und wie diese im Zusammenhang

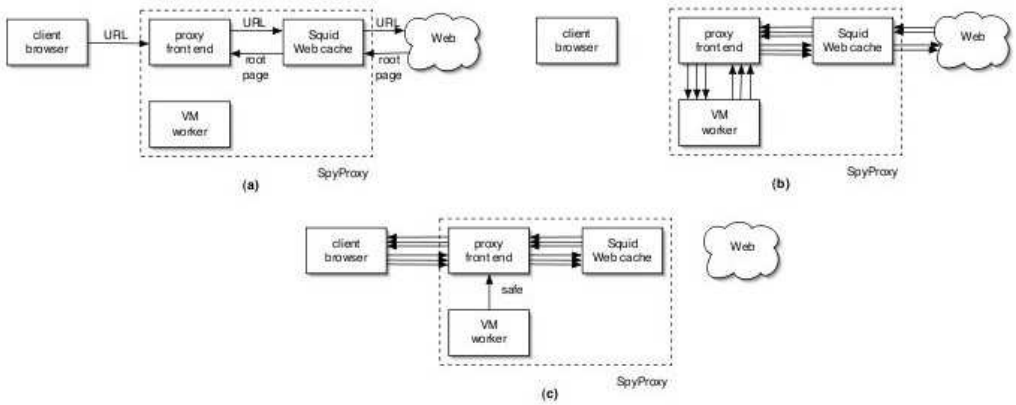

Abbildung 4.9: SpyProxy Architektur. (a) Ein Client Browser fragt eine Seite ab; das Proxy Front End schaltet sich dazwischen und empfängt die Seite und analysiert sie. (b) Wenn die Seite nicht als sicher definiert werden kann, schickt das Front End die Seite an den VM Worker. Ein Browser in der VM lädt die Seite herunter und erzeugt den Seiteninhalt. Der gesamte HTTP Transfer geht durch das Proxy Front End und den Squid Cache. (c) Wenn die Seite sicher ist, meldet die VM dies dem Front End und die Seite kann im Client aufgebaut werden [48].

stehen mit bezahltem Linkranking [20]. Weiters gibt es eine Untersuchung, in der die Seiten entsprechend ihrer Top Level Domain (TLD) klassifiziert werden [49]. Es hat sich gezeigt, dass die meisten bösartigen Seiten aus Samoa stammen, gefolgt von Internetseiten mit der TLD ".biz".

4.3.4 SpyProxy

SpyProxy ist ein high-interaction Honeyclient, der an der Universität von Washington entwickelt wurde. Eine Vorversion, UWSpycrawler, wurde schon 2005 für eine Studie [47] über die Anzahl von Spyware im Internet verwendet. Der Honeyclient wurde 2007 weiterentwickelt und in einem Paper von Moshchuk et. al. [48] vorgestellt.

Der grundlegende Aufbau ist bei dieser Implementierung anders, als bei einem herkömmlichen Honeyclient, weil ein Proxy verwendet wird (siehe Abbildung 4.9).

Dabei läuft auf einer eigenen Maschine der Proxy, welcher die Anfragen des Clients entgegennimmt und ins Internet schickt. Die Antwort darauf nimmt dieser entgegen und schickt sie weiter an das so genannte Proxy Front End. Dieses überprüft die erhaltenen Daten des Proxys, ob sie aus statischen Inhalten oder dynamischen Inhalten bestehen. Dynamisch ist jedes Javascript Element oder jeder ActiveX Aufruf. Statisch sind Inhalte, die vom Browser dargestellt werden können, ohne vorher ausgeführt zu werden. Da sich in der Vergangenheit gezeigt hat, dass auch Bilder genutzt werden können, um Sicherheitslücken in Browsern auszunutzen [44], werden nur Objekte als statisch angenommen, welche als Content Type HTML besitzen. In einer Studie wurde überprüft, wie viel des Internetverkehrs statisch ist und dabei hat sich gezeigt, dass 58% der transferierten HTML Seiten passiven Inhalt enthielten. Dadurch wird der Honeyclient wesentlich schneller, da nicht alle Seiten an die virtuelle Maschine übergeben werden müssen.

In der virtuellen Maschine lädt ein Browser nochmals die Seite herunter und zeigt den Inhalt an. Um Aktivitäten des Browsers zu überwachen, werden so genannte „event triggers“ installiert, welche mögliche Angriffe erkennen und dadurch ein Ergebnis über die angefragte Seite bereitstellen. Außerdem wird nach dem Laden der Seite der Rechner mit dem Programm AdAware [81] überprüft, um Spyware zu finden. Dieses Ergebnis wird an das Proxy Front End weitergegeben, welches entscheidet, ob die Seite an den Client geschickt wird oder nicht.

Da der gesamte Ablauf des SpyProxys manchmal viel Zeit kostet, wurden einige Performance Optimierungen eingebaut. So wird jede Seite im Proxy zwischengespeichert, um das Herunterladen aus dem Internet zu beschleunigen. Weiters wird jedes Ergebnis beziehungsweise jede Entscheidung des Proxy Front Ends gespeichert, um für einen nochmaligen Aufruf der Seite schneller den Client bedienen zu können. Natürlich muss dabei überprüft werden, ob sich an der Seite nichts geändert hat. Allerdings ist dies die Aufgabe des Proxys.

Zusätzlich wird in der Seite analysiert, welche dynamischen Teile es gibt, und diese werden einzeln überprüft. Ist ein Teil als sicher eingestuft worden, wird er an den Client weiter geschickt. So kann dieser schon einen Teil der Seite aufbauen und muss nicht auf das Gesamtergebnis warten.

Im Honeyclient wird als Proxy Squid [94] eingesetzt. Das Proxy Front End wurde selber programmiert und als virtuelle Maschine ein Windows XP mit einem

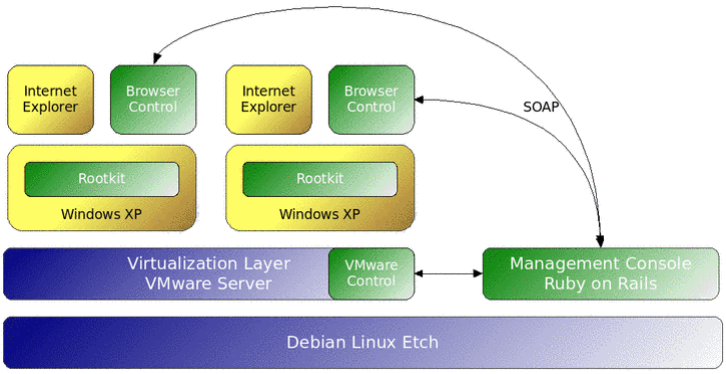

Abbildung 4.10: Die Architektur des Web Exploit Finders. In jeder virtuellen Maschine wird ein Rootkit und das BrowserControl installiert, welches die Aktionen in der Maschine registriert und mittels SOAP an die Management Konsole schickt. Die Management Konsole steuert den VMWare Server, um neue virtuelle Maschinen anzulegen und zu starten.[74]

ungepatchten Internet Explorer verwendet.

Es muss bei diesem Ansatz darauf geachtet werden, dass sowohl das Proxy System als auch der Client genau dieselbe Version des Betriebssystems und des Browsers besitzen. Auch der Client sollte keine weiteren Patches enthalten, weil sich gezeigt hat, dass selbst Patches Fehler enthalten können, welche von Angreifern ausgenutzt werden können [34].

4.3.5 WEF – The Web Exploit Finder

WEF ist ein high-interaction Honeyclient und wurde von drei Studenten an der Hochschule der Medien in Stuttgart entwickelt. Zurzeit gibt es die Version 2.0 des Clients.

Das System des WEF besteht aus zwei Teilen (siehe Abbildung 4.10). Auf der einen Seite gibt es die Managementkonsole, mit welcher der Client gesteuert wird. Mittels dieses Web Interfaces werden die Adressen der zu besuchenden Seiten eingegeben. Es gibt eine virtuelle Maschine, welche ein modifiziertes Windows XP enthält, die die Seitenaufrufe entgegennimmt. Die Modifikation des Betriebssystems besteht darin, dass es ein so genanntes Browser Control Modul gibt, wel-

ches ein Rootkit aktiviert beziehungsweise deaktiviert. Dieses Rootkit überwacht sämtliche Systemaufrufe wie das Erstellen einer Datei, das Löschen einer Datei oder die Ausführung eines Programmes. Nachdem eine Seite in der virtuellen Maschine geladen worden ist, wird das Rootkit abgefragt, ob ein Systemaufruf ausgeführt wurde. Wenn kein Aufruf ausgeführt wurde, ist die Seite sicher, ansonsten wird eine genaue Liste der Systemaufrufe und wann diese statt gefunden haben ausgegeben. Wenn die Maschine als infiziert markiert wurde, wird sie auf den Ausgangszustand zurückgesetzt und kann die nächste Seite laden. Die Kommunikation des Rootkits und der Managementkonsole basiert auf dem Simple Object Access Protocol (SOAP).

In Zukunft soll ein Crawler implementiert werden, welcher Adressen auf bösartigen Seiten herausfiltert und diese zusätzlich überprüft.

4.3.6 Shelia

Shelia ist ein high-interaction Client Honeypot, welcher versucht, Inhalte eines Email Clients auf Sicherheit zu überprüfen [53].

Dazu wird als Eingabe ein Verzeichnis eines Outlook Accounts ausgewählt und es werden sowohl alle Adressen, die in den Nachrichten vorkommen, als auch die angehängten Dateien überprüft. Für die Überprüfung gibt es folgende Systemkomponenten:

- Den Email Prozessor, welcher versucht, alle Internetadressen aus den Nachrichten und den Anhängen zu filtern und an den Client Emulator weiterzugeben.
- Der Client Emulator dient dazu, die passende Applikation für die jeweilige Datei aufzurufen. Diese soll in einer kontrollierten Umgebung aufgerufen werden, in der die Überprüfung auch stattfindet. Dadurch ist es möglich, auch Schwachstellen in anderen Programmen und nicht nur im Browser zu erkennen.
- Der Prozessmonitor dient dazu, während ein Programm ausgeführt wird, die Zugriffe auf das Dateisystem, das Erstellen neuer Prozesse etc. zu beobachten. Dies geschieht mittels API Hooking, wobei an bestimmte Funktionen

der Win32 API angedockt wird. Mit dieser Technik ist es möglich, bestimmte Aufrufe des Betriebsystems mit eigenen Funktionen zu überschreiben. Bei Shelia wird mit Hilfe dieser Methode der Aufruf von bestimmten Methoden protokolliert.

- Die Attack Detection Engine überprüft auf welchen Speicherbereich ein Programm zugreift. Diese Informationen enthält sie von dem Prozessmonitor. Wird ein Zugriff auf einen Speicherbereich ausgeführt, in dem normalerweise Daten stehen, wird eine Warnung erzeugt und es wird der gesamte Speicherbereich ausgegeben. Dadurch kann im Nachhinein genauer analysieren werden, welche Intention das Schadprogramm hatte.

4.3.7 Capture-HPC

Capture-HPC ist ein high-interaction Client Honeypot, welcher an der Universität von Wellington entwickelt wurde.

Der Client Honeypot besteht aus zwei Teilen, dem Capture Server und dem Capture Client. Der Server verwaltet die Clients und schickt ihnen Adressen von Seiten, welche überprüft werden sollen. Dadurch kann ein Server auch mehrere Clients verwalten und das System ist skalierbarer. Am Anfang nimmt der Server eine Liste von Adressen, welche überprüft werden sollen, entgegen. Danach werden in einer virtuellen Maschine die Clients gestartet und die Adressen werden übergeben. Jeder Client steht in Verbindung mit dem Capture Server, um die Resultate zu übermitteln. Der Client überwacht, während er eine Seite aufruft, den Status seines Dateisystems, der Registry und der Prozesse. Es können dem Server auch Listen übergeben werden, in denen Ausnahmen für Verzeichnisse, Prozesse und Registryzweige enthalten sind. Diese werden bei der Überwachung auf der virtuellen Maschine nicht überprüft. Wurde ein Angriff erkannt, werden die gesammelten Informationen an den Server übermittelt und die virtuelle Maschine auf den Ausgangszustand zurückgestellt. Der Client kann unmittelbar neue Instruktionen vom Server erhalten.

Mit Capture-HPC ist es nicht nur möglich den Internet Explorer mit Adressen zu versorgen, sondern auch Firefox oder Opera. In Zukunft soll auch für andere Client Programme wie Microsoft Office oder Outlook eine Schnittstelle implementiert

werden.

4.4 Grenzen von Honeyclients

Honeyclients bieten eine Erhöhung der Sicherheit gegen client-seitige Angriffe. Allerdings gibt es Bereiche, in denen Honeyclients keinen Sicherheitsgewinn bringen. Diese werden in dem folgendem Kapitel näher erläutert.

4.4.1 Erkennung von Honeyclients

Schafft es der Betreiber einer Malwareseite einen Honeyclient zu erkennen, kann dieser eine Seite ohne schadhaften Inhalt als Antwort senden. Dies kann einerseits passieren, wenn Honeyclients nur auf bestimmten Rechnern, zum Beispiel von Forschungseinrichtungen, verwendet werden und diese Information publik gemacht wird.

Eine weitere Möglichkeit könnte das Erkennen einer virtuellen Maschine am Client Rechner sein. Erkennt die Schadsoftware, dass sie sich in einer virtuellen Maschine befindet, wird diese nicht installiert. Gelingt dies im Kontext des Aufrufs und muss kein eigener Prozess gestartet werden, kann ein high-interaction Honeyclient diesen Angriff nicht erkennen. Allerdings verwenden viele Firmen und private Benutzer eine virtuelle Umgebung und es kann daher nicht nur auf einen Honeyclient geschlossen werden.

Weiters kann ein Webserver einer Malwareseite auf bestimmte HTTP Header, Cookies oder andere Client Informationen achten und dementsprechende Inhalte verschicken. Verwendet ein low-interaction Honeyclient einen eigenen HTTP Header, kann diesem eine Seite ohne Angriff zugesendet werden.

Es wurden schon Seiten entdeckt, die ihre Angriffe nicht bei jedem Aufruf verschicken, sondern nur jedes zweite oder dritte Mal. Diese Methode wird verwendet, damit der Angriff länger unerkannt bleibt. Solche Angriffe kann ein Honeyclient nicht zuverlässig erkennen, da diese nicht jedes Mal verschickt werden. Eine Möglichkeit wäre, jede Seite einige Male aufzurufen, allerdings steigt dadurch die Verarbeitungszeit des Honeyclients.

4.4.2 Problematik nicht indizierter Internetseiten

Verwendet ein Honeyclient einen Crawler für seine Suche oder werden Ergebnisse von Suchmaschinen verwendet, wird immer nur das so genannte "publicly indexable web" verwendet [33]. Dabei handelt es sich um das von Suchmaschinen indizierte Internet. Allerdings gibt es eine große Anzahl an Servern, welche nicht im Index einer Suchmaschine erfasst worden sind. Weiters können keine dynamischen Inhalte von Suchmaschinen indiziert werden und diese scheinen bei einer Websuche nicht auf.

4.4.3 Peer-to-peer Netzwerke

Mit der heutigen Generation von Honeyclients können nur Angriffe entdeckt werden, welche auf einem Server gespeichert sind. Peer-to-peer Netze sind dezentrale Netzwerke, bei denen die starre Client Server Aufteilung nicht mehr existiert. Jeder Knoten in dem Netzwerk ist Client und Server gleichzeitig und es gibt direkte oder indirekte Verbindungen zu jedem Teilnehmer des Netzwerkes. Inhalte und Informationen über das Netzwerk werden von den einzelnen Teilnehmern ausgetauscht. Beispiele für Programme, die diese Struktur verwenden, sind Kazaa [89], oder Limewire [90].

Da sich solche Netzwerke großer Beliebtheit erfreuen, ist auch der Austausch von Malware darüber beträchtlich. In einer Studie [29] hat sich gezeigt, dass bis zu 68% der heruntergeladenen Dateien Malware enthielten. Außerdem ist zum Beispiel das Programm des Kazaa Netzwerkes selbst ein Verteiler von Ad- und Spyware [8].

Für Peer-to-peer Netzwerke benötigen die aktuellen Honeyclients eine Adaptierung des Suchmechanismuses, um auch solche Bedrohungen zu analysieren.

4.4.4 Phishing

Bei Phishing handelt es sich um den Versuch eines Angreifers, dem Benutzer über eine gefälschte Internetseite sensible Daten zu entlocken [30, S. 270]. Dazu versendet der Angreifer meistens ein Email an eine große Anzahl von Benutzern mit einem Verweis zu einer Internetseite, die das Corporate Design einer Firma nachahmt. Darin ist ein Verweis auf einen Server enthalten, bei dem man sensible Daten

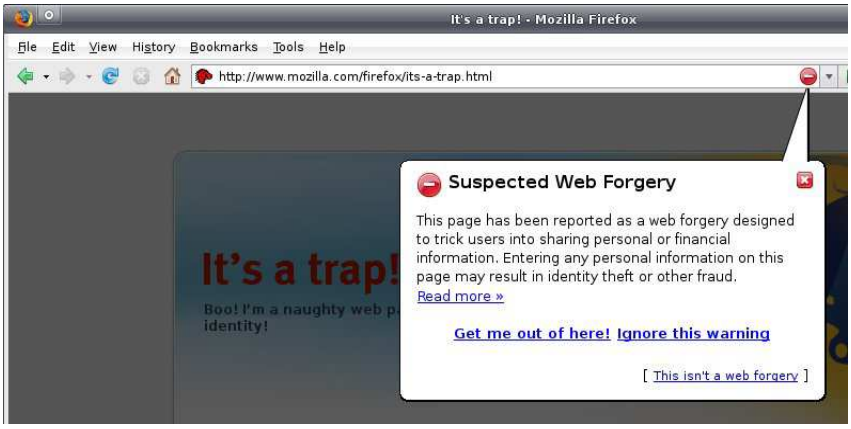

Abbildung 4.11: Der eingebaute Phishingschutz des Firefox Browsers.

eingeben soll. Die Webseite auf dem Server ist, genau wie die Email, täuschend echt dem Internetauftritt einer Firma nachempfunden, oft ist auch der Domainname der echten Institution nachempfunden. In [13] werden Softwarepakete analysiert, welche komplette Phishing Webseiten enthalten, die sehr einfach auf einem Webserver installiert werden können. Dadurch ist es auch nicht versierten Benutzern möglich, diese Technik für kriminelle Zwecke zu verwenden.

Solche Angriffe basieren auf Benutzerfehler und sind sehr schwer mit Computerprogrammen zu beseitigen. Folgende Ansätze gibt es dazu:

- Der Email Agent überprüft, ob es einen Unterschied zwischen Verweisen und der dargestellten Adresse in der Textform in einer Email gibt.

- Der Browser überprüft beim Aufruf, ob die angezeigte Adresse in einer Liste bekannter Phishingseiten vorkommt. Diese Liste muss natürlich kontinuierlich aktualisiert werden. Zum Beispiel enthält der Firefox Browser ab Version 2.0 einen eingebauten Phishingschutz [103]. Es werden in einer Datenbank gefährliche Internetadressen gespeichert und jedes Mal wenn man eine darin enthaltene Seite aufruft, erhält man eine Hinweismeldung. Diese Meldung ist in Abbildung 4.11 dargestellt.

Für einen Honeyclient ist eine Phishingseite eine Seite ohne Angriffe und daher kann eine solche Attacke nicht erkannt werden.

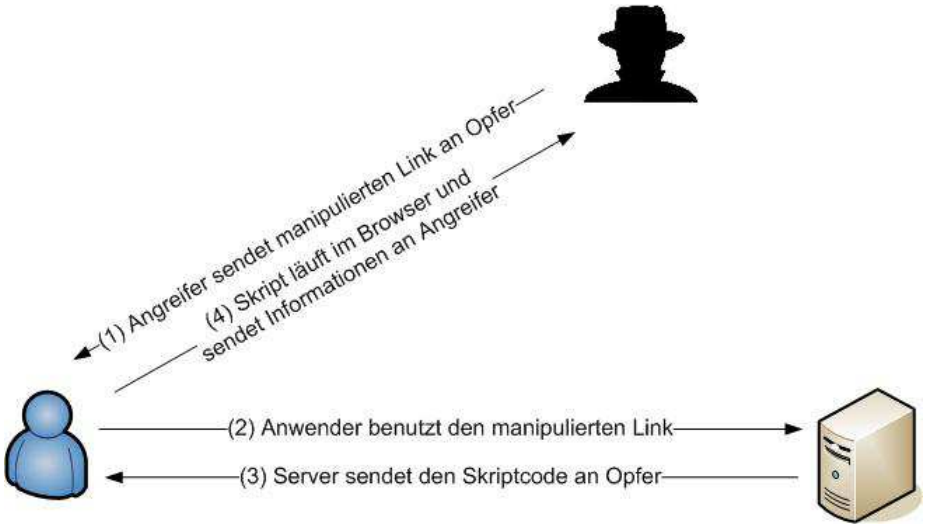

Abbildung 4.12: Eine mögliche Cross-Site Scripting Angriffsart. Der Angreifer sendet einen manipulierten Link (1) mit Skriptinformationen an den Benutzer. Dieser klickt auf den Link (2) und sendet dadurch auch Skriptelemente an den Server. Der Server sendet das Skript wieder an den Benutzer (3) wo es ausgeführt wird. Das Skript sendet Informationen an den Angreifer (4).

4.4.5 Cross-Site Scripting

> Cross-site scripting (XSS) ist ein Angriff auf Internetanwendungen, der Skriptcode in die Ausgabe der Anwendung eingeschleust und diesen anschließend an die Browser von Benutzern verschickt.[72].

Dadurch ist es möglich, dass ein Angreifer Informationen aus einem vertrauenswürdigen Kontext ausliest, obwohl sein Angriff in einem nicht vertrauenswürdigen Kontext gestartet wurde. Dieser Angriff basiert auf Javascript, Actionscript oder anderen Skriptsprachen, welche im Internet verwendet werden. Ein solcher Angriff ist in Abbildung 4.12 dargestellt.

Meistens versucht ein Angreifer mit diesen Methoden Cookies des Webbrowsers auszulesen, um sensible Informationen wie Benutzerkonten zu erhalten. In solchen Fällen kann kein Honeyclient, nicht einmal ein high-interaction Honeyclient, den Angriff erkennen. Cookies befinden sich bei den meisten high-interaction Honeyclients in den Ausnahmelisten und es tritt daher keine Warnung auf. Lediglich wenn

ein solcher Angriff einen Prozess startet oder ein Programm installiert, kann ein high-interaction Honeyclient diesen erkennen.

Eine Möglichkeit, diese Angriffe zu verhindern, ist es, auf Serverseite die Eingaben der Benutzer zu prüfen, ob spezielle Zeichenfolgen, welche auf Skriptbefehle hindeuten, vorkommen. Diese müssen verhindert werden, damit der Schadcode nicht ausgeführt werden kann. Eine weitere Möglichkeit ist Javascript und andere Skriptsprachen im Browser zu deaktivieren. Das führt allerdings dazu, dass viele Internetseiten nicht mehr funktionieren, da viele Skripte eingesetzt werden, die diese Sprache benötigen.

Für diese Problematik gibt es spezielle Browser Plugins, die das Ausführen von Skripten nur auf bestimmten Seiten zulassen, wie zum Beispiel No Script für den Firefox Browser [100].

Eine weitere Methode wird in Vogt et al. [72] vorgestellt. Dabei werden wichtige Inhalte des Browsers wie Cookies oder auch Elemente von Formularen markiert und wenn ein Javascript Programm darauf zugreifen will, kann der Benutzer entscheiden, ob diese Aktion ausgeführt werden soll oder nicht.

Weiters kann nicht nur auf der Browserseite gegen XSS Angriffe vorgegangen werden, sondern auch auf der Seite der Webapplikation. Eine Möglichkeit bei einer Anwendung zu überprüfen, ob diese Schwachstellen enthält wird in [35] behandelt. Dabei wird der Quellcode von Internetseiten mit statischen und dynamischen Methoden auf Schwachstellen überprüft.

Eine Methode die Webanwendung direkt zu überprüfen wird in [37] vorgestellt. Dabei wird versucht die Parametern von HTML Formularen auszunutzen und XSS Attacken auszuführen.

4.5 Proxy Modul für Honeyclients

Die meisten Honeyclients verwenden Listen von Adressen, welche überprüft und klassifiziert werden. Für Forschungszwecke wird diese Funktion benötigt, da oftmals eine Liste von Adressen und möglicherweise die Verweise auf diesen Seiten überprüft werden müssen. Dadurch kann man einen guten Überblick erhalten, auf welchen Seiten beziehungsweise in welchen Bereichen des Internets sich Malware befindet. Hiermit decken Honeyclients die Anforderungen von Forschungsinstitu-

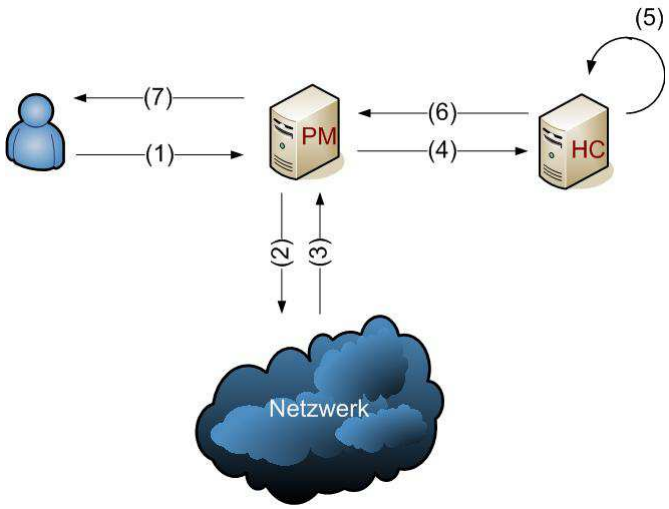

Abbildung 4.13: Die Darstellung eines Proxy Moduls. PM ist das Proxy Modul, HC der Honeyclient. Der Ablauf ist in folgende Schritte unterteilt: (1) der Benutzer wählt eine Webseite im Browser an, die Anfrage wird an den Proxy gesendet. Das Proxy Modul sendet die Anfrage an den Webserver (2) und nimmt die Antwort entgegen (3). Daraufhin sendet es die Seite an den Honeyclient (4), welcher mit der Analyse beginnt (5). Der Status der Überprüfung wird an das Proxy Modul (6) gesendet. Wenn die Seite gutartig ist, wird diese an den Benutzer geschickt, ansonsten erhält dieser einen Warnhinweis (7).

tionen ab.

Allerdings ist es in dieser Arbeit nötig, alle Anfragen von Internetseiten mit einem Honeyclient zu überprüfen.

Ein Benutzer gibt die Adresse der gewünscht Internetseite meistens im Browser ein und wartet auf die Antwort des Webservers. Auf der erhaltenen Seite ruft dieser durch das Anwählen von Verweisen neue Seiten auf oder gibt Daten in Formularen ein.

Dieser Ablauf soll bei der Verwendung eines Honeyclients möglichst beibehalten werden, da der Benutzer damit vertraut ist. Verwendet man eine Liste für das Überprüfen von Seiten, muss der Anwender jedesmal, wenn dieser einen Verweis anwählt, die Adresse in diese Liste einfügen und dem Honeyclient zum Überprüfen übergeben. Der dadurch entstandene Vorgang ist sehr unkomfortabel.

Deshalb ist die Implementierung eines so genannten Proxy Moduls für solche Honeyclients sinnvoll, welches die Aufgabe übernimmt, die eingegebene Internetadresse an den Honeyclient zu senden. Der Aufbau ist dem Proxy des Honeyclients SpyProxy (siehe Kapitel 4.3.4) sehr ähnlich. Abbildung 4.13 zeigt die Verwendung eines solchen Proxys. Damit die Webseite nicht vom Proxy Modul und vom Honeyclient jeweils einzeln angefordert werden muss, stellt das Proxy Modul diese dem Honeyclient zur Verfügung. Je nach Klassifizierung der Seite sendet das Proxy Modul diese oder einen Warnhinweis an den Benutzer zurück. Die Ergebnisse des Honeyclients kann das Proxy Modul speichern und bei einem erneuten Aufruf, falls sich die angeforderte Seite nicht geändert hat, gleich das Ergebnis weiterleiten.

Im Browser des Benutzers muss nur die Adresse des Proxy Moduls als Proxy eingetragen werden. Der Benutzer kann das gewohnte Verhalten für den Aufruf von Internetseiten beibehalten, allerdings werden diese von einem Honeyclient analysiert.

4.6 Vergleich von low-interaction Honeyclients und IDS

Netzwerk-basierte IDS schützen ein Netzwerk, indem eingehende Pakete auf bekannte Angriffsmuster untersucht werden. Findet das System eine Übereinstimmung, wird diese gemeldet und kann genauer analysiert werden.

Low-interaction Honeyclients untersuchen den eingehenden Netzverkehr ebenfalls auf bestimmte Muster, um dadurch client-seitige Angriffe zu erkennen. Es gibt Implementierungen, die IDS Systeme für diesen Prozess verwenden. Aufgrund der verwendeten Methodik sind IDS und low-interaction Honeyclients sehr ähnlich. Einige Unterschiede der beiden Systeme werden im Folgenden erläutert.

Bei einigen Implementierungen von Honeyclients muss erst eine Erweiterung entwickelt werden, damit diese vom Benutzer möglichst transparent verwendet werden können. Eine Möglichkeit dafür stellt ein Proxy Modul dar, welches in Kapitel 4.5 vorgestellt wurde. Wird jedoch ein IDS für die Analyse des Netzverkehrs eingesetzt, muss für den Benutzer keine weitere Anpassung vorgenommen werden.

Im Forschungsbereich bieten low-interaction Honeyclients einen Vorteil, da die

gesamte Funktionalität des Suchens nach gefährlichen Internetseiten in einem Programm gekapselt ist. Wenn man dagegen ein IDS verwenden will, muss erst ein Queuer Modul (wie in Kapitel 4.1 beschrieben) implementiert werden, um die Funktionalität eines Honeyclients (siehe Abschnitt 4.1) zu haben.

Allerdings erkennt sowohl ein IDS als auch ein low-interaction Honeyclient, unbekannte Angriffe nicht. Für dieses Szenario wurden high-interaction Honeyclients entwickelt.

5 Sicherheit im universitären Netzwerk

Ein universitäres Netzwerk ist eine besondere Form eines Netzwerkes, da darin besondere Regeln gelten. Wie so ein Netzwerk aufgebaut ist und welche Anforderungen darin gelten, wird in den folgenden Kapiteln behandelt. Neben den theoretischen Erläuterungen wird die praktische Umsetzung einer solchen Infrastruktur am Beispiel des Netzwerkes der Technische Universität (TU) Wien, welches auch TUNET genannt wird, behandelt.

5.1 Aufbau eines universitären Netzwerkes

Ein universitäres Netzwerk stellt die Verbindung verschiedener universitärer Einrichtungen untereinander her. Diese sind die verschiedenen Institute, Verwaltungseinrichtungen, Computerräume für Studenten und externe Verbindungen sowie die Universitätsbibliothek. Außerdem wird über dieses Netzwerk zum Internet und zu anderen akademischen Netzwerken verbunden.

Physikalisch werden zusammenliegende Institute (im selben Gebäude) gruppiert und zwischen solchen Knoten und dem Netzwerk werden Hochgeschwindigkeitsverbindungen etwa über Glasfaser hergestellt.

An der TU Wien sind viele Institute auf verschiedene Gebäude in der Stadt Wien aufgeteilt und liegen nicht an einem einzigen Standort.

Das TUNET ist zum Beispiel an das Forschungsnetz Austrian Academic Computer Network (ACOnet) angeschlossen. Damit sind auch die anderen Universitäten Österreichs verbunden, sowie etliche andere wissenschaftliche Institutionen. Von dort aus besteht die Verbindung ins Internet. Ein genauer Überblick über die verschiedenen Anbindungen findet sich in [64].

Studenten und Mitarbeiter der TU Wien können in Computerräumen und über Wireless Local Area Network (WLAN) eine Verbindung zum TUNET herstellen. Die WLAN Verbindungen sind primär unverschlüsselt, allerdings kann man über diese einen gesicherten VPN Tunnel aufbauen.

5.2 Anforderungen an ein universitäres Netzwerk

Universitäre Netzwerke besitzen spezielle Anforderungen. Wird ein Honeyclient in einer solchen Umgebung installiert, müssen diese berücksichtigt werden.

- In einem universitärem Netzwerk besteht eine hohe Bandbreite zu verschiedenen anderen Netzen, wie auch dem Internet. Die TU Wien besitzt zur Zeit eine Anbindung an das Internet mit 250 MBit/s [66]. Eine Besonderheit eines universitären Netzwerkes ist weiters, dass es viele Benutzer gibt, die sehr unterschiedliche Seiten im Internet aufrufen. Eine Forschungsgruppe etwa ruft ganz andere Adressen auf als ein Student, während etwa in einem Firmennetzwerk vorwiegend Internetseiten zu einem bestimmten Thema angewählt werden.

  Das TUNET verfügt zur Zeit über ca 10450 angeschlossene Systeme (siehe [67]), WLAN Geräte sind darin nicht enthalten.

- Es sind viele verschiedene Computer mit unterschiedlichen Betriebssystemen in einem universitären Netzwerk eingesetzt. Außerdem können sich Mitarbeiter und Studenten mit ihrem eigenen Rechner mit dem Netzwerk verbinden und die Sicherheitseinstellungen dieser Computer können nicht überprüft werden. Diese Rechner sind wesentlich schwieriger zu verwalten als zum Beispiel Desktoprechner in den Rechnerräumen oder auf den Instituten.

- In einem universitären Netzwerk befinden sich verschiedene Forschungsgruppen, welche sich in unterschiedlichen Bereichen betätigen. Diese Bereiche können auch die Sicherheit von Computersystemen oder die Internetbenutzung darstellen. Der Betreiber eines universitären Netzwerkes kann daher nicht Teile des Internets sperren, auch wenn diese gefährliche Inhalte bereitstellen, da sonst die Forschung eingeschränkt wäre.

5.3 Unterschiede zu einem Firmennetzwerk

In einem Firmennetzwerk gelten, wie in einem universitären Netzwerk, spezielle Anforderungen. In einer Firma können Regeln formuliert werden, wie sie in einer Universität auf Grund der Forschungsfreiheit nicht gelten können. So kann eine Firma bestimmte Teile des Internets sperren oder bestimmte Programme ausschließen.

Weiters gibt es in einer Firma nicht so eine hohe Fluktuation von Netzwerkteilnehmern. Zwar gibt es auch schon vielfach WLAN Zugänge, aber die meisten Mitarbeiter besitzen einen festen Rechner im Netzwerk. Die Hardware und Software dieser Rechner wird vom Betrieb festgesetzt und muss eingehalten werden. Private Rechner finden meist keinen Zugang zum Firmennetzwerk. An der Universität verwenden viele Benutzer das Netzwerk nur für eine kurze Zeit und können mit verschiedenen Hardwarekonfigurationen und Betriebssystemen Zugang erhalten.

Einen weiteren Unterschied stellt die Administration des Netzwerkes dar. In einem Firmennetzwerk gibt es einen Administrator oder eine Abteilung, die sich um die Struktur und die verwendeten Komponenten kümmert. Es kann der Einsatz von bestimmten Konfigurationen festgelegt werden. In einem universitären Netzwerk kann jedes Institut unterschiedliche Komponenten verwenden und es gibt verschiedene Administratoren. Dadurch kann nicht von einer einheitlichen Infrastruktur ausgegangen werden. Dieser Umstand wird dadurch verschärft, dass in einem universitären Netzwerk das Internet sehr intensiv genutzt wird.

5.4 Sicherheit im TUNET

Zwischen den verschiedenen Instituten und dem TUNET befinden sich so genannte Institutsfirewalls. Außerdem ist zwischen TUNET und dem Internet eine Firewall vorhanden [17]. Neben den einzelnen Instituten können auch von außerhalb Verbindungen ins TUNET aufgebaut werden. Dies können einerseits Internet Zugänge für Mitarbeiter und Studenten der TU Wien sein, wie zum Beispiel über ADSL, andererseits VPN Verbindungen. Diese Verbindungen sind wiederum mit einer Firewall geschützt [23].

Für den Betrieb des TUNET ist der Zentrale Informatikdienst (ZID) zuständig. Dieser kann bestimmte Sicherheitsrichtlinien erlassen und muss für das Funktionieren des Netzwerkes garantieren. So wurde eine Security Policy [65] herausgegeben, welche Regeln für Benutzer aufgelistet, um das intakte Funktionieren des Netzwerkes garantieren zu können.

Um die Sicherheit des TUNETs weiter zu erhöhen, hat der ZID zusätzlich folgende Maßnahmen gesetzt [28].

- Regelmäßige Kontrolle der Firewalls.
- Regelmäßige Schulung der Systemadministratoren der einzelnen Institute.
- Das ACOnet verwendet spezielle Detektiertools, um DDoS Angriffe oder Botnetze zu erkennen.
- Der ZID hat einen eigenen Server für Windows Updates installiert.

Mit Hilfe dieser Maßnahmen können Angriffe im Netzwerk schneller erkannt werden oder die Verbreitung von Viren und Würmern entsteht durch regelmäßige Sicherheitsupdates erst gar nicht. So konnten die Betreiber des TUNETs feststellen, dass sich bei den meisten Benutzern von Arbeitsplatzrechnern ein höheres Sicherheitsbewusstsein entwickelt hat. Dies zeigt sich auch in der höheren Verwendung des internen Windows Update Servers.

6 Evaluierungsumgebung für die Analyse von Honeyclients

Es werden nun ausgewählte Honeyclients genauer evaluiert, ob diese in einem universitären Netzwerk eingesetzt werden können. Diese wurden ausgewählt, da sie frei verfügbar sind.

Für die Evaluierung werden sie auf einem Rechner installiert und einigen Tests unterzogen. In diesem Kapitel wird die Konfiguration des Testrechners beschrieben und es werden verwendete Softwarekomponenten erläutert.

6.1 Testumgebung

Die Analyse der Honeyclients wurde auf einem AMD XP 2100+ Rechner mit 1GB RAM und einer 80 GB Festplatte durchgeführt. Als Betriebssystem wurde Ubuntu 7.10 [95] gewählt, da alle untersuchten Honeyclients auf dem System ausgeführt werden konnten. Die meisten Honeyclients benötigen eine Linux Umgebung, andere sind so entwickelt worden, dass sie unabhängig von einer bestimmten Plattform ausgeführt werden können.

6.1.1 Gastsystem der high-interaction Honeyclients

Bei high-interaction Honeyclients gibt es ein Betriebssystem, auf dem die zu untersuchenden Seiten aufgerufen werden. Danach wird überprüft, ob ein Angriff stattgefunden hat. Aus naheliegenden Gründen wird dieses System oft virtuell erstellt, da es dadurch wesentlich einfacher zu administrieren ist.

Einige Produkte, wie etwa VMWare, bieten eine so genannte Snapshot Funktion an, welche ein Abbild des aktuellen Zustandes einer virtuellen Maschine macht und

diesen später wiederherstellen kann. Diese Funktion erleichtert den Betrieb eines high-interaction Honeyclients, da nach einem Angriff alle Aktionen des Angriffs rückgängig gemacht werden müssen um ein sauberes, nicht infiziertes Testsystem zu erhalten. Einen Snapshot zu aktivieren geht wesentlich schneller als jede einzelne Aktion rückgängig zu machen oder gar ein neues Betriebssystem zu installieren. Im Durchschnitt benötigt die Aktion circa 10 Sekunden. Allerdings muss nicht nach jedem Seitenaufruf eines Honeyclients wieder auf den Snapshot zurückgestellt werden, sondern nur dann, wenn eine Veränderung am System stattgefunden hat.

Es ist auch möglich, eine Kopie beziehungsweise einen Klon einer virtuellen Maschine zu erstellen und diesen zu starten. Dies wird bei high-interaction Client Honeypots verwendet, um mehrere Abbilder von Betriebssystemen zu erstellen und diese gleichzeitig nach schädlichen Seiten im Netzwerk suchen zu lassen.

Für die Gastsysteme der high-interaction Honeyclients wurde Microsoft Windows XP Professional Edition mit Service Pack 2 gewählt, da die meisten clientseitigen Angriffe für dieses Betriebssystem geschrieben werden. Dies hat vor allem mit der Verbreitung von Windows zu tun. Bei einer Umfrage des ZID hat sich herausgestellt, dass 61% der Befragten Windows als Betriebssystem verwenden [68].

Bei der Standardinstallation ist der Browser von Microsoft, welcher für das Aufrufen der Seiten benutzt wurde, der Internet Explorer 6, schon enthalten. Es wurde dieser Browser gewählt, da die meisten Angriffe für ihn geschrieben werden [57].

Auf dem System sind die automatischen Updates deaktiviert und es ist kein Antivirusprogramm installiert. Dies soll ein automatisches Herunterladen von Sicherheitsupdates verhindern. Außerdem soll nicht der Antivirus die Angriffe erkennen, bevor der Honeyclient eine Möglichkeit dazu hat.

6.1.2 Metasploit Framework

Das Metasploit Framework geht aus dem Metasploit Project hervor, einem Open Source Security Projekt, das Informationen über sehr viele Schwachstellen bereitstellt und hilft, Signaturen für IDS zu schreiben. Es sammelt diese Informationen in einem leicht bedienbaren Programm, mit dem es möglich ist, Angriffe gegen entfernte Ziele zu starten. Entwickelt wurde es für Sicherheitsspezialisten, die neue

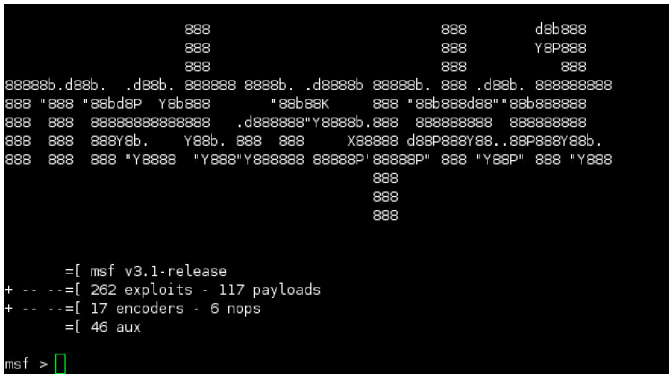

Abbildung 6.1: Die Konsole des Metasploit Frameworks nach dem Start

Schwachstellen erforschen und diese in das Framework einbauen.

Die aktuelle Version des Frameworks ist 3.1 und es ist mit der Programmiersprache Ruby entwickelt worden. Dadurch ist es auf vielen unterschiedlichen Systemen einsatzbereit. Es gibt das Programm mit verschiedenen Benutzerschnittstellen, wobei alle auf denselben Kern zurückgreifen.

- msfconsole ist die Konsolenanwendung, welche die Hauptschnittstelle darstellt. Die anderen Benutzerschnittstellen greifen auf die Konsole zurück, um Aktionen auszuführen. Alle Befehle werden über diese eingegeben und gestartet. Die Anwendung nach dem Start ist in Abbildung 6.1 zu sehen.

- msfweb ist ein Webinterface des Frameworks, das mit jedem Browser aufgerufen und bedient werden kann. Eine Testversion des Frameworks kann unter http://www.metasploit.com:55555 eingesehen werden. Damit ist es allerdings nicht möglich, Angriffe gegen Computer zu starten. Als Webserver wird die Ruby Bibliothek WEBrick [97] verwendet. In diesem Interface kann man auch auf die Konsole des Frameworks innerhalb des Browsers zurückgreifen.

- msfgui ist seit der Version 3.1 enthalten und stellt alle Funktionen der Konsolenversion in einem grafischen Benutzerinterface zur Verfügung. Die Exploits können mit einem Wizard konfiguriert und gestartet werden. Die Benutzerschnittstelle mit dem Wizard kann man in Abbildung 6.2 sehen. Auch hier kann man auf die Konsole des Frameworks zurückgreifen.

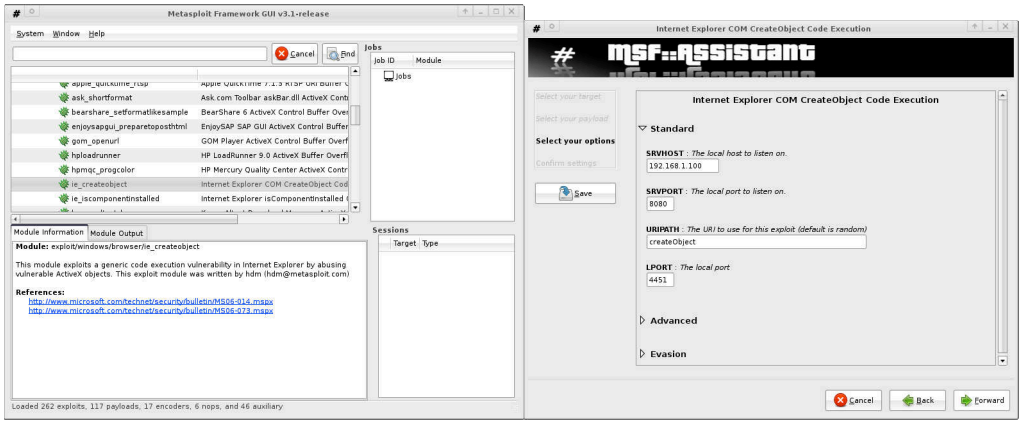

Abbildung 6.2: Das grafische Benutzerinterface des Metasploit Frameworks mit Exploit Wizard

Die Angriffe sind thematisch sortiert, um ein Auffinden dieser zu erleichtern. Die für die Evaluierung der Honeyclients benötigten Angriffe sind zum Beispiel in der GUI Version unter windows / browser eingeordnet. Hier können allerdings nicht alle Exploits für diesen Testlauf verwendet werden, da viele auf spezielle Plugins im Browser abzielen und nicht alle für den Internet Explorer geschrieben wurden. Weiters gibt es auch eine Suche, mittels der man Angriffe aus der Liste heraus filtern kann.

Jedem Benutzerinterface verwendet für das Starten eines Exploits die Konsole von Metasploit. Dort kann auch der aktuelle Status und Verlauf beobachtet werden kann. Weiters bietet das Framework zu jedem Angriff weitere Informationen und weiterführende Artikel an. Es ist auch möglich, sich den Sourcecode jedes Angriffes anzeigen zu lassen, um die genaue Funktionsweise zu analysieren.

6.2 Ausgewählte Bedrohungen und Angriffe

Wie im vorherigen Kapitel beschrieben, ist das Testsystem auf Microsoft Windows und den Internet Explorer zugeschnitten, daher wurden die Angriffe auch für diesen ausgewählt. In diesem Kapitel wird auf die einzelnen Angriffe eingegangen und analysiert, welche Schwachstelle im System sie ausnutzen.

Es können nur solche Angriffe für die Evaluierung der Honeyclients gewählt

werden, die einerseits client-seitige Angriffe ausführen und andererseits auf der Testplattform eingesetzt werden können. Die Angriffe wurden zum Teil aus dem Metasploit Framework entnommen.

6.2.1 WMF handler SetAbortProc GDI Escape vulnerability

Diese Sicherheitslücke wurde im Dezember 2005 entdeckt und erlaubt es einem Angreifer remote Code auszuführen [45]. Dabei handelt es sich um einen Fehler im Windows Graphic Display Interface (GDI). Dieses wird dazu verwendet, Grafiken und formatierten Text auf dem Bildschirm und dem Drucker auszugeben. Damit können unter anderem Linien, Texte oder Kurven gezeichnet werden.

Die betroffene Funktion der Schwachstelle ist "SETABORTPROC", welche eine GDI Escape Funktion darstellt. Mit dieser ist es Anwendungen möglich, auf Ressourcen eines Gerätes, welches nicht direkt verfügbar ist, zuzugreifen. Diese Funktion wird unter anderem vom Windows Metafile (WMF) verwendet, welches Texte oder Grafiken darstellen kann.

Wird eine manipulierte WMF Datei geöffnet, ist es dem Angreifer möglich, beliebigen Schadcode auszuführen. Da diese Funktion von der Windows Bild- und Faxanzeige genutzt wird, welche auch andere Programme, wie der Windows Explorer, Mozilla Firefox oder Outlook für die Anzeige benutzen, kann die Schwachstelle auch in diesen Programmen ausgenutzt werden. Weiters greifen manche Programme wie IrfanView [88] oder XnView [98] direkt auf die Funktion zu und sind dadurch ebenfalls angreifbar. Die unterschiedlichen Möglichkeiten die Funktion aufzurufen werden in Abbildung 6.3 dargestellt. Eine detaillierte Abhandlung über die Schwachstelle ist in [70] zu finden.

Microsoft hat die Schwachstelle unter dem Security Bulletin MS06-001 gespeichert [44].

6.2.2 Windows ANI LoadAniIcon Stack Overflow

Bei dieser Sicherheitslücke ist es einem Angreifer möglich, über einen manipulierten animierten Mauszeiger remote Code auszuführen. Dabei handelt es sich um einen Fehler der Funktion LoadAniIcon in der Bibliothek user32.dll, der durch einen Stack Overflow Angriff ausgenutzt werden kann [15].

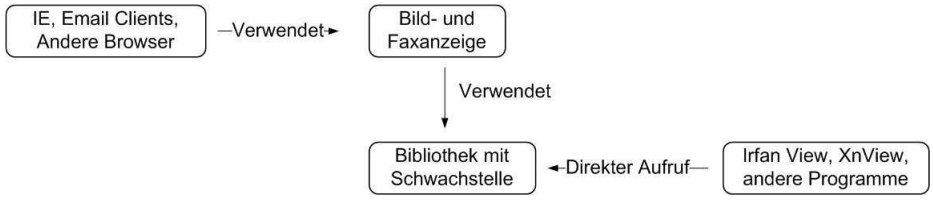

Abbildung 6.3: Unterschiedliche Arten von Angriffs Vektoren für die WMF Sicherheitslücke. Einerseits wird die schadhafte Funktion von der Bild- und Faxanzeige verwendet, welche wiederum von einigen Programmen für die Anzeige von Bildern benutzt wird, andererseits greifen manche Applikationen direkt auf die Funktion zu.

Jedes Programm, das einen animierten Mauszeiger öffnet, kann diesen Angriff hervorrufen, also nicht nur der Internet Explorer, sondern auch der Mozilla Firefox, der Windows Explorer, Winword oder Powerpoint. Es können die Systeme Windows 2000 SP4, Windows XP, Windows Server 2003 und Windows Vista angegriffen werden.

Microsoft hat den Exploit mit dem Security Bulletin MS05-002 [41] teilweise behoben. Allerdings konnte der Fehler mit speziellen Parametern noch immer ausgenutzt werden, worauf Microsoft einen weiteren Security Bulletin (MS07-017 [40]) mit einem Sicherheitspatch eröffnet hat.

6.2.3 WebViewFolderIcon Overflow

Bei dieser Sicherheitslücke werden in der Funktion setSlice des ActiveX Moduls WebViewFolderIcon die Eingabeparameter ungenau überprüft, wodurch es einem Angreifer möglich ist, auf der Clientseite Programmcode auszuführen [69]. Diese Funktion wird normalerweise dazu genutzt, in der Windows Explorer Web Anzeige Icons darzustellen.

Microsoft hat diese Sicherheitslücke unter dem Security Bulletin MS 06-057 gespeichert [39], der Angriff betrifft alle Windows Betriebssystem Versionen. Das erste Mal publiziert wurde der Angriff am 27. September 2006, am 10. Oktober 2006 stellt Microsoft einen Sicherheitspatch zur Verfügung. Eine weitere Lösung für diese Sicherheitslücke ist das WebViewFolderIcon ActiveX Modul im Internet Explorer oder alle ActiveX Module im Browser zu deaktivieren.

6.2.4 Schwachstelle in Microsoft Data Access Components

Bei dieser Schwachstelle können Komponenten des Microsoft Data Access Components (MDAC) so initialisiert werden, dass diese nicht mehr an den eingeschränkten Sicherheitskontext des Internet Explorers gebunden sind, sondern an den Benutzer. Dadurch ist es einem Angreifer möglich, beliebigen Schadcode auszuführen. Eine mögliche Komponente des MDAC ist das "RDS.Dataspace Active X Control".

Bei MDAC handelt es sich um ein Framework, welches Entwicklern erlaubt, einheitlich auf verschiedene Datenbankschnittstellen zuzugreifen. Diese sind zum Beispiel ActiveX DataObjects (ADO), OLE DB oder Open Database Connectivity (ODBC).

Die Schwachstelle kann durch einen Sicherheitspatch oder durch ein Update des MDAC behoben werden. Diese Sicherheitslücke wird im Microsoft Security Bulletin 06-014 [42] beschrieben.

6.2.5 VML Fill Method Code Execution

Bei diesem Angriff kann mittels eines Stack-based Buffer Overflows eine Schwachstelle in der Vector Graphics Rendering Engine (vgx.dll) ausgenutzt werden, welche normalerweise die Funktion hat, Bilder auf dem Computermonitor darzustellen. So kann mittels einer Vector Markup Language (VML) Datei mit zu langen Parametern Schadcode ausgeführt werden. Diese Schwachstelle existiert in Microsoft Outlook und dem Internet Explorer.

Diese Schwachstelle wird von dem Trojaner "Trojan.Vimalov" benutzt, um sich auf einem Rechner zu installieren.

Microsoft beschreibt die Schwachstelle in seinem Security Bulletin MS06-055 [43].

7 Evaluierung ausgewählter Honeyclients

Es werden nun einige der in Kapitel 4 vorgestellten Honeyclients ausgewählt und miteinander verglichen. Dabei werden Unterschiede in der Implementierung aufgezeigt und wie sich diese auf verschiedene Ergebnisse auswirken. Für die Evaluierung der Honeyclients werden folgende Thesen aufgestellt und mit statistischen Methoden und Untersuchungen widerlegt oder bestätigt werden.

- Low- und high-interaction Honeyclients unterscheiden sich sowohl in ihrer Verarbeitungsgeschwindigkeit als auch in der Erkennungsrate von Angriffen. Unbekannte Angriffe werden von low-interaction Honeyclients nicht erkannt, während high-interaction Honeyclients diese erkennen.
- Low-interaction Honeyclients sind in der Analyse schneller als high-interaction Honeyclients.
- Unterschiedliche Honeyclients sind für unterschiedliche Einsatzszenarien geeignet. Nicht jeder Honeyclient eignet sich für den Einsatz in einem universitären Netzwerk.

7.1 Vergleichskriterien für Honeyclients

Die Evaluierung der Honeyclients erfolgt nach verschiedenen Kriterien. Diese Kriterien versuchen auch Aspekte, welche nicht direkt mit dem Betrieb eines Honeyclients in Verbindung gebracht werden, abzudecken. Die Vergleichskriterien sind:

Geschwindigkeit: Jeder Honeyclient hat eine unterschiedliche Verarbeitungsgeschwindigkeit, welche sich nach der Architektur, dem Aufbau und den einge-

setzten Technologien richtet. Vor und nach jedem Testdurchlauf wird die Zeit gemessen, um danach die Bearbeitungszeit zu ermitteln. Hierbei ist auch zu berücksichtigen, wie das zu überprüfende Objekt aufgebaut ist. Außerdem muss bei entfernten Ressourcen die Verbindungsgeschwindigkeit berücksichtigt werden. Eine weitere Frage ist, ob sich die Verarbeitungsgeschwindigkeit eines Clients von Seiten, auf denen Angriffe gefunden wurden, von solchen ohne gefundener Angriffe unterscheidet.

Benutzbarkeit: Die Benutzung eines Honeyclients sollte für den User möglichst transparent sein. Das bedeutet, dass die normalen Arbeitsabläufe in gewohnter Weise erfolgen sollen, welches sich vor allem auf das Aufrufen einer Internetseite bezieht. Da der Honeyclient den Datenverkehr zwischen Browser und dem Netzwerk untersuchen muss, wird analysiert, wie die einzelnen Programme dies lösen.

Auswertbarkeit: Die gesammelten Ergebnisse und Meldungen der Honeyclients müssen analysiert und ausgewertet werden. Es kann dabei unterschieden werden, ob die Ergebnisse detailliert ausgegeben werden oder der Benutzer nur eine Warnung bekommt, wenn dieser ein gefährliches Objekt aufgerufen hat. Gerade in einem universitären Netzwerk treffen sehr viele unterschiedliche Benutzergruppen aufeinander, welche auch über einen unterschiedlichen Wissensstand in Bezug auf Computersicherheit verfügen. Hier soll analysiert werden, wie die Meldungen der unterschiedlichen Clients ausgewertet werden und diese Informationen weiter verarbeitet werden können.

Alarmierung: In diesem Punkt geht es darum, wie die erhaltenen Informationen genutzt werden können, um ganz oder teilweise automatisierte Meldungen an Systemadministratoren zu schicken. Gerade in einem universitären Netzwerk gibt es verschiedene Stellen der Administration. Eine weitere Möglichkeit wäre, ab einem gewissen Schwellenwert Meldungen eines Institutes automatisch an zentrale Organisationseinheiten zu schicken.

Erweiterbarkeit und Anpassungen: Es soll untersucht werden, wie die verschiedenen Honeyclients erweitert werden können, um Anpassungen für ein universitäres Netzwerk vorzunehmen. Der Administrator eines Netzwerkes ver-

wendet bestimmte Programme, um den Zustand zu überwachen. Diese sollen möglichst auch Daten des Honeyclients empfangen können und weiter analysieren. Wenn definierte Schnittstellen vorhanden sind, ist es leichter, Erweiterungen zu programmieren.

7.2 Beschreibung der ausgewählten Honeyclients

Für die Evaluierung der Honeyclients wurden folgende Systeme ausgewählt:

- Capture-HPC 2.0
- WEF 2.0
- HoneyC 1.3
- Monkey Spider
- SpyBye 0.3

Diese wurden gewählt, da sie frei verfügbar sind. Es gibt mehrere andere interessante Beispiele von Honeyclients, welche in Kapitel 4 vorgestellt wurden, allerdings sind diese nicht alle frei zugänglich. Etliche Versionen wurden nur für Forschungsprojekte implementiert, um Möglichkeiten von Honeyclients aufzuzeigen.

Bei der Auswahl sind zwei high-interaction und drei low-interaction Systeme enthalten. Dadurch können auch Vergleiche zwischen den beiden Typen von Honeyclients gezogen werden und durch Ergebnisse belegt werden.

7.3 Beschreibung der Testdurchläufe

Die Testdurchläufe für die Analyse der Honeyclients sind in drei Phasen untergliedert, welche verschiedene Szenarien für Honeyclients untersuchen. Die Szenarien decken verschiedene Arten von Internetseiten ab und es kann analysiert werden, wie Honeyclients darauf reagieren. Gerade in einem universitären Netzwerk kann davon ausgegangen werden, dass viele verschiedene Internetseiten aufgerufen werden, welche auch schadhafte Inhalte enthalten können. Die verschiedenen Phasen versuchen diese Umstände abzudecken.

In [22] wird das Ausmaß von Malware in einem universitären Netzwerk untersucht. Dabei wurden 16000 IP Adressen in einem universitärem Umfeld acht Wochen lang beobachtet und die Ergebnisse analysiert. Insgesamt wurden über 13 Millionen erfolgreiche Exploits gefunden, welche von 2000 verschiedenen Malwareprogrammen ausgingen.

7.3.1 Phase 1: Internetseiten ohne Angriffe

Die Honeyclients werden mit Seiten, die keine Angriffe enthalten, konfrontiert. Diese stellen den Großteil des Internets dar und es soll untersucht werden, ob alle getesteten Seiten richtig klassifiziert werden. Gerade bei high-interaction Honeyclients kann durch falsch konfigurierte Ausnahmelisten eine falsche Einschätzung entstehen. Ist etwa das Cache Verzeichnis des Browsers nicht in der Liste enthalten, sendet der Honeyclient jedes Mal eine Meldung, wenn der Browser darin eine Seite ablegt. Der Cache dient einem Browser dazu, Informationen aus dem Netzwerk zu speichern. Bei einem erneuten Aufruf der Ressource wird diese nicht nochmals aus dem Internet geladen, sondern aus einem Verzeichnis ausgelesen. Dieser Vorgang ist normalerweise wesentlich schneller. Allerdings muss überprüft werden, ob die aktuellen Informationen zur Verfügung stehen oder neue Teile aus dem Netzwerk nachgeladen werden müssen [16].

Für diese Phasen wurden die 100 von Österreich aus meistbesuchten Internetseiten nach einer Liste von Alexa verwendet [101]. Dabei handelt es sich um eine Firma, welche unterschiedliche Statistiken über Internetseiten und das Surfverhalten anbietet. Sie erhält diese Daten, indem Benutzer ein Browser Plugin installieren, welches Informationen über die aufgerufenen Internetseiten an einen Server schickt. Diese werden statistisch erfasst und ausgewertet.

Beim Vergleich der verschiedenen Phasen muss darauf geachtet werden, dass es sich hierbei um Internetseiten handelt, welche manchmal viele Bilder enthalten. Deshalb wird neben der Zeit pro Adresse auch die Anzahl der Megabyte pro Minute berechnet. Dadurch kann der Datendurchsatz jedes Honeyclients verglichen werden. Die Gesamtgröße der Internetseiten in Phase 1 beträgt 24 Megabyte, es werden 1819 unterschiedliche Inhalte wie Text, Bilder, Javascript untersucht. Den Großteil der Datenmenge stellen dabei HTML Seiten dar, gefolgt von Bildern. Wei-

ters muss bei der Verarbeitungsgeschwindigkeit berücksichtigt werden, dass es sich dabei um eine Internetverbindung handelt und es im Netzwerk zu unterschiedlichen Antwortzeiten kommen kann. Die Server stellen eine weitere Möglichkeit für Verzögerungen dar, da diese mehrere Anfragen gleichzeitig beantworten müssen und die Bearbeitungsgeschwindigkeit dadurch unterschiedlich sein kann.

7.3.2 Phase 2: Internetseiten mit bekannten Angriffen

In Phase 2 werden Seiten überprüft, die bösartige Inhalte aufweisen, welche aber den Honeyclients bereits bekannt sind. Dies gilt nur für low-interaction Honeyclients, da diese auf Signaturen basieren. High-interaction Honeyclients enthalten keine Signaturen, weshalb diese Phase für sie äquivalent mit dem Szenario aus Kapitel 7.3.3 ist.

Es wird überprüft, ob die Signaturen auf die Seiten richtig angewendet werden und dadurch eine richtige Klassifikation ergeben. Hier werden Angriffe, welche in Kapitel 6.2 beschrieben sind, verwendet.

7.3.3 Phase 3: Internetseiten mit unbekannten Angriffen

Zum Schluss wurden Internetseiten gewählt, welche für Honeyclients unbekannte Angriffe enthalten. Die hier verwendeten Angriffe wurden in Kapitel 6.2 erläutert. Um die Situation von unbekannten Angriffen für low-interaction Honeyclients zu simulieren, wurden die benutzten Signaturen um diese reduziert. Diese Phase soll die Hypothese bestätigen, dass low-interaction Honeyclients diese Internetseiten falsch klassifizieren, während high-interaction Honeyclients sie als bösartig erkennen. Die Internetseiten werden auf einen Webserver geladen, welchen die Honeyclients abfragen und analysieren. Die Liste der Internetadressen ist für alle Honeyclients dieselbe, unter Umständen muss diese jedoch für einige Honeyclients in ein anderes Format transformiert werden. Die Verbindung ist für alle Systeme gleich, da sich der Webserver im lokalen Netzwerk befindet und die Kommunikation nicht durch äußere Faktoren wie hohe Netzlast beeinflusst wird.

7.4 Beschreibung der Evaluierung

Für die Durchführung der Evaluierung werden die ausgewählten Honeyclients auf einem System installiert. Außerdem werden die Internetseiten mit Angriffen bereitgestellt, damit diese in Phase 2 und Phase 3 genutzt werden können.

Es werden nun alle Honeyclients für jede Phase gestartet, die Zeit wird gemessen und die Ergebnisse werden gespeichert. Sind alle Systeme mit einer Phase fertig, werden die Honeyclients auf die nächste Phase vorbereitet und diese wird gestartet. In der Vorbereitung werden die Adressenlisten aufbereitet und für low-interaction Honeyclients die richtigen Signaturen eingestellt. Gerade beim Abarbeiten von Phase 2 und Phase 3 sind die verwendeten Signaturen ausschlaggebend, da diese das Ergebnis bestimmen.

7.5 Konkrete Analyseergebnisse

Es werden die Ergebnisse für die einzelnen Honeyclients beschrieben und erörtert. Dadurch erhält man für jeden Honeyclient einen genauen Überblick über sein Verhalten während der Testphasen. In Kapitel 7.6 werden die Ergebnisse zusammengefasst und es werden allgemeine Aussagen über Honeyclients gemacht.

7.5.1 Capture-HPC

Die zu besuchenden Adressen werden in eine Textdatei eingetragen, welche der Client zeilenweise durcharbeitet. Durch seinen Aufbau ist es möglich, mehrere Honeyclients gleichzeitig zu benutzen. Bei allen Phasen wird eine Zeit von mindestens 10 Sekunden eingestellt, die ein Besuch einer Seite mindestens benötigt. Die Einstellung wurde so gewählt, damit auch Angriffe mit kurzen Wartezeiten erkannt werden können. Bei einigen Angriffen müssen zum Beispiel an bestimmten Speicherstellen große Mengen an Daten geschrieben werden. Dieser Vorgang benötigt allerdings etwas Zeit, welche der Honeyclient aber abwarten muss, damit er den Angriff erkennen kann.

Die mit Capture-HPC mitgelieferten Ausnahmelisten sind für eine englische Version von Windows XP ausgelegt. Daher mussten sie vor der Verwendung auf eine

deutsche Version umgestellt werden. Das betrifft vor allem die Verzeichnisnamen wie “Eigene Dateien” oder “Programme”. Zusätzlich wurde noch der Windows Debugger zur Ausnahmeliste hinzugefügt.

Bei Phase 1 benötigt der Honeyclient für alle Adressen 31 Minuten und 40 Sekunden. Das ergibt eine Zeit von 19 Sekunden pro Adresse. Wird bei einer Adresse kein Angriff festgestellt, wird die virtuelle Maschine nicht neu initialisiert. Bei einem Fund wird die virtuelle Maschine auf den vorher definierten Snapshot zurückgesetzt. Der Browser der virtuellen Maschine wartet jedes Mal bis die Seite vollständig geladen ist, bevor eine neue Seite aufgerufen wird. Daher kann bei einem langsamen Server auch eine Zeit von über einer Minute entstehen, die der Client für den Aufruf eine Adresse benötigt.

In dieser Phase wurde testweise die Wartezeit pro Adresse von Capture-HPC auf eine Sekunde gestellt. Wie erwartet verringert sich dadurch die benötigte Zeit. Der Honeyclient benötigt pro Adresse im Durchschnitt 9 Sekunden weniger als beim vorigen Durchlauf.

Weiters wurde in dieser Phase statt einer virtuellen Maschine eine zweite verwendet und beide teilen sich die zu überprüfenden Seiten auf. Hier hat sich gezeigt, dass die Analysezeit um 10 Minuten reduziert werden konnte, auf 21 Minuten und 25 Sekunden. Dies bedeutet eine Steigerung um 32,4%. Testweise wurde noch eine dritte virtuelle Maschine hinzugefügt. Allerdings ist hier der Verwaltungsaufwand für das System höher und die Verarbeitungszeit wurde mit 25 Minuten und 17 Sekunden länger als bei zwei virtuellen Instanzen. Dies liegt an der Leistung des verwendeten Systems, mit einem schnelleren System können bessere Ergebnisse erzielt werden.

Bei Phase 2 und 3 hat Capture-HPC einige Probleme mit den bereitgestellten Angriffen. Erkannt wurden alle Angriffe, jedoch friert bei einigen dieser Seiten der Internet Explorer ein und kann von der Honeyclient Software nicht beendet werden. Dies geschieht aufgrund der Tatsache, dass bei den Angriffen Buffer Overflows verwendet werden, sodass der Internet Explorer seine Routinen nicht wie gewohnt abarbeiten kann. In diesem Fall muss der Browser über den Taskmanager manuell beendet werden. Wird ein Programm auf diese Weise beendet, startet Windows einen Debugger und speichert Informationen über den Speicherstatus ab. Wird der Debugger nicht auf die Ausnahmelisten des Honeyclients gesetzt, wird der

Datei	Beschreibung
safe.log	Enthält die Liste der Adressen, welche besucht wurden und als gutartig eingestuft wurden.
progress.log	Enthält alle Informationen über Adressen, welche zur Zeit besucht werden oder bereits besucht wurden. Wenn ein Fehler beim Aufrufen einer Adresse entsteht, wird er in dieser Datei gespeichert. Capture-HPC versucht eine Adresse fünfmal aufzurufen, wenn es ihm nicht gelingt, die Adresse zu erreichen, wird diese in der Datei error.log eingetragen.
error.log	Enthält Adressen, welche nicht besucht werden konnten, und auch Informationen über Fehler bei der Client Server Kommunikation.
malicious.log	Die Datei enthält eine Liste von Adressen, welche als schadhaft eingestuft wurden.
server_timestamp.zip	Enthält alle detaillierten Berichte über eine schadhafte Adresse in einem gepackten Archiv.

Tabelle 7.1: Die Dateien, die beim Ausführen von Capture-HPC angelegt werden, und eine Beschreibung ihres Inhaltes

Aufruf des Debuggers als Angriff klassifiziert. Dabei werden die gesamten Speicherinformationen, welche der Debugger liefert, an den Honeyclient gesendet. Für die Analyse eines Angriffes benötigt Capture-HPC eine Zeit von 36 Sekunden pro Adresse.

Wird der Browser manuell beendet, führt der Honeyclient die Analyse bei der nächsten Seite fort. Sind am Ende der Liste alle Seiten abgearbeitet, wartet der Client auf neue Eingaben in der Liste oder er kann manuell beendet werden.

Aufgrund der Tatsache, dass nach einem gefundenem Angriff der Snapshot des sauberen Systems aktiviert werden muss, benötigt der Honeyclient für eine bösartige Seite länger als für eine gutartige.

Die Ergebnisse der Suche werden in ein eigenes Verzeichnis sowohl in dem Dateisystem der virtuellen Maschine als auch auf dem Server geschrieben. Darin befinden sich die generellen Logdateien, welche in Tabelle 7.1 erklärt sind. Weiters wird für jeden Fund eine Datei angelegt, in der festgehalten wird, welche Aktion, wann, von welchem Programm aus gestartet wurde. Auch Einträge im Registrierungseditor werden so erfasst.

Für die Benutzung des Clients muss als Eingabe die Liste mit Adressen gefüllt werden. Dadurch müssen die zu überprüfenden Seiten vorher feststehen und eingetragen werden, eine Überprüfung einer Seite während des Aufrufens mit dem Browser ist hier nicht möglich. Allerdings könnte man für einen Browser ein Plugin schreiben oder ein in Kapitel 4.5 vorgestelltes Proxy Modul, welches jede eingegebene Adresse mit Capture-HPC überprüft.

Die erstellten Textdateien mit den Meldungen können analysiert und ausgewertet werden. Da es auch Dateien mit Zusammenfassungen der Ergebnisse gibt, können grundlegende Ergebnisse einfach präsentiert werden.

Die Protokolldateien können bei erhöhtem Aufkommen anderen Administratoren des Netzwerks automatisiert zugeschickt werden. Diese können aufgrund der enthaltenen Informationen prüfen, wie das weitere Vorgehen beziehungsweise der weitere Aufruf bestimmter Internetseiten ablaufen soll.

Da der Quellcode sowohl für den Client- als auch für den Serverteil des Honeyclients frei zu Verfügung steht, können Erweiterungen in das System eingebaut werden. Die Verwendung von verschiedenen Programmen für die Abarbeitung ist im System eingebaut. Es kann bei jeder Internetadresse ein Programm festgelegt werden, mit dem diese untersucht werden soll.

7.5.2 WEF

Bei WEF werden die zu untersuchenden Objekte über die Managementkonsole oder direkt in die MySQL Datenbank eingetragen. Mehrere Instanzen von virtuellen Maschinen können auch darüber erstellt werden, um die Internetseiten gleichzeitig zu untersuchen.

Die mitgelieferten Ausnahmelisten von WEF sind gut ausgearbeitet und bedürfen keiner weiteren Adaptierung.

Bei Seiten ohne Angriffe hat WEF keine Probleme. Nur bei einer einzigen Seite ist das Browser Control Programm in der virtuellen Maschine abgestürzt und musste manuell neu gestartet werden. Die Gründe dafür konnten nicht ermittelt werden. Für alle Seiten benötigte der Honeyclient 36 Minuten und 59 Sekunden, daher für eine Adresse im Durchschnitt 22 Sekunden. Wenn im Internet Explorer eine Meldung mit Benutzereingabe erscheint, kann WEF den Browser nicht

schließen und erzeugt für die nächste Seite eine neue Instanz. Es sollten daher beim Erstellen der virtuellen Maschine alle Standard- und Hinweismeldungen des Internet Explorer, wie etwa am Beginn einer verschlüsselten Verbindung, so eingestellt werden, dass diese nicht mehr erscheinen. Ansonsten benötigt jede Instanz des Browsers unnötig Speicher in der virtuellen Maschine.

Die Verwendung von mehreren virtuellen Maschinen hat hier ein ähnliches Ergebnis wie bei Capture-HPC hervorgerufen. Die Verarbeitungsdauer bei zwei virtuellen Instanzen betrug 25 Minuten und 30 Sekunden, das ist um 31% schneller als mit einer. Die Verwendung einer dritten virtuellen Maschine hat auch hier weder zu einem besseren, noch zu einem schlechteren Ergebnis geführt.

Seiten mit Angriffen bereiten diesem Honeyclient Probleme. Wie schon bei Capture-HPC erwähnt, kann ein Angriff einen Bufferoverflow ausnutzen und der Internet Explorer kann nicht beendet werden. In solchen Fällen stürzt manchmal auch das Browser Control Programm ab und muss manuell neu gestartet werden. Allerdings erhält daraufhin die virtuelle Maschine von der Management Konsole dieselbe Adresse erneut zum Überprüfen zugewiesen. Hier muss die Adresse aus der Datenbank entfernt werden, damit WEF die weiteren Seiten überprüfen kann. Wird jede Adresse einzeln überprüft und die Zeit zusammengezählt, benötigt WEF 3 Minuten und 15 Sekunden für alle Adressen.

Die Ergebnisse der Überprüfung werden in die Datenbank geschrieben. Über die Managementkonsole können diese Informationen ausgelesen werden. Es wird für jede Seite detailliert mitgespeichert, welches Programm Aktionen ausgeführt hat. Die Ergebnisse können auch automatisch aus der Datenbank ausgelesen werden und weiter verarbeitet werden. Dadurch wäre es möglich, einen Alarmierungsmechanismus zu implementieren.

Der Quellcode des Honeyclients ist frei verfügbar und kann an eigene Bedürfnisse angepasst werden.

7.5.3 Monkey Spider

Bei dem Honeyclient Monkey Spider werden die zu untersuchenden Seiten zuerst mit dem Webcrawler Heritrix heruntergeladen und danach wird deren Inhalt mit dem ClamAV Antivirus analysiert.

Die Liste der Adressen wird dem Crawler übergeben, welcher die Inhalte der Adressen lokal speichert. Durch die Verwendung dieses Crawlers ist es möglich, viele verschiedene Einstellungen für das Herunterladen zu aktivieren. Für die Evaluierung des Honeyclients soll der Crawler immer nur die aktuelle Seite mit dem gesamten Inhalt herunterladen. Es ist auch möglich, bestimmte Inhalte wie Videos oder Musikdateien für das Herunterladen auszuschließen. Weiters kann Heritrix auch den Verweisen auf einer Seite folgen und dort Inhalte herunterladen.

In Phase 1 benötigt der Crawler für das Herunterladen der Seiten 5 Minuten und 44 Sekunden. Für die Analyse mit ClamAV benötigt der Honeyclient 70 Sekunden. In Summe sind das 414 Sekunden oder für jede Seite im Durchschnitt 4,14 Sekunden. Es wurde keine Seite fehlerhaft klassifiziert.

Alle Signaturen des Antivirus sind in Phase 2 aktiviert, damit mögliche Angriffe erkannt werden können. Der Crawl- und Scanvorgang benötigte genauso lange wie in Phase 3, nämlich 31 Sekunden. Es wurden alle Seiten als gutartig eingestuft.

Für Phase 3 werden dem Antivirus die Signaturen für die verschiedenen Angriffe entfernt, damit die Voraussetzungen eingehalten werden. Der Crawler benötigt für das Herunterladen der Seiten 14 Sekunden, die Analyse mit ClamAV dauert 17 Sekunden. Dadurch ergibt sich eine Gesamtzeit von 31 Sekunden. Es wurden, wie bei einem low-interaction Honeyclient erwartet, alle Seiten falsch klassifiziert.

Durch den modularen Aufbau des Honeyclients in einzelne Skripte können auch eigene Skripte geschrieben werden. Alle Skripte können angesehen und editiert werden.

Der Honeyclient erzeugt eine Datei, die alle Funde enthält und in der die dazu passende Signatur aufscheint. Außerdem extrahiert er die bösartigen Internetseiten in ein eigenes Verzeichnis. Dadurch können sie im Nachhinein genauer untersucht werden. Wenn man genauere Informationen über einen Angriff erhalten will, muss man die Virendatenbank von ClamAV abfragen. Diese Protokolldatei kann für eine automatische Alarmierung verwendet werden und ab einem Schwellenwert an Funden an den Administrator versendet werden.

7.5.4 SpyBye

SpyBye ist der einzige getestete Honeyclient, bei dem Internetseiten direkt im Browser eingegeben werden können und dort überprüft werden. Die Ergebnisse werden auch direkt im Browserfenster ausgegeben. Allerdings stellt dies eine automatische Überprüfung, wie es in dem Testdurchlauf verwendet wurde, vor einige Probleme, da mehrere Adressen hintereinander oder gleichzeitig analysiert werden müssen. SpyBye bietet aber eine Möglichkeit, direkt eine Adresse anzugeben und diese zu analysieren.

Mit Hilfe eines Skripts konnten alle Internetadressen von Phase 1 dem Browser übergeben werden. Dort überprüft er die Seite und erst wenn alle Seiten getestet worden sind, gilt der Durchlauf als beendet. Damit der Browser von den vielen Adressen nicht überfordert wird, wurde die Liste der Adressen aufgeteilt und die Zeiten einzeln gemessen. Zusammen benötigt der Honeyclient für alle Adressen 9 Minuten und 38 Sekunden. Das ergibt im Durchschnitt eine Zeit von 5 Sekunden pro Adresse. Allerdings hat SpyBye vier Internetseiten nicht überprüft. Bei diesen Seiten wird das Fenster mit den Analyseinformationen geschlossen und der Browser versucht Daten vom Server aufzurufen. Der Browser gelangt in eine Art Endlosschleife, da auch nach einigen Minuten kein Inhalt angezeigt wird, obwohl ein Transfer statt findet. Anscheinend hat der Proxy mit speziellen Internetseiten Probleme, diese herunterzuladen.

Für Phase 2 wird der Webserver, der die Angriffe enthält, in die Liste der schlechten Pattern eingetragen. Außerdem muss der Client mit einer speziellen Option gestartet werden, damit Adressen aus einem privaten Netzwerk überprüft werden. Der Honeyclient erkennt nun alle Seiten als bösartig und markiert sie richtig. Die Analyse benötigt 32 Sekunden.

In Phase 3 wurden für SpyBye sowohl Dateien mit guten als auch schlechten Patterns erstellt. Diese dienen dem Programm als Grundlage für die Analyse. Die Datei mit schlechten Patterns war natürlich leer, während die mit guten Patterns die Adressen des Webservers enthielt. Wie erwartet wurden alle Seiten falsch klassifiziert. Für die Analyse benötigt SpyBye 34 Sekunden. Hier wird wie in Phase 1 so lange gewartet, bis alle Seiten von SpyBye klassifiziert wurden.

Für den Benutzer stellt dieser Honeyclient den optimalen Fall dar. Es können

wie gewohnt die Internetadressen aufgerufen werden.

Erweiterungen können nur direkt im Sourcecode des Honeyclients implementiert werden, welcher allerdings frei zur Verfügung steht. Andere Schnittstellen, an denen Erweiterungen angehängt werden können, gibt es nicht.

Der Honeyclient gibt nur bekannt, welche Inhalte einer Seite bösartig sind und welche als gutartig eingestuft werden. Hat SpyBye keine Informationen über einen Seiteninhalt, wird dieser als unknown markiert. In der Konsole des Honeyclients erscheinen zwar alle Anfragen einer Seite aufgelistet, allerdings wird das Ergebnis der Überprüfung dort nicht angezeigt. Dadurch ist ein automatisiertes Einlesen der Analyseergebnisse nicht möglich.

Eine automatische Alarmierung gestaltet sich aufgrund der Protokollierung in der Konsole genauso problematisch.

7.5.5 HoneyC

Für HoneyC werden zunächst für jede Phase eigene Konfigurationsdateien geschrieben, welche je nach ihrem Inhalt die passenden Adressen und die dazu gehörigen Regeln aufrufen. Als Snort Regeln werden die aktuellen Sourcefire VRT zertifizierten Regeln verwendet, welche die offiziellen Regeln von Snort [92] sind.

In Phase 1 benötigt HoneyC für die Analyse der 100 Internetseiten 18 Minuten und 36 Sekunden oder 11 Sekunden pro Adresse. Es werden alle Seiten richtig klassifiziert.

In Phase 2 werden alle Regeln verwendet und die Seiten mit Angriffen werden analysiert. HoneyC erkennt alle Seiten richtig. Die Analyse benötigt 3 Sekunden pro Adresse.

Für Phase 3 werden jene Snort Regeln entfernt, die Angriffe erkennen könnten. Es soll das Verhalten auf unbekannte Angriffe getestet werden, für die noch keine Signaturen existieren. Hier benötigt der Honeyclient 3 Sekunden für die Analyse aller Seiten. Es wurde kein Angriff gefunden und bei allen Seiten wurde der Status auf clean gesetzt.

Ist der Honeyclient mit dem Queuer, Visitor und der Analysis Engine konfiguriert, müssen nur die einzelnen Adressen in eine Konfigurationsdatei eingetragen werden.

Da HoneyC modular aufgebaut ist, können in jeder Phase eigene Programme und Skripte verwendet werden. Der Client führt je nach Konfigurationsdatei ein bestimmtes Programm aus, um eine Aufgabe durchzuführen.

Während der Analyse wird für jede Seite, auf die eine Snort Regel zutrifft, eine Ausgabe gemacht. Nach der Analyse gibt HoneyC detaillierte Auskünfte über den aktuellen Durchlauf. So wird die Anzahl der geprüften Seiten und die Zeit dafür ausgegeben, sowie die Anzahl der gefunden Übereinstimmungen mit Snort Regeln. Außerdem werden die verschiedenen Content Types aufgelistet und wie viele es davon gegeben hat. Die Größe der einzelnen Content Types wird auch aufgezeigt.

Anhand dieser Ergebnisse kann man eine Alarmierung durchführen. Auch Details zu einzelnen Angriffen können weiter versendet werden und genauer überprüft werden.

7.6 Evaluierung der Verarbeitungsdauer

In diesem Kapitel werden die einzelnen Ergebnisse jedes Honeyclients aus Kapitel 7.5 in Bezug auf die Verarbeitungsdauer zusammengeführt und analysiert. Dadurch können neben allgemeinen Vergleichen zwischen verschiedenen Implementierungen auch Unterschiede zwischen high-interaction und low-interaction Honeyclients herausgearbeitet werden.

Es werden zunächst die einzelnen Phasen verglichen und mögliche Thesen, welche in Kapitel 7 aufgestellt wurden, bestätigt.

7.6.1 Ergebnisse der Phase 1

Die Ergebnisse aus Phase 1 sind in Tabelle 7.2 zusammengefasst. Anhand dieser läßt sich die erste These bestätigen. High-interaction Honeyclients sind langsamer als low-interaction Honeyclients. Weiters erkennt man Unterschiede in den verschiedenen Implementierungen. Diese sind bei high-interaction Honeyclients nicht so groß wie bei low-interaction Honeyclients. Ein möglicher Grund, weshalb MonkeySpider einen so hohen Datendurchsatz besitzt, könnte darin liegen, dass dieser als Crawler Heritrix verwendet. Dieser Crawler wird in großen Projekten eingesetzt, und ist deshalb sehr effizient entwickelt worden. Er verwendet Threads, um

Honeyclient	Gesamte Zeit für Phase 1	Zeit für eine Adresse	Datendurchsatz
Capture-HPC	31'40"	19"	0,76 MB/min
Capture-HPC 2 VM	21'25"	12"	1 MB/min
WEF	36'59"	22"	0,65 MB/min
WEF 2VM	25'30"	15"	0,85 MB/min
SpyBye	9'38"	6"	2,5 MB/min
MonkeySpider	6'54"	4"	3,5 MB/min
HoneyC	20,27"	12"	1,17 MB/min

Tabelle 7.2: Ergebnisse der verschiedenen Honeyclients aus Phase 1. Bei high-interaction Honeyclients werden die Daten sowohl mit einer als auch mit einer zweiten virtuellen Maschine (2 VM) angegeben.

parallel Seiten abarbeiten zu können.

Auch zwischen den high-interaction Honeyclients gibt es Unterschiede bei der Verarbeitungsgeschwindigkeit, allerdings sind diese nicht so eindeutig wie bei low-interaction Honeyclients. Der Unterschied beider liegt nur im verwendeten Rootkit und im Verwaltungsmechanismus der virtuellen Maschinen. High-interaction Honeyclients benötigt die meiste Zeit für das Aufrufen der Seite in der virtuellen Maschine, welches bei beiden Implementierungen ungefähr gleich schnell abläuft.

Die Verwendung von mehreren virtuellen Maschinen bei high-interaction Honeyclients hat in dieser Systemkonfiguration eine signifikante Steigerung der Testergebnisse hervorgerufen. Vermutlich ist eine weitere Steigerung bei der Verwendung weiterer virtueller Maschinen zu erwarten. Allerdings konnte dies auf dem verwendeten Testsystem nicht evaluiert werden, da es zu langsam für mehrere virtuelle Maschinen ist. Dadurch würde sich in Bezug auf die Verarbeitungsgeschwindigkeit der Abstand zu low-interaction Honeyclients deutlich verringern. Allerdings muss beachtet werden, dass es sich hierbei um Seiten handelt, welche keine Angriffe enthalten.

7.6.2 Ergebnisse der Phase 2

In Phase 2 erkennt man signifikante Unterschiede zwischen low-interaction und high-interaction Honeyclients. Capture-HPC und WEF benötigen fünf- bis sechs-

Honeyclient	Gesamte Zeit für Phase 2	Datendurchsatz
Capture-HPC	3’	0,14 MB/min
WEF	3’ 15”	0,13 MB/min
SpyBye	32”	0,78 MB/min
MonkeySpider	32”	0,78 MB/min
HoneyC	3”	8,4 MB/min

Tabelle 7.3: Ergebnisse der verschiedenen Honeyclients aus Phase 2

Honeyclient	Gesamte Zeit für Phase 3	Datendurchsatz
Capture-HPC	3’	0,14 MB/min
WEF	3’ 15”	0,13 MB/min
SpyBye	34”	0,75 MB/min
MonkeySpider	31”	0,81 MB/min
HoneyC	3”	8,4 MB/min

Tabelle 7.4: Ergebnisse der verschiedenen Honeyclients aus Phase 3

mal länger für die Analyse der Seiten als low-interaction Honeyclients. Dies liegt daran, dass nach jeder Seite der Snapshot der virtuellen Maschine neu geladen werden muss. Zwischen den beiden high-interaction Honeyclients besteht bei dieser Testphase beinahe kein Unterschied. Bei den low-interaction Honeyclients liefert HoneyC ein überdurchschnittlich gutes Ergebnis. Gründe für diese Ergebnis konnten keine evaluiert werden.

7.6.3 Ergebnisse der Phase 3

In Phase 3 zeigen sich keine signifikanten Unterschiede bei der Bearbeitungszeit gegenüber Phase 2. Die Ergebnisse sind in Tabelle 7.4 dargestellt. Dadurch kann man schließen, dass low-interaction Honeyclients für Seiten mit und ohne enthaltenen Angriffen gleich lang für die Analyse benötigen. Dieser Schluss kann für high-interaction Honeyclients nicht gezogen werden, da diese bei Seiten mit enthaltenen Angriffen wesentlich länger für die Analyse benötigen.

In dieser Phase erkennen high-interaction Honeyclients die Angriffe, während alle low-interaction Honeyclients die Seiten als gutartig klassifizieren. Dies bestätigt die anfangs angenommene Hypothese.

In Summe sinkt die Verarbeitungsgeschwindigkeit in Phase 2 und 3 im Vergleich zu Phase 1 enorm, ausgenommen bei HoneyC. Die beiden anderen low-interaction Honeyclients sind nur so schnell wie high-interaction Honeyclients in Phase 1. Dies könnte daran liegen, dass in dieser Phase nur Internetseiten ohne großen Inhalt verwendet wurden. So war in den Testseiten nur der Angriff eingebettet. Es könnte daher der Aufbau einer Verbindung überdurchschnittlich mehr Zeit beanspruchen als das Herunterladen der Inhalte.

7.7 Evaluierung der restlichen Vergleichskriterien

Bisher wurde nur die Verarbeitungsgeschwindigkeit als Vergleichskriterium ausgewertet. Allerdings wurden mehrere Kriterien definiert, welche Unterschiede zwischen den verschiedenen Honeyclients ausarbeiten. Eine Zusammenfassung der Ergebnisse jedes Vergleichkriteriums ist in Tabelle 7.5 abgebildet.

Wie schon in der Definition des Kriteriums der Benutzbarkeit beschrieben, soll die Verwendung eines Honeyclients für den Benutzer möglichst transparent sein. Dies konnte am besten SpyProxy umsetzten, da der Benutzer seine gewohnten Arbeitsabläufe nicht umstellen muss. Für jeden anderen Honeyclient muss eine Erweiterung, zum Beispiel ein Proxy Modul, welches in Kapitel 4.5 vorgestellt wurde, implementiert werden. Dadurch können die vom Benutzer eingegebenen Adressen automatisch überprüft werden und ein Ergebnis ausgegeben werden. Hier ist das Kriterium der Erweiterbarkeit und Anpassung wichtig. Es nutzen alle Honeyclients definierte Standardschnittstellen oder Protokolle, welche von einem eigens entwickelten System verarbeitet werden können.

Weiters stellt sich die Frage, ob der Benutzer bei jeder Aktion eine Rückmeldung bekommen soll oder nur bei schadhaften Inhalten. Andere Meldungen könnten nur an den Administrator des Netzwerkes verschickt werden, damit dieser weitere Analysen durchführen kann. Bei SpyProxy erhält der Benutzer immer eine Rückmeldung, welche nicht gefiltert werden kann. Weiters können eigene Module programmiert werden, welche die Alarmierung vornehmen und die gesammelten

Ereignisse filtern.

7.8 Verwendung von Honeyclients in einem universitären Netzwerk

Da der Honeyclient in einem universitären Netzwerk verwendet werden soll, wird dieser möglichst zentral installiert und nicht auf jedem einzelnem System eine Version bereitgestellt. Dadurch muss nur ein System gewartet werden. Bei diesem Punkt stellt sich die Frage, ob ein Honeyclient für das gesamte universitäre Netzwerk verwendet wird oder ob jedes Institut individuell entscheidet, ob es diese Sicherheitseinrichtung nutzt. Natürlich ist zu beachten, dass an ein zentrales System, das den gesamten Universitätbereich abdeckt, wesentlich mehr Benutzer Anfragen schicken als an ein Institutssystem. Dadurch muss auch die Leistung des Systems angepasst werden. Auf Grund der ermittelten Daten ist es sinnvoll, zuerst einen Honeyclient für ein Institut zu installieren und nach einer Evaluierungsphase diesen möglicherweise zentral für das gesamte universitäre Netzwerk einzurichten.

Ist der Honeyclient auf einem System am Institut positioniert, gibt es die Möglichkeit, dass bei einer Häufung von Ereignissen die zentrale Verwaltungseinheit des universitären Netzwerkes informiert wird. Dadurch kann zentral auf bestimmte Angriffe reagiert werden.

Da in einem universitären Netzwerk sehr viele unterschiedliche Systeme mit unterschiedlichen Betriebssystemen und verschiedenen Patchstadien im Einsatz sind, ist der Einsatz eines high-interaction Honeyclients nicht möglich. Dieser kann nur die Sicherheit eines Systems erhöhen, welches er in einer seiner virtuellen Maschinen verwendet. Allerdings gibt es sehr viele unterschiedliche Betriebssysteme in unterschiedlichen Patchstadien. Es wäre zwar möglich, mehrere Honeyclients für die verwendeten Betriebssysteme bereitzustellen, allerdings müsste dies auch für jedes unterschiedliche Patchstadium geschehen. Es ist auch nicht sinnvoll, nur für bestimmte Aktualisierungsstadien eines Betriebssystems, wie etwa Servicepacks bei Microsoft Versionen, Honeyclients anzubieten, da sich gezeigt hat, das auch Softwareaktualisierungen Fehler enthalten können. Diese könnte ein Angreifer verwenden und der Honeyclient kann diesen nicht erkennen.

Kriterium	Capture-HPC	WEF	SpyBye	Monkey-Spider	HoneyC
Geschwindigkeit	∼	∼	+	+	∼
Benutzbarkeit	∼	∼	+	∼	∼
Auswertbarkeit	+	+	-	+	+
Alarmierung	+	+	-	+	+
Erweiterbarkeit	+	+	-	+	+

Tabelle 7.5: Die Ergebnisse aller Vergleichskriterien. Das Symbol - bedeutet, dass das jeweilige Kriterium nicht oder sehr schlecht umgesetzt wurde, ein ∼ bedeutet, dass es grundsätzlich umgesetzt wurde, allerdings gibt es bessere Implementierungen. Ein + Symbol bedeutet, dass das jeweilige Kriterium sehr gut umgesetzt wurde.

In einem Firmennetzwerk, welches meistens einheitliche Systeme verwendet, wäre der Einsatz eines high-interaction Honeyclients möglich.

Für die Installation eines Honeyclients in einem universitären Netzwerk bietet sich ein Modell mit mehreren Phasen an. In jeder Phase soll die aktuell verwendete Konfiguration evaluiert werden und es kann ein Umstieg auf eine andere Phase erörtert werden. Die vorgestellten Phasen sind:

Phase 1: Die Verwendung von SpyBye. Dieser Honeyclient ist sehr schnell konfiguriert und bietet als einziger ein Proxy Modul. Allerdings müssen die verwendeten Signaturen überprüft werden, ob sie den Anforderungen entsprechen oder möglicherweise erweitert werden müssen. Es können zusätzlich zum Beispiel Blacklists für bösartige Seiten eingearbeitet werden.

Ein weiterer Nachteil dieses Honeyclients ist, dass nicht zentral protokolliert werden kann, welche Meldungen aufgetreten sind. Dadurch besitzt der Administrator des Netzwerkes keine Daten über den Wirkungsgrad des Systems.

Phase 2: In nächster Instanz kann man die Inbetriebnahme von Monkey Spider evaluieren. Dieser bietet den Vorteil, dass die verwendeten Signaturen automatisch aus dem Internet nachgeladen werden können. Allerdings muss für diesen ein Proxy Modul geschrieben werden.

Die Kosten eines Honeyclients in Bezug auf Hardware sind je nach benötigter

Rechnerausstattung unterschiedlich. Weiters muss man die Kosten für die Installation und Wartung eines Honeyclients betrachten. Da Honeyclients eine sehr neue Technologie sind haben die meisten Administratoren keine Erfahrung mit der Inbetriebnahme eines Honeyclients, wodurch deren Einschulung kostenintensiv und zeitaufwändig ist. Außerdem müssen die Kosten für die Implementierung eines Proxy Moduls einberechnet werden.

Die laufenden Kosten für die Wartung eines Honeyclients werden sehr ähnlich denen anderer Netzwerkkomponenten sein, wie etwa einer Firewall. Die Ergebnisse müssen regelmäßig kontrolliert werden und bei Bedarf muss auf diese reagiert werden. Die Administration eines Institutsnetzwerks gestaltet sich oft schwierig, da die Tätigkeit des Netzwerkadministrators häufig nur nebenbei ausgeübt wird und es vielfach keine Regelungen der Zuständigkeiten bei Urlaub oder Krankheitstand gibt.

Alternativ zu einem low-interaction Honeyclient kann man in einem universitären Netzwerk auch ein IDS verwenden. Die Unterschiede zwischen den beiden Systemen wurden in Kapitel 4.6 beschrieben. Es muss allerdings evaluiert werden, welches der beiden Systeme in Bezug auf die Verarbeitungsgeschwindigkeit Vorteile bietet, da dies ist ein sehr wichtiges Kriterium für ein universitären Netzwerk darstellt.

8 Zusammenfassung und Ausblick

Heutzutage sind viele Angriffe Teil von organisiertem Verbrechen. Dessen Ziel ist es, Malware auf dem Rechner der Opfer zu installieren oder diesen sensible Informationen, wie etwa Kontodaten, zu stehlen. Die Situation für kriminelle Angreifer hat sich jedoch in den letzten Jahren verschlechtert, da etliche Angriffswege durch Sicherheitsmaßnahmen wie Firewalls und Virenscanner nicht mehr so einfach und effizient durchgeführt werden können, wie das noch vor einigen Jahren der Fall war. Deshalb hat sich eine neue Methode für Angriffe, die so genannten client-seitigen Angriffe, entwickelt, die sich immer mehr durchsetzt. Damit ist es möglich, gezielt Schwachstellen der Clientsoftware auszunutzen und durch den Aufruf einer entfernten Ressource Malware auf einem System zu installieren. Dieser Angriff ist besonders hinterhältig, da er vom User völlig unbemerkt durchgeführt werden kann.

Während sich für die Erforschung server-seitiger Angriffe Honeypots als Werkzeuge etabliert haben, ist diese Technik nicht für client-seitige Angriffe einsetzbar. Da ein Detektionswerkzeug für diese Angriffsart jedoch äußerst wichtig ist, um die Computersicherheit zu erhöhen, wurden dafür so genannte Honeyclients entwickelt. Diese neu entwickelte Technik gibt es, wie Honeypots, in verschiedenen Ausführungen, die jeweils in einigen Implementierungen existieren. Da Honeyclients ein sehr neues Forschungsgebiet sind, ist in der Literatur bisher relativ wenig über Vergleiche zwischen den einzelnen Systemen und Implementierungen bekannt. Die vorliegende Arbeit bietet daher einen Beitrag zu einem besseren Überblick über die einzelnen Systeme und evaluiert diese in Hinblick auf ihre Eignung für eine bestimmte Problemstellung.

Als ein besonderer Anwendungsbereich von Honeyclients wurde in dieser Arbeit ein universitäres Netzwerk betrachtet. Jedes Netzwerk stellt spezielle Anforderungen an einen Honeyclient und soll durch dessen Verwendung nicht gestört werden.

In dieser Arbeit wurde überprüft, ob Honeyclients die Sicherheit in einem solchen Netzwerk erhöhen können und wie eine mögliche Konfiguration für diesen Anwendungsbereich aussehen könnte. Für die Evaluierung von Honeyclients wurden verschiedene Szenarien ausgearbeitet und einige Implementierungen untersucht.

Dabei hat sich gezeigt, dass es gerade in einem universitären Netzwerk sinnvoll ist, einen Honeyclient einzusetzen, da dieser die Sicherheit signifikant erhöht. Durch diese Technik können Angriffe erkannt werden, welche bisher nur schwer zu identifizieren waren. Aufgrund der steigenden Zahl von client-seitigen Angriffen ist dies zunehmend von großer Bedeutung, um Schäden in universitären Netzwerken auch in Zukunft auf ein Minimum beschränken zu können.

Erwartungsgemäß waren die verschiedenen Implementierungen von Honeyclients in der Lage, ihre Anforderungen, wie aus der Literatur bekannt, zu erfüllen. Bemerkenswert ist jedoch, dass im Rahmen der vorliegenden Diplomarbeit auch sehr interessante und unerwartete Ergebnisse beim Vergleich der verschiedenen Techniken auftraten. Besonders überraschend war etwa die Erkenntnis, dass high-interaction Honeyclients unter gewissen Rahmenbedingungen gleich schnell operieren können, wie low-interaction Honeyclients.

Auffällig bei der Analyse der Honeyclients auf ihre Eignung für ein universitäres Netzwerk war, dass zurzeit die Mehrzahl der Implementierungen für Forschungsarbeiten und nicht für die Benutzung als Detektionswerkzeug für einzelne Websites ausgelegt sind. Es ist bei diesen Systemen nicht möglich, diese ohne eigens programmierte Erweiterungen für eine automatisierte Analyse aller aufgerufenen Internetseiten in einem Netzwerk zu verwenden. Da client-seitige Angriffe jedoch eine wachsende Bedrohung für die Computersicherheit von Netzwerken, insbesondere dem Internet, darstellen, wird es in Zukunft unbedingt nötig sein, Honeyclients in diese Richtung zu entwickeln. Derzeit ist es lediglich möglich, die Schnittstellen zwischen den einzelnen Teilen der Honeyclients zu benutzen, um Erweiterungen zu implementieren, die eine Überprüfung aller in einem Netzwerk aufgerufenen Internetseiten zu ermöglichen. Hier sollte man in Zukunft weitere Projekte in Betracht ziehen, um solche Erweiterungen zu implementieren und auf ihre Benutzerfreundlichkeit hin zu evaluieren. Dies würde den Einsatz der Honeyclients als Sicherheitswerkzeug in einem Netzwerk ermöglichen und damit die Sicherheit in Netzwerken signifikant erhöhen.

Derzeit wird der Netzwerkverkehr meist mit Hilfe von IDS untersucht, die durch die Verwendung geeigneter Signaturen, client-seitige Angriffe ebenfalls erkennen können. Daher stellen sie sehr ähnliche Funktionen wie low-interaction Honeyclient zur Verfügung. Es ist daher nötig, in weiteren Projekten zu evaluieren, welches der Systeme für ein universitäres Netzwerk besser geeignet und sinnvoller ist. Die Ergebnisse dieser Arbeit weisen jedoch darauf hin, dass im Moment IDS, auf Grund der Verarbeitungsdauer und Administrierbarkeit, für den zentralen Einsatz noch besser geeignet sind.

Zusammenfassend lässt sich sagen, dass Honeyclients eine sehr interessante und spannende neue Technik darstellen, bei der allerdings noch einige Details verbessert werden können. Da die Verwendung der Systeme von möglichst vielen und auch unerfahrenen Benutzern das Ziel darstellen soll, müssen noch einige Verbesserungen vorgenommen werden, bevor Honeyclients als umfassendes Werkzeug zur Verbesserung der Netzwerksicherheit genutzt werden können.

Literaturverzeichnis

[1] Ahlheim Karl-Heinz : *Meyers großes Lexikon.* Bibliographisches Institut and F.A. Brockhaus, 1980.

[2] AlephOne: *Smashing the stack for fun and profit.* Phrack 7(49), 1996.

[3] AOL und NCSA: *Online Safety Study.* `http://www.staysafeonline.info/pdf/safety_study_2005.pdf`.

[4] Bächer, Paul, Kötter Markus, Holz Thorsten, Dornseif Maximillian und Freiling Felix C.: *The Nepenthes Platform: An Efficient Approach to Collect Malware.* In: *RAID*, Seiten 165–184, 2006.

[5] Badhusha, Akbar, Buhari Seyed, Junaidu Sahalu und Saleem Mohammed: *Automatic signature files update in antivirus software using active packets.* Computer Systems and Applications, ACS/IEEE International Conference on. 2001, Seiten 457–460, 2001.

[6] Bailey, Michael, Jon Oberheide, Jon Andersen, Zhuoqing Morley Mao, Farnam Jahanian und Jose Nazario: *Automated Classification and Analysis of Internet Malware.* In: *RAID*, Seiten 178–197, 2007.

[7] Boehm, Barry W.: *Software Engineering Economics.* Prentice Hall, Englewood Cliffs, NJ, 1981.

[8] Boldt, Martin, Bengt Carlsson und Jacobsson Andreas: *Exploring Spyware Effects.* In: *NordSec 2004: Proceedings of the Eighth Nordic Workshop on Secure IT Systems*, 2004.

[9] BSI: *IT-Grundschutz Katalog.* `http://bsi.bund.de/gshb/deutsch/index.html`.

[10] Carpenter, Matthew, Liston Tom und Skoudis Ed: *Hiding Virtualization from Attackers and Malware*. IEEE Security and Privacy, 5(3):62–65, 2007.

[11] Casavant, T.L. und McMillin B.M.: *Safe computing*. Potentials, IEEE, 8(3):29–31, Oct 1989.

[12] Cohen, Frederick B.: *A short course on computer viruses (2nd ed.)*. John Wiley & Sons, Inc., New York, NY, USA, 1994.

[13] Cova, Marco, Christopher Kruegel und Giovanni Vigna: *There Is No Free Phish: An Analysis of "Free" and Live Phishing Kits*. In: *Proceedings of the USENIX Workshop on Offensive Technologies*, 2008.

[14] Crispin, Cowan, Beattie Stevie, Johansen John und Wagle Perry: *PointGuard^{TM}: Protecting Pointers from Buffer Overflow Vulnerabilities*. In: *Proc. of the 12th Usenix Security Symposium*, Aug 2003.

[15] Determina Security Research: *Windows Animated Cursor Stack Overflow Vulnerability*. `http://www.determina.com/security.research/vulnerabilities/ani-header.html`.

[16] Dingle, Adam und Pártl Tomáš: *Web cache coherence*. In: *Proceedings of the fifth international World Wide Web conference on Computer networks and ISDN systems*, Seiten 907–920, Amsterdam, The Netherlands, The Netherlands, 1996. Elsevier Science Publishers B. V.

[17] Donnaberger, Elisabeth und Kainrath Johann: *Firewall und Internet Security an der TU Wien*. `http://www.zid.tuwien.ac.at/zidline/zl01/firewall.html`, 1999.

[18] Dornseif, Maximillian, Holz Thorsten und Klein Christian: *NoSEBrEaK - Attacking Honeynets*. 5th Annual IEEE Information Assurance Workshop, 2004.

[19] Dragovic, Boris, Fraser Keir, Harris Tim, Ho Alex, Pratt Ian, Warfield Andrew, Barham Paul und Neugebauer Rolf: *Xen and the*

Art of Virtualization. In: *Proceedings of the ACM Symposium on Operating Systems Principles*, October 2003.

[20] Edelman, Benjamin und Rosenbaum Hannah: *The Safety of Internet Search Engines*. `http://www.siteadvisor.com/studies/search_safety_may2006.html`.

[21] Freiling, Felix C., Holz Thorsten und Wicherski Georg: *Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks*. In: *ESORICS*, Seiten 319–335, 2005.

[22] Goebel, Jan, Thorsten Holz und Carsten Willems: *Measurement and Analysis of Autonomous Spreading Malware in a University Environment*. In: *DIMVA*, Seiten 109–128, 2007.

[23] Gollmann Georg: *Firewalls und externe Netze*. `http://www.zid.tuwien.ac.at/zidline/zl08/ports.html`, 2003.

[24] Holz, Thorsten und Raynal Frederic: *Detecting honeypots and other suspicious environments*. Proceedings of the 2005 IEEE Workshop on Information Assurance and Security, 2005.

[25] Holz, Thorsten und Wicherski Georg: *Effektives Sammeln von Malware mit Honeypots*. DFN-CERT, Proceedings of 13th DFN-CERT Workshop, 2006.

[26] Holz Thorsten: *Learning more about attack patterns with honeypots*. Sicherheit 2006: Sicherheit - Schutz und Zuverlässigkeit, 2006.

[27] Ikinci, Ali, Thorsten Holz und Felix C. Freiling: *Monkey-Spider: Detecting Malicious Websites with Low-Interaction Honeyclients*. In: Alkassar, Ammar und Joerg H. Siekmann (Herausgeber): *Sicherheit*, Band 128 der Reihe *LNI*, Seiten 407–421. GI, 2008.

[28] Jaitner, Ingmar und Husinsky Irmgard: *IT Security, ein Praxisbericht*. `http://www.zid.tuwien.ac.at/zidline/zl15/security.html`.

[29] Kalafut, Andrew, Acharya Abhinav und Gupta Minaxi: *A study of malware in peer-to-peer networks.* In: *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, Seiten 327–332, New York, NY, USA, 2006. ACM.

[30] Kappes Martin: *Netzwerk- und Datensicherheit.* Teubner, 2007.

[31] Kreibich, Christian und Jon Crowcroft: *Honeycomb: creating intrusion detection signatures using honeypots.* SIGCOMM Comput. Commun. Rev., 34(1):51–56, 2004.

[32] Kriha, Walter und Schmitz Roland: *Internet-Security aus Software-Sicht: Ein Leitfaden zur Software-Erstellung für sicherheitskritische Bereiche (Xpert.press).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[33] Lawrence, Steve und Giles Lee C.: *Searching the World Wide Web.* Science, 280(5360):98-100, 1998.

[34] Lemos Robert: *Microsoft path opens user to attack.* `http://www.securityfocus.com/news/11408`, August 2006.

[35] Lucca, G. A. Di, A. R. Fasolino, M. Mastoianni und P. Tramontana: *Identifying Cross Site Scripting Vulnerabilities in Web Applications.* wse, 0:71–80, 2004.

[36] Ma, Justin, Dunagan John, Wang Helen J., Savage Stefan und Voelker Geoffrey M.: *Finding diversity in remote code injection exploits.* In: *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, Seiten 53–64, New York, NY, USA, 2006. ACM.

[37] Martin, Michael und Monica S. Lam.: *Automatic Generation of XSS and SQL Injection Attacks with Goal-Directed Model Checking.* In: *Proceedings of the 17th USENIX Security Symposium*, 2008.

[38] Meyna Arno and Peters Olaf H.: *Handbuch der Sicherheitstechnik.* Carl Hanser Verlag München, 1985.

[39] Microsoft: *MS Security Bulletin MS06-057*. http://www.microsoft.com/technet/security/bulletin/MS06-057.mspx.

[40] Microsoft: *Vulnerabilities in GDI Could Allow Remote Code Execution*. http://www.microsoft.com/technet/security/Bulletin/MS07-017.mspx.

[41] Microsoft: *Vulnerability in Cursor and Icon Format Handling Could Allow Remote Code Execution*. http://www.microsoft.com/technet/security/Bulletin/MS05-002.mspx.

[42] Microsoft: *Vulnerability in the Microsoft Data Access Components (MDAC) Function Could Allow Code Execution*. http://www.microsoft.com/technet/security/bulletin/ms06-014.mspx.

[43] Microsoft: *Vulnerability in Vector Markup Language Could Allow Remote Code Execution*. http://www.microsoft.com/technet/security/bulletin/MS06-055.mspx.

[44] Microsoft: *Microsoft Security Bulletin MS06-001: Vulnerability in Graphics Rendering Engine Could Allow Remote Code Execution*. http://www.microsoft.com/technet/security/bulletin/MS06-001.mspx, 2006.

[45] Mölsä, Jarmo: *Mitigating denial of service attacks: a tutorial*. J. Comput. Secur., 13(6):807–837, 2005.

[46] Moore, David, Shannon Colleen und Brown Jeffery: *Code-Red: a case study on the spread and victims of an internet worm*. In: *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, Seiten 273–284, New York, NY, USA, 2002. ACM.

[47] Moshchuk, Alex, Bragin Tanya, Gribble Steven D. und Levy Henry M.: *A Crawler-based Study of Spyware in the Web*. In: *NDSS*, 2006.

[48] Moshchuk, Alexander, Bragin Tanya, Deville Damien, Gribble Steven D. und Levy Henry M.: *SpyProxy: execution-based detection*

of malicious web content. In: *SS'07: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, Seiten 1–16, Berkeley, CA, USA, 2007. USENIX Association.

[49] Nunes, Dan und Shane Keats: *Mapping the Mal Web.* `http://www.siteadvisor.com/studies/map_malweb_mar2007.html`.

[50] Provos, Niels, Panayiotis Mavrommatis, Moheeb A. Rajab und Fabian Monrose: *All Your iFRAMEs Point to Us.* Technischer Bericht, Google Inc, 1600 Amphitheatre Parkway, Mountain View, CA, February 2008.

[51] Provos, Niels und Holz Thorsten: *Virtual Honeypots: From Botnet Tracking to Intrusion Detection.* Addison Wesley Professional, 2007.

[52] Rieck, Konrad, Thorsten Holz, Carsten Willems, Patrick Duessel und Pavel Laskov: *Learning and Classification of Malware Behavior.* In: Zamboni, Diego (Herausgeber): *DIMVA*, Band 5137 der Reihe *Lecture Notes in Computer Science*, Seiten 108–125. Springer, 2008.

[53] Rocaspana Joan R.: *SHELIA: A Client Honeypot for Client-side Attack Detection.* 2007.

[54] Saroiu, Stefan, Gribble Steven D. und Levy Henry M.: *Measurement and analysis of spywave in a university environment.* In: *NSDI'04: Proceedings of the 1st conference on Symposium on Network+ed Systems Design and Implementation*, Seite 11, Berkeley, CA, USA, 2004. USENIX Association.

[55] Schneier Bruce: *Secret & Lies: IT-Sicherheit in einer vernetzten Welt.* dpunkt Verlag, 2004.

[56] Seifert, Christian, Welch Ian und Komisarczuk Peter: *HoneyC - The Low-Interaction Client Honeypot.* 2006.

[57] Seifert, Christian, Steenson Ramon und Holz Thorsten: *Know Your Enemy:Malicious Web Servers.* 2007.

[58] Skoudis Ed: *Counter hack: A step-by-step guide to computer attacks and effective defense.* Prentice Hall PTR, 2002.

[59] Smith, J. E. und Ravi Nair: *The architecture of virtual machines*. Computer, 38(5):32–38, 2005.

[60] The Honeynet Project: *Know Your Enemy: Defining Virtual Honeynets*. http://www.honeynet.org/papers/virtual/index.html, 2003.

[61] The Honeynet Project: *Know Your Enemy: Sebek: A kernel based data capture tool*. http://www.honeynet.org/papers/sebek.pdf, 2003.

[62] The Honeynet Project: *Know your Enemy: Honeywall CDROM Eeyore*. http://www.honeynet.org/papers/cdrom/eeyore/index.html, 2004.

[63] The Honeynet Project: *Know Your Enemy: Honeynets*. http://www.honeynet.org/papers/honeynet/index.html, 2006.

[64] TU Wien, ZID der: *Einbindung von TUNET in nationale und internationale Netze*. http://www.zid.tuwien.ac.at/kom/tunet/tunet_ein_begriff/einbindung/.

[65] TU Wien, ZID der: *Security Policy der TU Wien*. http://www.zid.tuwien.ac.at/sts/security/policy/security_policy_deutsch/.

[66] TU Wien, ZID der: *TUNET - Historische Entwicklung*. http://www.zid.tuwien.ac.at/kom/tunet/tunet_ein_begriff/historische_entwicklung/.

[67] TU Wien, ZID der: *TUNET - Kenndaten*. http://www.zid.tuwien.ac.at/kom/tunet/tunet_ein_begriff/kenndaten_des_netzes/.

[68] TU Wien, ZID der: *Umfrage über Soft- und Hardwareausstattung von Studentenrechnern*. http://www.zid.tuwien.ac.at/zidline/zl16/umfrage/.

[69] US Cert: *US Cert Vulnerability Note VU#753044*. http://www.kb.cert.org/CERT_WEB/services/vul-notes.nsf/0/390162e264d23d2d852571f600720a9c?OpenDocument.

[70] US-Cert: *Vulnerability Note VU#181038: Microsoft Windows Metafile handler SETABORTPROC GDI Escape vulnerability*. http://www.kb.cert.org/vuls/id/181038.

[71] Verbowski, Chad, Emre Kiciman, Arunvijay Kumar, Brad Daniels, Shan Lu, Juhan Lee, Yi-Min Wang und Roussi Roussev: *Flight data recorder: monitoring persistent-state interactions to improve systems management.* In: *USENIX'06: Proceedings of the 7th conference on USENIX Symposium on Operating Systems Design and Implementation*, Seiten 9–9, Berkeley, CA, USA, 2006. USENIX Association.

[72] Vogt, Philipp, Nentwich Florian, Jovanovic Nenad, Kirda Engin, Kruegel Christopher und Vigna Giovanni: *Cross Site Scripting Prevention with Dynamic Data Tainting and Static Analysis.* February 2007.

[73] Weaver, Nicholas, Paxson Vern, Staniford Stuart und Cunningham Robert: *A taxonomy of computer worms.* In: *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malcode*, Seiten 11–18, New York, NY, USA, 2003. ACM.

[74] Xnos Lab: *WEF System Architecture.* `http://www.xnos.org/security/web-exploit-finder/system-architecture.html`.

[75] Xu, Jun und Nithin Nakka: *Defeating Memory Corruption Attacks via Pointer Taintedness Detection.* In: *DSN '05: Proceedings of the 2005 International Conference on Dependable Systems and Networks*, Seiten 378–387, Washington, DC, USA, 2005. IEEE Computer Society.

[76] Yi-Min, Wang, Beck Doug und Jiang Xuxian: *Automated web patrol with Strider HoneyMonkeys.* NDSS06, 2005.

[77] Zhang, Yin und Paxson Vern: *Detecting backdoors.* In: *SSYM'00: Proceedings of the 9th conference on USENIX Security Symposium*, Seite 12, Berkeley, CA, USA, 2000. USENIX Association.

[78] Zhuge, Jianwei, Thorsten Holz, Xinhui Han, Chengyu Song und Wei Zou: *Collecting Autonomous Spreading Malware Using High-Interaction Honeypots.* In: *ICICS*, Seiten 438–451, 2007.

[79] Zimmer Dennis: *VMWare & Microsoft Virtual Server.* Galileo Press, 2005.

[80] Zou, C., Towsley D. und Gong W.: *Email virus propagation modeling and analysis*, 2003.

Die Internetadressen wurden am 22.9.2008 auf ihre Aktualität überprüft.

Verzeichnis der Projekte und Tools

[81] *AdAware*. http://www.lavasoft.de/software/adaware/.

[82] *Chkrootkit*. http://www.chkrootkit.org/.

[83] *Clam Antivirus*. http://www.clamav.net/.

[84] *Heritrix*. http://crawler.archive.org/.

[85] *The Honeynet Project*. http://www.honeynet.org/.

[86] *Honeywall*. http://www.honeynet.org/tools/cdrom/.

[87] *Internet Archive*. http://www.archive.org/index.php.

[88] *IrfanView*. http://www.irfanview.com/.

[89] *Kazaa*. http://www.kazaa.com/.

[90] *Limewire*. http://www.limewire.com/.

[91] *Microsoft Virtual PC*. http://www.microsoft.com/windows/products/winfamily/virtualpc/default.mspx.

[92] *Snort*. http://www.snort.org/.

[93] *SpyBye*. http://www.spybye.org/.

[94] *Squid*. http://www.squid-cache.org/.

[95] *Ubuntu*. http://www.ubuntu.com/.

[96] *User-mode Linux*. http://user-mode-linux.sourceforge.net/.

[97] *WEBrick*. http://www.webrick.org/.

[98] *XnView*. http://www.xnview.de/.

[99] *Yahoo Search API*. http://developer.yahoo.com/search/.

[100] Action, Inform: *NoScript*. http://noscript.net/.

[101] Alexa: *Top Sites Austria*. http://www.alexa.com/site/ds/top_sites?cc=AT&ts_mode=country&lang=none.

[102] Mitre Honeyclient Project: *User Guide*. http://www.honeyclient.org/trac/wiki/UserGuide/.

[103] Mozilla: *Phishing Protection*. http://www.mozilla.com/en-US/firefox/phishing-protection/.

[104] VMWare Inc.: *VMWare*. http://www.vmware.com/.

Die Internetadressen wurden am 22.9.2008 auf ihre Aktualität überprüft.

Abbildungsverzeichnis

Tabellenverzeichnis