

# TrumanBox - Transparente Emulation von Internetdiensten

Christian Gorecki  
Universität Mannheim  
Lehrstuhl Praktische Informatik I  
gorecki@informatik.uni-mannheim.de

## Zusammenfassung

Mit dem Aufkommen der dynamischen Analyse von Schadsoftware, bei der bösartige Programme verhaltensbasiert analysiert und somit ausgeführt werden, stellt sich die Frage, wie ausgehende Verbindungsversuche behandelt werden sollen. Unterbindet man diese, verschlechtern sich in der Regel die Analyseergebnisse, erlaubt man sie, so nimmt man eine eventuelle Beeinträchtigung weiterer Systeme in Kauf. In dieser Arbeit stellen wir einen Ansatz vor, mit dem durch eine transparente Emulation von Internetdiensten ein besser skalierbares Vorgehen ermöglicht wird. Anstatt Verbindungen zum Internet während der Analyse zu unterbinden, können durch das hier vorgestellte System Verbindungsversuche auf generische Dienste umgeleitet werden. Dabei werden die unterstützten Dienste Port-unabhängig bedient. Das System, namens *TrumanBox*, kann in verschiedenen Umgebungen, insbesondere in Kombination mit dynamischen Analyseplattformen, eingesetzt werden.

## 1 Einleitung

Bösartige Programme, auch Schadsoftware oder Malware genannt, sind heutzutage beinahe überall im Internet anzutreffen und es werden täglich mehr. Um etwas gegen diese unerwünschten Programme unternehmen zu können, bedarf es gründlicher Analysen der entsprechenden ausführbaren Dateien. In der Vergangenheit wurde Schadsoftware in erster Linie mittels statischer Verfahren, wie *reverse engineering* und Untersuchung des Quellcodes, analysiert. Obwohl diese Vorgehensweise gute Ergebnisse liefert, ist sie nicht sehr performant und somit nicht geeignet, um neu auftretende Schadsoftware adäquat und zeitnah zu verarbeiten. Aus diesen Gründen wird die Nachfrage nach Werkzeugen zur sogenannten *dynamischen Analyse* immer größer.

Bei der dynamischen Analyse werden potentiell bösartige Programme in einer Analyseumgebung ausgeführt. Diese verhaltensbasierte Analyse kann im Allgemeinen wesentlich schnell-

ler durchgeführt werden und ermöglicht es daher dem hohen Aufkommen an Schadsoftware angemessener entgegenzutreten zu können. Die so gewonnenen Informationen können in einem Protokoll aufgezeichnet werden und als Signatur interpretiert, bei der Unterscheidung von böartigen Programmen weiterhelfen. Das Ausführen von Schadsoftware, auch in einer speziell präparierten Umgebung, muss mit äußerster Vorsicht geschehen. Da viele dieser Programme so programmiert sind, dass sie versuchen sich eigenständig zu verbreiten, müssen gegebenenfalls Vorkehrungen zur Unterbindung dieser Ausbreitungsversuche unternommen werden. Der triviale Ansatz hierbei wäre das Verbot von ausgehender Netzwerkkommunikation, beziehungsweise die gesamte Analyseumgebung ohne eine Anbindung an weitere Netzwerke, wie zum Beispiel das Internet, zu betreiben. Da eine Schadsoftware üblicherweise zuerst testet, ob sie über eine Internetverbindung verfügt, können viele Verhaltensmuster nur in Kombination mit einer Anbindung an das Internet beobachtet werden. Zum Beispiel sind viele der Funktionen eines typischen Bots oft erst zu beobachten, nachdem er sich erfolgreich zu seinem *command & control* (C&C) Server verbunden und von diesem Instruktionen erhalten hat. In einem solchen Fall, wäre es wünschenswert, den entsprechenden Verbindungsaufbau zu ermöglichen. Beide Ansätze, sowohl volle Konnektivität, als auch strikte Unterbindung jeglichen Datenverkehrs, können auf triviale Art und Weise umgesetzt werden.

Einen dritten Ansatz in diesem Bereich stellen *Honeywalls* dar, welche ein wesentlich genaueres Ausloten zwischen den kontrahierenden Kriterien - Vermeiden der Beeinträchtigung von Systemen Unbeteiligter und gute Ergebnisse bei der dynamischen Analyse - ermöglichen. Zu diesem Zweck wird eine Honeywall, in der Regel als transparente Netzwerkbrücke (Bridge), zwischen die zu schützende Netzwerkstruktur (z.B. das Internet) und die kritische Netzwerkinfrastruktur (z.B. eine dynamische Analysestation oder auch ein Honeypot/-net) geschaltet. Dort wirkt sie dann ähnlich wie eine „umgekehrte“ Firewall: jeglicher, aus dem Internet kommender Datenverkehr wird durchgelassen, während die ausgehende Kommunikation stark eingeschränkt wird, um so gegebenen Vorgaben bezüglich Sicherheit gerecht zu werden. Auf diese Weise kann mittels Honeywalls, auf sehr flexible Art und Weise, zwischen „kein Risiko“ und „gute Analyseergebnisse“ skaliert werden. Somit sollte eine Qualität der Ergebnisse bei der dynamischen Malwareanalyse entsprechend der Abbildung 1 zu erwarten sein.

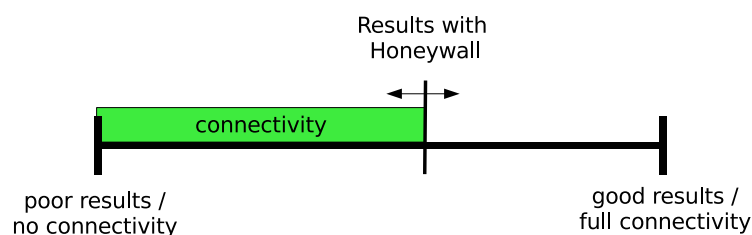


Abbildung 1: Erwartete Qualität der Ergebnisse bei der dynamischen Analyse von Malware

## 2 Herausforderung

Motiviert durch die guten Ergebnisse, die bei uneingeschränkter Internetverbindung von dynamischen Analysesystemen bereits erreicht werden können und dem Bestreben keine Systeme Unbeteiligter zu beeinträchtigen, entstand die Idee Internetdienste zu emulieren. Um unsere

Idee zu veranschaulichen stellen wir uns ein Szenario wie folgt vor. Angenommen wir haben ein dynamisches Analysesystem, welches über eine transparente Netzwerkbrücke direkt mit dem Internet verbunden ist. Durch den Einsatz einer Honeywall-Implementierung auf der Netzwerkbrücke, können wir bösartige, ausgehende Verbindungen unterbinden, indem die entsprechenden Datenpakete verworfen werden. Entsprechend würden Datenverbindungen, die eine auf dem Analysesystem laufende Schadsoftware herstellen möchte, nicht zu Stande kommen und somit könnten weitere Aktivitäten nicht beobachtet beziehungsweise protokolliert werden. Um auch diese Aktivitäten zu erkennen werden Datenpakete, die nicht in das Internet gelangen sollen, in unserem System nicht einfach verworfen, sondern auf entsprechende, lokal laufende Dienste transparent umgeleitet. Diese akzeptieren alle Verbindungsanfragen, soweit der entsprechende Dienst unterstützt wird. Der Unterschied zwischen den beiden Ansätzen, Honeywall und Emulation, wird in Abbildung 2 veranschaulicht.

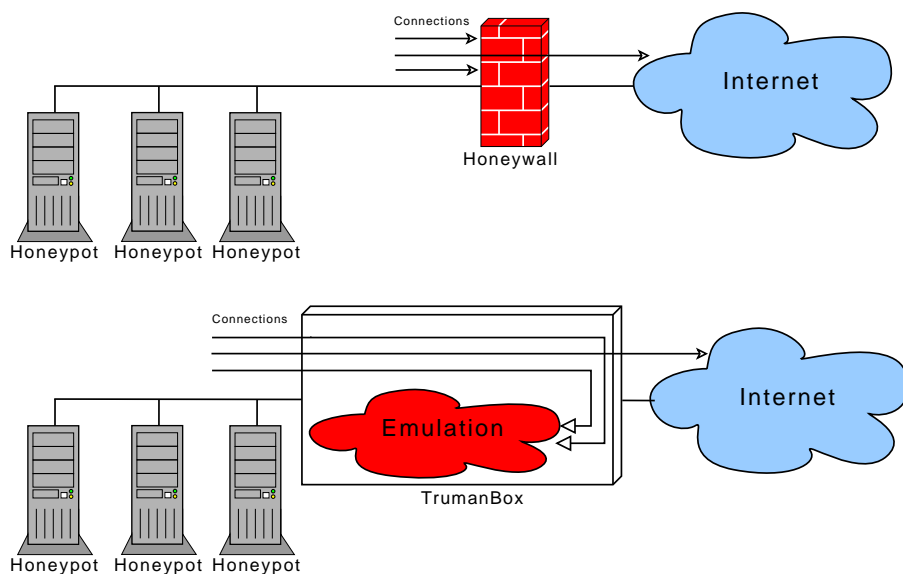


Abbildung 2: Unterschied zwischen Honeywall (oben) und TrumanBox (unten)

## 2.1 Umlenken der Verbindungsanfragen

Damit der Client keinen Verdacht schöpft, muss die Umleitung von Verbindungsanfragen transparent geschehen. Hierfür eignet sich die Kombination der Techniken *Bridging* (siehe Abschnitt 3.1) und Paketumleitung in OSI Level 2. Sobald die Verbindungsanfrage des Clients angenommen wurde können wir unter Umständen übermittelte Datenpakete beobachten, die uns Aufschluss über den jeweils angeforderten Netzwerkdienst geben. Auf diese Weise können wir zum Beispiel HTTP- oder IRC-Anfragen beobachten, die uns beim Verwerfen der Pakete entgangen wären. Um sinnvolle Antworten auf Client-Anfragen generieren zu können - einen Spezialfall stellen hierbei Protokolle wie SMTP und FTP dar, bei denen der Client nach dem Verbindungsaufbau auf ein initiales Datenpaket vom Server wartet - ist die Identifikation des verwendeten Protokolls notwendig. Traditionell wird das Protokoll einer TCP- oder UDP-Verbindung über den Zielport der Verbindungsanfrage bestimmt. Demnach sollte das Bestimmen des verwendeten Protokolls einer Verbindung nicht weiter schwierig sein. Weiter

wäre ein einfaches Umlenken der ins Internet gerichteten Verbindungsanfragen auf unser lokales System völlig ausreichend. Wir müssten lediglich garantieren, dass entsprechende Dienste lokal auf ihren Standardports lauschen.

## 2.2 Identifizierung des Anwendungsprotokolls

Leider kann man gerade bei bösartiger Software immer häufiger Kommunikation über Nichtstandardports beobachten. Nicht nur IRC-Verbindungen auf Port 51115, sondern auch Kommunikationsmuster wie HTTP auf Port 21 oder FTP auf Port 80 nehmen ständig zu. Somit werden neue Techniken zur Protokollbestimmung benötigt, die traditionelle Ansätze durch größere Flexibilität übertreffen. Um diese Flexibilität zu erreichen setzen wir eine hybride Protokollidentifizierung ein. Durch die Kombination aus herkömmlichen Ansatz und Payload-basierter Protokollerkennung erhalten wir eine zuverlässigere Methode, die auch bei Schadsoftware gute Ergebnisse liefert.

Als nächstes betrachten wir die Beantwortung eingehender Anfragen. Davon ausgehend, dass das Protokoll einer eingehenden Verbindung erkannt wurde, gibt es im Wesentlichen zwei Möglichkeiten: Zum einen könnten die entsprechenden Dienste, die wir unterstützen wollen, emuliert werden. Hierbei könnten wir von Vorteilen ähnlich denen bei sogenannten *low-interactive* Honeypots [PH07] profitieren: Sicherheit und Performanz. Zum anderen wäre das Nutzen von bereits vorhandenen Diensten möglich. Auf diese könnten die eingehenden Verbindungen, abhängig von dem jeweils bestimmten Protokoll umgeleitet werden. Um unsere Aufmerksamkeit zunächst völlig der Realisierung auf Netzwerkebene widmen zu können, entscheiden wir uns für die zuletzt genannte Variante.

## 2.3 Zuordnung von Verbindungsanfrage und Dienst

Ausgestattet mit der Funktionalität der Protokollerkennung und den entsprechenden Diensten auf dem lokalen System, benötigen wir nur noch eine Lösung, um die eingehenden Verbindungen auf die jeweiligen Dienste zu schalten. Hierzu existieren wiederum zwei verschiedene Möglichkeiten. Einerseits können bei der Umlenkung der Datenpakete die Zielports unverändert beibehalten und ein der Protokollbestimmung entsprechender Dienst an den beobachteten Zielport, welchen wir den Kopfzeilen des Datenpaketes entnehmen können, gebunden werden. Anschließend wird der Dienst auf diesem Port angesprochen. Andererseits ist es auch möglich die unterstützten Dienste auf ihren Standardports laufen zu lassen, alle eingehenden Verbindungen zwecks Protokollidentifizierung auf eine Art Verteilerdienst (auf einem festgelegten Port) umzulenken und nach der Bestimmung des Protokolls die Verbindung auf den entsprechenden Service weiterzuleiten. Da der erste Ansatz nicht nur Schwierigkeiten in der Protokollidentifikation mit sich bringt, sondern auch zu Performanzproblemen führen kann (viele Instanzen des gleichen Dienstes), entscheiden wir uns für die letztere Variante und implementieren einen Verteilerdienst.

## 2.4 Die unterstützten Dienste

Selbstverständlich ist es nicht möglich alle im Internet zu findenden Dienste zu unterstützen. Darum begnügen wir uns zunächst mit der Bereitstellung von (passivem) FTP, HTTP, IRC und SMTP. Neben den bereits erwähnten Bestrebungen Transparenz unseres Systems auf Netzwerkprotokollebene zu erreichen, möchten wir auch Ideen vorstellen, wie die Emulation auch auf Protokollebene so realitätsnah wie möglich gestaltet werden kann. Hierbei möchten wir nicht bestehende Serverdienste anpassen, sondern die Emulation lediglich durch Veränderungen der Nutzlastdaten verbessern. Auf diese Art kann unser System mit beliebigen Serverdiensten für die unterschiedlichen Protokolle betrieben werden, und der zuständige Administrator kann sich weitestgehend uneingeschränkt für die von ihm favorisierten Dienstprogramme entscheiden. Somit braucht er nicht erst die Konfiguration bestimmter Serverdienste zu erlernen, wenn er mit Programmen gleicher Funktionalität bereits vertraut ist. Des Weiteren werden wir, abhängig von den eingehenden Anfragen, Änderungen an der Verzeichnisstruktur vornehmen. Diese spiegeln nicht nur die vom Client erwartete Serververzeichnisstruktur wieder, sondern geben dem Client bis zu einem gewissen Grad auch das Gefühl, wirklich mit dem von ihm beabsichtigten Server verbunden zu sein.

Ein berechtigter Einwand an dieser Stelle könnte sein, dass wir mit diesem Ansatz niemals einen menschlichen Benutzer täuschen können. Dies ist auch nicht unser Anspruch. Das gesamte System ist in erster Linie darauf ausgerichtet Schadsoftware zu täuschen, um möglichst viele der Funktionen, die in der jeweiligen Malware implementiert sind, auszulösen.

## 3 Technischer Hintergrund

Bevor wir uns der eigentlichen Implementierung zuwenden, möchten wir in diesem Abschnitt die verwendeten technischen Hilfsmittel vorstellen und eine abstrakte Sicht auf das System geben, um die Integrationsmöglichkeiten in ein bestehendes Netzwerk zu veranschaulichen.

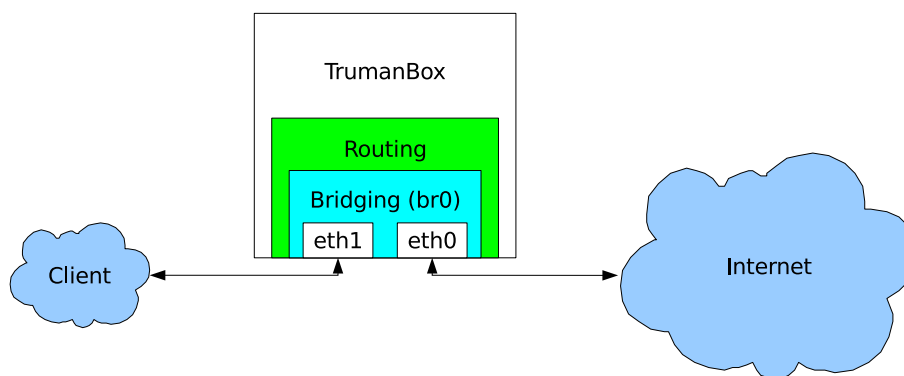


Abbildung 3: Konvention in der Bezeichnung

Als Basis nehmen wir einen Computer mit zwei Netzwerkkarten auf dem ein natives Linux System läuft. Dieses System stellt die unterstützten Anwendungsprotokolle auf ihren jeweiligen Standardports zur Verfügung. Während das eine Netzwerkkarte (eth1) mit dem Client (z.B. ein dynamisches Malwareanalyzesystem oder ein ganzes Netzwerk) verbunden wird stellen wir über das andere Interface (eth0) eine physikalische Verbindung zu dem zu schützen-

den Netzwerk (z.B. das Internet) her. Die beiden Netzwerkinterfaces werden einem logischen Bridgeinterface (`br0`) zugeordnet und von da an als dessen Bridgeports bezeichnet. Die beschriebene Konfiguration ist in Abbildung 3 dargestellt. Die vorgestellte Konfiguration erlaubt dem Client mit dem externen Netzwerk zu kommunizieren und garantiert gleichzeitig, dass sämtliche Datenpakete das Emulationssystem, die *TrumanBox* passieren.

### 3.1 Bridging

Im Hinblick auf unsere Anforderungen, nach denen wir ein möglichst transparentes und flexibles System haben möchten, scheint die Wahl der Integration in ein bestehendes Netzwerk als Bridge, ausgesprochen gut zu sein. Da eine Bridge standardmäßig auf OSI Level 2 operiert, ist die gewünschte Transparenz auf OSI Level 3 somit automatisch gegeben. Datenpakete werden zwischen den beiden Bridgeports weitergereicht und bei einer unbekanntenen Ziel MAC-Adresse über beide Interfaces gesendet. Durch Beobachtung des gesendeten Datenverkehrs lernt die Bridge, welche MAC-Adressen über welches Interface erreichbar sind. Somit wird unnötiges Weiterreichen vermieden, indem Pakete, die auf dem Port ankommen, der auch mit dem Netzwerkbereich verbunden ist, in welchem sich die Ziel MAC-Adresse befindet, verworfen werden. Auf diese Weise wird gleichzeitig auch die Datenlast in beiden durch die Bridge verbundenen Netzwerksegmenten deutlich reduziert. Eine abstrakte Sicht auf die beschriebene Konfiguration ist in Abbildung 4 gegeben. Soll wie in unserem Fall auch die Bridge selbst mit anderen Rechnern (z.B. im Internet) kommunizieren können, muss dem logischen Bridgeinterface eine IP-Adresse zugewiesen werden. Zusätzlich müssen entsprechende ARP Anfragen, die von der Clientseite kommen, geblockt werden, um die Transparenz der Bridge zu erhalten,

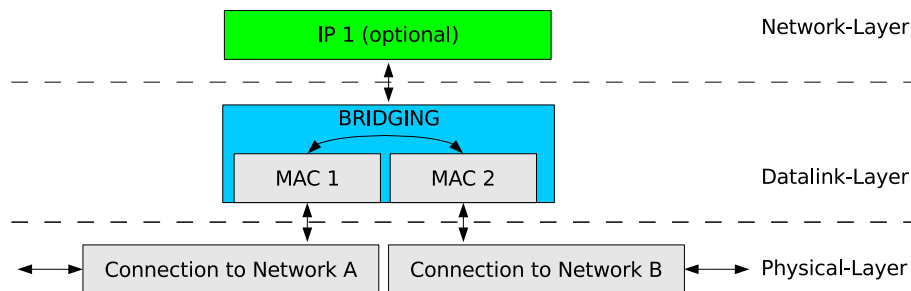


Abbildung 4: Konfiguration der Bridge

### 3.2 Datenflusskontrolle

Des Weiteren ist es für unseren Ansatz entscheidend zu wissen, wie Datenpakete die Bridge passieren. Eine schematische Darstellung zum sequentiellen Ablauf wird in Abbildung 5, getrennt nach Hooks, die über *iptables* [Rus07] beziehungsweise *eatables* [Sch07] angesprochen werden können, gegeben. Für jeden Hook können hierbei Regeln aufgestellt werden, die durch eine Muster/Aktion-Kombination auf entsprechende Pakete angewendet werden.

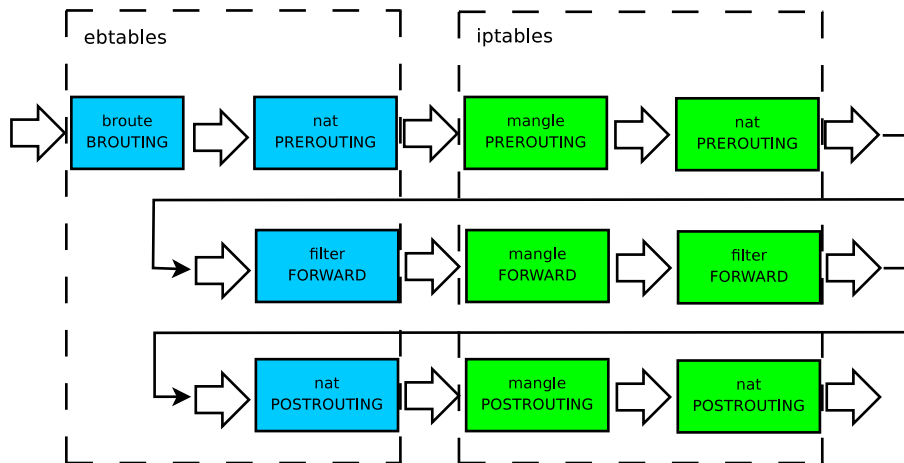


Abbildung 5: Datenfluss durch eine Linux Bridge

Auf diese Weise werden entsprechende Daten auf unser lokales System (die Bridge) umgelenkt. Da beim Umlenken der Daten die ursprünglichen Header-Informationen verloren gehen, benutzen wir zwei hintereinanderliegende Hooks. In dem ersten reichen wir die Datenpakete in den Userspace, um die unveränderten Header-Informationen zu extrahieren. Dies geschieht mittels des von uns implementierten *Intercept*-Prozesses, welcher die C Bibliothek *libipq* [Mor07] verwendet. Anschließend wird das jeweilige Paket wieder in den Kernspace zurück gereicht, wo es seinen Weg durch den Kernel fortführt. In dem darauffolgenden Hook wird das Paket auf das lokale System, auf einen bestimmten Port, umgelenkt. Auf diesem Port lauscht unser *Dispatch*-Prozess, welcher für die weitere Verarbeitung zuständig ist. Die Interaktion der beiden von uns implementierten Prozesse mit den von Linux bereitgestellten Schnittstellen ist in Abbildung 6 veranschaulicht. Vergleicht man die Grafik mit Abbildung 5, sieht man, dass wir die Netzwerkpakete an zwei aufeinanderliegenden Hooks manipulieren, die noch vor dem eigentlichen Routing durchlaufen werden. Dies macht Sinn, wenn man bedenkt, dass wir den Datenstrom auf unser System umlenken wollen und somit schon vor der Routing-Entscheidung die entsprechenden Header-Informationen anpassen müssen.

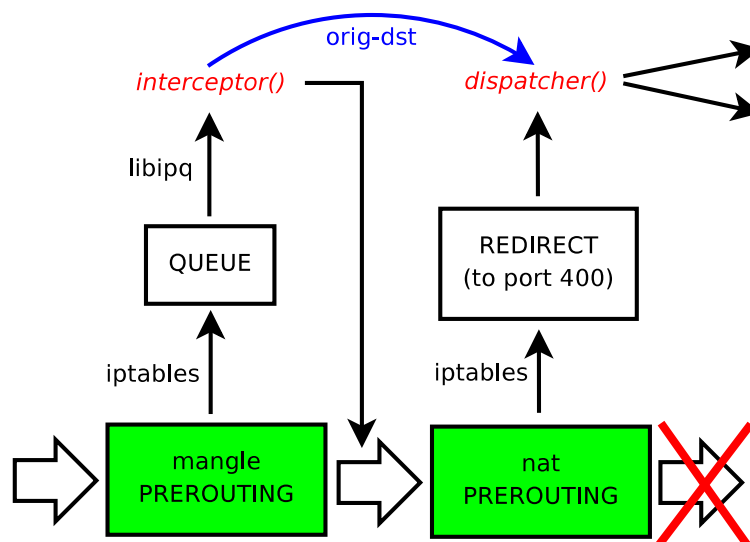


Abbildung 6: Datenfluss Intercept und Dispatch Prozess

## 4 Implementierung

Nachdem wir im vorangegangenen Abschnitt auf die Basiskonfiguration unseres Systems als Bridge eingegangen sind, können wir unser Augenmerk auf den eigentlichen Implementierungsteil richten. Wie schon erwähnt haben wir im Wesentlichen zwei verschiedene Prozesse, welche im Userspace die erhaltenen Daten behandeln. Während der Intercept-Prozess lediglich Header-Informationen (IP-Adressen und Port-Nummern) über Quelle und Ziel der Datenpakete extrahiert werden alle übrigen Aufgaben vom Dispatch-Prozess, dessen Funktionalität sich in unterschiedliche Teilaufgaben gliedern lässt, übernommen. In Abbildung 7 sind sowohl der interne Informationsfluss (durchgezogene Pfeile), als auch die möglichen Wege der Datenpakete (gestrichelte Pfeile) vereinfacht dargestellt.

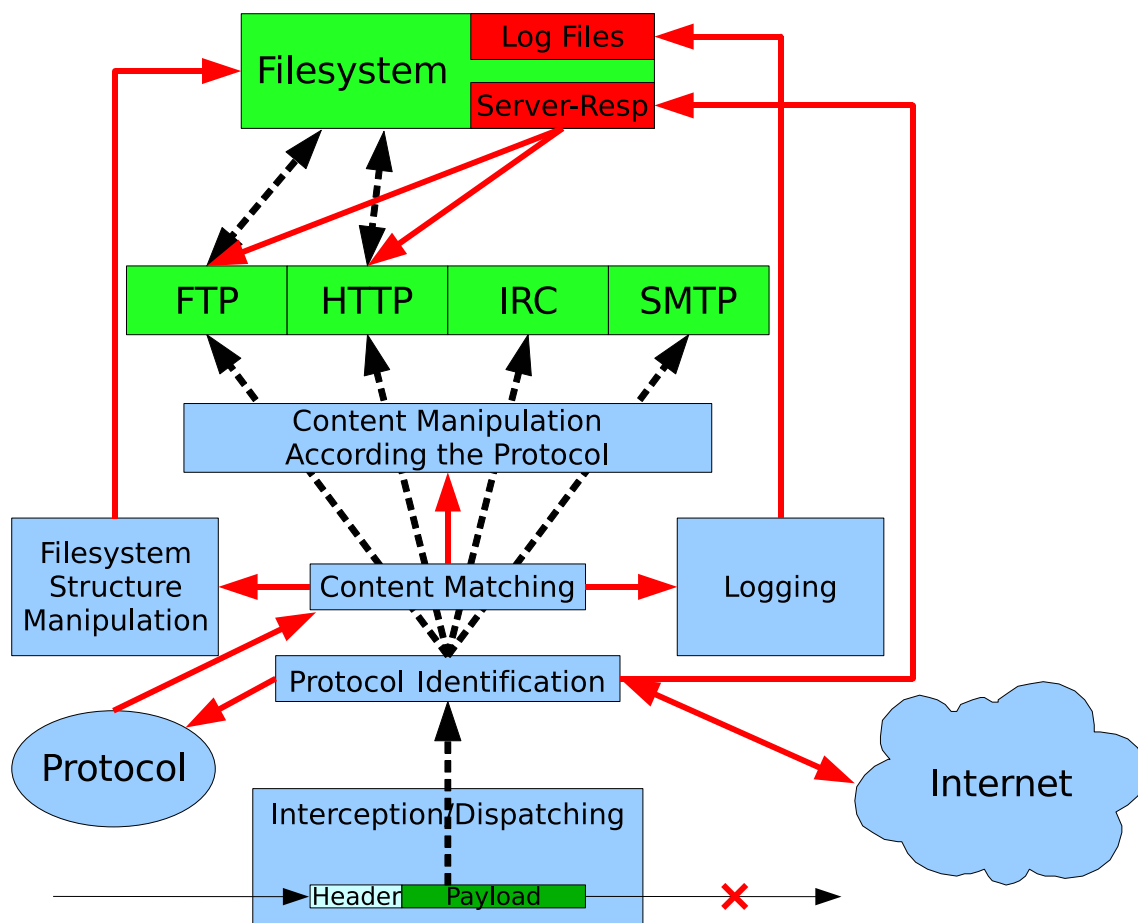


Abbildung 7: Vereinfachte Darstellung der Programmstruktur

Neu eingehende Verbindungsversuche werden zunächst auf den konfigurierbaren Port, an dem der Dispatch-Prozess lauscht, gelenkt. Dieser nimmt die Verbindung an und veranlasst den Aufruf der weiter benötigten Prozeduren. Die Funktionalität der einzelnen Prozeduren wird im folgenden genauer beschrieben. Um verschiedenen Restriktionen, die in der Regel von der Umgebung in der das System verwendet werden soll abhängen, gerecht zu werden, unterstützt die TrumanBox verschiedene Betriebsmodi: Simulation, Halb-Proxy, Proxy und Transparent. Mittels dieser können Art und Menge der ausgehenden Datenverbindungen reguliert werden. Während im Simulations Modus alle Anfragen ohne weitere Informationen aus dem Internet

beantwortet werden können, werden im Halb-Proxy Modus bestimmte Informationen aus dem Internet abgefragt, um die Emulation zu verbessern. Proxy und Transparent Modus bieten keine neue Funktionalität sondern ermöglichen lediglich einen Betrieb, vergleichbar mit andern Proxy Lösungen, beziehungsweise Netzwerk-Sniffen.

## 4.1 Protokollidentifikation

Gemäß unserer ursprünglichen Motivation, die dynamische offline Analyse von Malware zu verbessern, müssen wir uns auch mit der Problematik beschäftigen, dass Schadsoftware häufig Nichtstandard-Ports zur Kommunikation mittels Standardprotokollen verwendet. Natürlich sind wir uns bewusst, dass auch modifizierte Protokolle und sogar verschlüsselte Varianten bereits zu beobachten sind. Diese wollen wir jedoch im Rahmen dieser Arbeit unbeachtet lassen. Unter Berücksichtigung der von uns überwiegend beobachteten Malware, ist diese Einschränkung tragbar, da zumindest bislang Verschlüsselung eher selten zum Einsatz kommt. Der von uns verwendete Ansatz zur Protokollidentifizierung arbeitet sowohl Payload- als auch Port-orientiert, wobei wir die Payload-basierte Protokollbestimmung priorisieren. Unmittelbar nachdem der Dispatch-Prozess eine neue Verbindung angenommen hat, wird die Payload nach bekannten vordefinierten Mustern gemäß den unterstützten Protokollen durchsucht. Wenn eines dieser Muster gefunden wird ist das Protokoll identifiziert. Andernfalls wird in einem zweiten Anlauf versucht das Protokoll über die vordefinierten Portnummern zu bestimmen. Auch wenn dieser zweite Versuch nicht ganz so vielversprechend erscheint, bietet die Beobachtung, dass es auch unter Malware typische Nichtstandard-Ports für bestimmte Protokolle gibt, eine unserer Meinung nach ausreichende Rechtfertigung. Im Falle von Protokollen, bei denen der Server zuerst Daten sendet (z.B. FTP und SMTP), hängt die Payload-basierende Protokollidentifizierung von dem Modus ab, in dem die TrumanBox betrieben wird. Falls tolerierbar, versucht sich das System, wenn es keine Daten von Clientseite empfängt, zu dem eigentlichen Server zu verbinden, um an Hand der Antwort das verwendete Protokoll zu identifizieren. Antworten, die während der Analyse aus dem Internet abgefragt werden, werden lokal gespeichert, um, im Falle einer wiederholten Anfrage an den gleichen Server, die entsprechende Antwort von der Festplatte zu lesen und somit ausgehende Datenverbindungen zu reduzieren. Falls das verwendete Protokoll einer Verbindung nicht bestimmt werden kann, werden die zugehörigen Payloads aufgezeichnet und ebenfalls abgespeichert. Diese Informationen helfen bei der nachträglichen Analyse und vereinfachen die Erweiterung um neue Protokolle.

Auch wenn die Protokollidentifizierung in unseren Testläufen gute Ergebnisse geliefert hat, ist sie keinesfalls ausgereift. Hier würde es vermutlich Sinn machen über die Verwendung bereits vorhandener Verfahren [DFM06, CKW07] nachzudenken.

## 4.2 Content Matching

Wenn das verwendete Protokoll einer Datenverbindung erfolgreich bestimmt wurde, wird diese Information als Status weitergereicht und davon abhängig eine zweite Verbindung zu dem entsprechenden (lokalen) Dienst aufgebaut. Abhängig vom Modus, in dem die TrumanBox betrieben wird, kann die zweite Verbindung auch zu dem original Dienst im Internet aufgebaut

werden. In diesem Fall erübrigen sich die meisten der im weiteren Verlauf ausgeführten Emulationsbemühungen und wir können als *Man-in-the-middle* die Datenverbindung, beziehungsweise die für uns relevanten Informationen, aufzeichnen und auch manipulieren. Zu diesem Zweck müssen entsprechende Funktionen individuell bereitgestellt werden.

Allgemein beobachtet die *Content Matching* Funktion die gesendeten Daten und ruft bei Bedarf abhängig von entsprechen Mustern in der Payload weitere Funktionen auf.

### 4.3 Logging

Das Logging wird über die Content Matching Funktion gesteuert. Wahlweise können entweder alle oder nur bestimmte Daten, die einem vordefinierten Muster entsprechen, im Textformat aufgezeichnet werden. Die mitgeschnittenen Kommunikationsdaten werden in Dateien, die nach IP-Adresse und Zielpport der ursprünglichen Verbindungsanfrage benannt werden, gespeichert. Da unter Umständen mehrere Verbindungsversuche auf den gleichen Serverdienst beobachtet werden können, wird den mitprotokollierten Daten jeweils ein Zeitstempel vorangestellt. Somit ist zum Beispiel ein nachträgliches Korrelieren von IP Adresse, Port, verwendete Zugangsdaten, Zugriffszeit und dem Protokoll, welches in der Verbindung verwendet wurde, möglich.

### 4.4 Manipulation der Dateisystemstruktur

Im Falle von HTTP- und FTP-Verbindungen wird bei entsprechenden Anfragen eine Manipulation der Dateisystemstruktur angestoßen. Hierbei werden neue Verzeichnisse gemäß der Client-Anfrage angelegt. Ausgehend davon, dass der Client nur Verzeichnisse oder Dateipfade anfragt, die er auf dem eigentlichen Zielsystem erwartet, erhalten wir somit zumindest teilweise eine Nachbildung des Verzeichnisbaumes des jeweiligen Servers. Im Gegensatz zum einfachen Logging ermöglicht uns dieser Ansatz das automatisierte Zuspielden von Inhalten, die wir kontrollieren. Dabei kann es sich sowohl um HTML-Daten, als auch um ausführbare Dateien oder Ähnliches handeln.

### 4.5 Manipulation der übermittelten Nutzdaten

Ein ganz entscheidender Punkt in der Emulation von Internetdiensten ist die Payload-Manipulation. Insbesondere in unserem Ansatz, bei dem wir eingehende Verbindungen auf beliebige, lokal laufende Standarddienste (für die jeweiligen Protokolle) umlenken. Wie bereits erwähnt werden abhängig von dem Modus, in dem wir die TrumanBox betreiben, unter Umständen weitere Informationen wie zum Beispiel FTP oder SMTP Begrüßungs-Banner heruntergeladen und gespeichert. Bei einem FTP-Dienst wird unter anderem getestet, ob der eigentliche Serverdienst einen anonymen Login zulässt. Die gespeicherten Banner werden dem Client vorgespielt, der ausschließlich mit dem lokal laufenden Dienst kommuniziert. Bei Bedarf werden auch Antworten im weiteren Verlauf der Kommunikation ersetzt. Beispielsweise wird die *Passive Mode* Direktive, welche dem FTP-Client mitteilt, auf welchem Port eine Datenverbindung aufgebaut werden soll, entsprechend manipuliert. So merkt der Client nicht, dass er gar nicht mit dem eigentlichen Server, den er zu kontaktieren glaubte, kommuniziert.

Aber nicht nur die vom Server gegebenen Antworten werden bei Bedarf angepasst. Auch Client-Anfragen, wie etwa Login-Daten, werden manipuliert, um abhängig von den vorliegenden Informationen über das eigentliche Zielsystem einen Zugang zu gewähren oder nicht. Bislang sind hier nur exemplarisch einige verschiedene FTP-Login Möglichkeiten implementiert. Im Wesentlichen wird mit jeder Benutzername/Passwort-Kombination ein Zugang gewährt, da wir auch hier davon ausgehen, dass ein Client in der Regel die Zugangsdaten entweder kennt oder versucht sie zu erraten. Da wir in beiden Fällen an dem weiteren Verhalten, welches dem erfolgreichen Login folgt, interessiert sind, macht es Sinn den Zugang immer zu gewähren. Dennoch sind wir uns bewusst, dass auf diese Weise ein Authentizitätstest, in Form eines Login-Versuches mit absichtlich falschen Benutzerdaten, unsere Emulation aufdecken kann. Für die Zukunft sind für diese Problematik verfeinerte Techniken geplant. Für den Moment begnügen wir uns damit, dass die meisten böartigen Programme derartige Bemühungen noch nicht anstellen.

## 4.6 Unterstützte Protokolle

Auch wenn wir in den vorangegangenen Abschnitten vereinzelt die unterstützten Anwendungsprotokolle bereits angesprochen haben, möchten wir an dieser Stelle noch einmal explizit auf die verschiedenen Protokolle, auch auf Netzwerkebene, eingehen. Nachdem in der ersten Version der TrumanBox lediglich TCP/IP unterstützt wurde, ist die Unterstützung der verschiedenen Anwendungsprotokolle (FTP, HTTP, IRC und SMTP) bislang auf dieses Netzwerkprotokoll eingeschränkt. Ob sich eine Erweiterung dahingehend, dass diese Anwendungsprotokolle auch über UDP Verbindungen unterstützt werden, lohnt, wird stark von den zukünftigen Auftrittswahrscheinlichkeiten für diese Kombinationen abhängen. Bislang scheint dafür noch keine ausreichende Motivation gegeben. Dennoch haben wir während der ersten Testläufe beobachten können, dass eine Unterstützung von UDP und ICMP gerade in der Malwareanalyse durchaus Sinn macht. Auch wenn UDP in Kombination mit DNS bereits in der ersten Version unterstützt wurde - hier haben wir DNS-Verbindungen einfach auf einen lokalen DNS-Dienst mit einer angepassten Konfiguration umgelenkt - war diese Option zunächst eher für Ausnahmesituationen gedacht, in denen nicht einmal DNS-Anfragen in das Internet gelangen sollten. Im Verlauf haben wir jedoch festgestellt, dass auch eine vereinfachte DNS-Namensauflösung sinnvoll sein kann. Wenn beispielsweise ein bestimmter Domainname nicht mehr aufgelöst werden kann, können wir eine von uns festgelegte IP-Adresse zurückliefern und so auch ältere Malware detailliert analysieren.

Doch nicht nur für das DNS-Protokoll ist eine Unterstützung von UDP angebracht. Während verschiedener Analysen von Malware konnten wir immer wieder Verbindungetests mittels UDP oder ICMP beobachten. Wenn auf ein entsprechendes Paket keine Antwort kam, hat in vielen Fällen die betrachtete Schadsoftware jegliche Aktivitäten eingestellt. Dieses Verhalten lässt darauf schließen, dass bestimmte Funktionen von Malware nur dann aufgerufen werden, wenn die Programme über eine Internetverbindung verfügen oder es für sie so scheint, als hätten sie diese. Darum haben wir in der aktuellen Version eine "UDP-Echo" Funktionalität implementiert. Während DNS-Anfragen je nach Konfiguration, entweder regulär von einem Nameserver oder aber von der TrumanBox beantwortet werden, können jegliche andere UDP-Verbindungen mit einem Zurücksenden der übermittelten Payload beantwortet werden. Im Falle von ICMP werden die Pakete lediglich auf die TrumanBox umgelenkt und somit

ebenfalls transparent beantwortet. In beiden Fällen ist die Möglichkeit einer einstellbaren oder zufälligen Verzögerung für die Zukunft geplant.

## 5 Ergebnisse

Auch wenn die TrumanBox gemäß der vorangegangenen Diskussion bereits in vollem Funktionsumfang implementiert ist, handelt es sich bislang nach wie vor um eine frühe Beta Version. Die von uns durchgeführten Tests in Kombination mit dem dynamischen Malware Analyse-system CWSandbox [WHF07] geschahen mit der ersten Version der TrumanBox, die noch keine UDP- und ICMP-Unterstützung besaß. Dennoch waren die erhaltenen Ergebnisse überraschend gut. Teilweise erhielten wir in Kombination mit der TrumanBox bessere Analyse-ergebnisse als mit einer uneingeschränkten Internetverbindung. Diese Beobachtung ist darauf zurückzuführen, dass mit Hilfe einer Emulation, wie sie die TrumanBox bietet, Schadsoftware auch dann noch detailliert und dynamisch untersucht werden kann, wenn angeforderte Dienste bereits vom Netz genommen wurden. Somit erhalten wir Ergebnisse wie in Abbildung 8 skizziert (vgl. Abbildung 1).

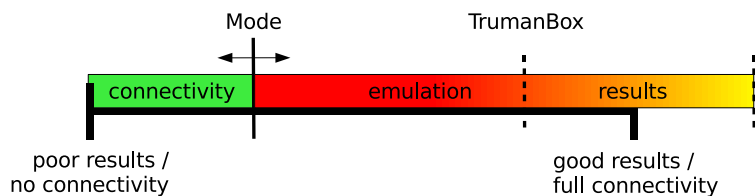


Abbildung 8: Qualität der Ergebnisse in der dynamischen Malwareanalyse

Während laufender Analysen der als *Storm Worm* [GSN<sup>+</sup>07] bekannt gewordenen Malware, war die TrumanBox in der Lage durchschnittlich 1,5 Emails/Sekunde, mit maximalen Raten von bis zu 8 Emails/Sekunde zu verarbeiten. Da es mittlerweile einige Beta-Tester gibt, hoffen wir bald umfangreichere Angaben zur Performanz machen zu können.

Um das Interesse an der TrumanBox zu erhöhen, gibt es mittlerweile auch die Möglichkeit, die TrumanBox komplett offline auf einem Computer mit nur einem Netzwerkinterface und Linux als Betriebssystem zu betreiben. Auf diese Weise kann die TrumanBox zum Beispiel auch für Präsentationen eingesetzt werden oder dem Testen bei der Entwicklung von Client Applikationen dienen. Da die TrumanBox als alleinstehende Applikation und somit unabhängig von anderen Komponenten entwickelt wird, bietet sie ein hinreichendes Logging, um bei der Datenverkehrsanalyse auch ohne ein Loggingsystem auf der Clientseite einen tiefgehenden Einblick zu geben.

## Literatur

- [CKW07] Weidong Cui, Jayanthkumar Kannan, and Helen J. Wang. Discoverer: Automatic protocol reverse engineering from network traces. In *Proceedings of 16th USENIX Security Symposium*, Boston, August 2007.

- [DFM06] Holger Dreger, Anja Feldmann, and Michael Mai. Dynamic application-layer protocol analysis for network intrusion detection. In *Proceedings of 15th USENIX Security Symposium*, Vancouver, August 2006.
- [GSN<sup>+</sup>07] Julian B. Grizzard, Vikram Sharma, Chris Nunnery, Brent ByungHoon Kang, and David Dagon. Peer-to-peer botnets: Overview and case study. In *Proceedings of 1st USENIX Workshop on Hot Topics in Understanding Botnets*, 2007.
- [Mor07] James Morris. libipq: iptables userspace packet queuing library, Accessed: July 2007. Internet: <https://svn.netfilter.org/netfilter/trunk/iptables/libipq/>.
- [PH07] Niels Provos and Thorsten Holz. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Addison-Wesley, 1st edition, 2007.
- [Rus07] Rusty Russell. Iptables, Accessed: July 2007. Internet: <http://http://www.netfilter.org/>.
- [Sch07] Bart De Schuymer. Ebttables, Accessed: July 2007. Internet: <http://ebtables.sourceforge.net>.
- [WHF07] Carsten Willems, Thorsten Holz, and Felix Freiling. Toward automated dynamic malware analysis using cwsandbox. *Security and Privacy Magazine, IEEE*, 5(2):32–39, March-April 2007.