Netzwerksicherheit mit probabilistischen Angriffsgraphen

Andreas Schmitz[1] · Heinrich Pettenpohl[2]

Fraunhofer ISST[1]
Andreas.Schmitz@isst.fraunhofer.de

TU Dortmund[2]
Heinrich.Pettenpohl@cs.tu-dortmund.de

Zusammenfassung

Durch die Entwicklung von intelligenten Computerwürmern wie Stuxnet bewegen sich Angreifer durch ein Netzwerk und erreichen ihr Ziel in mehreren Schritten. Dieser Artikel beschreibt einen ganzheitlichen Ansatz basierend auf der Topologischen Vulnerabilitiy Analysis (TVA), um sich gegen mehrschrittige Angriffe absichern zu können. Zunächst wird ein Netzwerk auf Schwachstellen aus einer Datenbank analysiert und anschließend alle möglichen Angriffspfade durch das Netzwerk berechnet. Zusätzlich werden unbekannte Schwachstellen mit Hilfe von Analysen einbezogen. Es gibt zwei Quellen für diese probabilistischen Schwachstellen: Generische Exploits basierend auf den Datenbanken der CWE und CAPEC, sowie mit stochastischen Modellen vorhergesagte Schwachstellen. Der daraus entwickelte Angriffsgraph wird anschließend analysiert, um das Netzwerk vor Angriffen zu sichern. Dieses gesamte Vorgehen wurde in einer Software umgesetzt.

1 Einleitung

Viele Angriffe aus dem Internet erreichen ihr Ziel durch Ausnutzung einer Schwachstelle, häufig im Browser oder ähnlichem. Durch die verstärkte Vernetzung in Unternehmen sind immer mehr Ziele angreifbar, das Risiko steigt stetig und fordert von Unternehmen immer stärkere Perimeter. So sind viele Server vom Internet abgeschottet und nicht mehr direkt zugänglich. Doch mit der Entwicklung von intelligenten Computerwürmern bewegen sich Angreifer in mehreren Schritten durch ein Netzwerk und erreichen damit tief versteckte und abgeschottete Systeme.

Einer der bekanntesten und spektakulärsten Angriffe wurde durch den Computerwurm Stuxnet im Jahr 2010 ausgeführt. Dieser manipulierte in einem Atomkraftwerk im Iran die Geschwindigkeit von Motoren, welche an eine Siemens Simatic S7 Steuerung angeschlossen waren. Um bis zu diesem Ziel zu gelangen verwendet Stuxnet zahlreiche Schritte. Viele technische Methoden werden für die Verbreitung des Wurms im Netzwerk benutzt, die bisher jedes Sicherheitssystem überfordern: Zero-Day-Exploits, ein Windows Rootkit, das erste PLC-Rootkit, Antivirus-Ausweichtechniken, komplexe Prozessinjektion, Hooking-Code, Netzwerk-Infektionsroutinen, Peer-to-Peer Updates und er besitzt ein Command und Control-Interface [FaMC11].

In diesem Artikel wird ein Verfahren basierend auf der Topologischen Vulnerabilitiy Analysis (TVA) [JaNo10] vorgestellt, um sich gegen mehrschrittige Angriffe wehren zu können. Das

P. Schartner · K. Lemke-Rust · M. Ullmann (Hrsg.) · D•A•CH Security 2015 · syssec (2015) 311-322.

gesamte Verfahren wurde in einer Software umgesetzt.

2 Verwandte Arbeiten

Es gibt zahlreiche Softwareprogramme und Arbeiten, die sich mit der Absicherung von Netzwerken beschäftigen. Zu den bedeutesten Programmen gehört **Nessus** [Ten] vom Hersteller tenable. Das Programm entdeckt Netzwerkteilnehmer und scannt diese auf Sicherheitslücken. Nach der Kommerzialisierung von Nessus ist der Ableger **Open Vulnerability Assessment System** (OpenVAS) [Ope15] entstanden. Ähnlich wie Nessus ist OpenVAS eine Sammlung von Diensten und Werkzeugen für Schwachstellenanalyse und -management und integriert Schwachstellendatenbanken. Die Datenbanken enthalten sogenannte "Network Vulnerability Tests", welche auf die spezifizierten Rechner angewendet werden. Ein weiteres Tool, das von vielen Unternehmen eingesetzt wird ist **Secunia CSI** [Sec], kurz für Corporate Software Inspector. Es erfasst die Software auf den Geräten des Anwenders, prüft auf nicht installierte Patches und kann diese auf Wunsch nachinstallieren. **BigBrother** [MaCr] ist eine Software, die auf dem Client-Server-Modell basiert und über eine Webseite die laufenden Netzwerk-Services darstellt. Für diese Software gibt es sowohl freie als auch kommerzielle Lizenzen. **CoreImpact** [CoS] ist eine kommerzielle Softwarelösung von Core Security. Sie kann das Netzwerk scannen, visualisieren und analysiert verschiedene Risiken unter Verwendung verschiedener Metriken. **NexPose** [Rapi] ist eine Softwarelösung von Rapid7 und bietet Lizenzen für Firmenkunden und Endanwender. Nexpose analysiert das Netzwerk auf Schwachstellen und lässt sich zusätzlich mit MetaSploit kombinieren. **Qualys** [QuS] ist eine cloudbasierte Software, sogenannte „Software as a Service", welche das Netzwerk auf Schwachstellen untersucht, und dadurch Schutz vor Bedrohungen bietet. **Retina** [bey] ist ein weiterer Netzwerk- und Schwachstellenscanner, kann einzeln oder verteilt eingesetzt werden und integriert Penetration Testing Tools.

Der Begriff „Attack Graph" - Angriffsgraph wurde 1998 von Phillips und Swiler geprägt [PhSw98]. Es gibt viele unterschiedliche Graphen, um Schwachstellen in Verbindung zu setzen und den resultierenden Graphen zu analysieren. Eine Übersicht, die mehr als 30 unterschiedliche Typen behandelt, haben Kordy *et al.* in [Koal14] zusammengestellt. Die bekanntesten sind NetSPA, MulVAL und TVA. Die Generierung der Angriffsgraphen basiert auf dem Algorithmus von Ammann *et al.* [Amal02]. NetSPA wurde von Artz *et al.* [Artz02] entwickelt. NetSPA wird mit Hilfe von GARNET visualisiert [WiLI08]. Später wurde GARNET durch das neue Programm NAVIGATOR ersetzt [Chal10]. MulVAL ist ein Open-Soruce Programm und benutzt für die Generierung des Angriffsgraphen ein Prolog-System namens XSB, welches Datalog-Interaktionsregeln analysieren kann [OuAp05, OuGA05, OuBM06].

Als Grundlage wird in diesem Artikel die TVA verwendet [JaNo10]. Wang *et al.* definieren in [Waal10] Zero-Day-Exploits und eine K-Zero-Day-Metrik, mit der die Sicherheit von Netzwerken bzgl. Zero-Day-Angriffen bestimmt werden kann. Der Angriffsgraph kann mit einer Rückwärtssuche optimiert werden, so dass keine überflüssigen Pfade mehr enthalten sind [WaAJ14]. Wang *et al.* haben in [WaAJ14] zwei Algorithmen beschrieben, mit denen wichtige Punkte gefunden werden, um das Netzwerk gegen Angreifer zu härten. Die Algorithmen unterscheiden sich in Laufrichtung, Laufzeit und Kostenberechnung. Eine erste grundlegende probabilistische Analyse des Angriffsgraphen wird ebenfalls von Wang *et al.* beschrieben [Waal08]. Diese berechnet die Wahrscheinlichkeit, mit der ein Zielsystem erreicht wird. Die Analyse wurde anschließend von Homer *et al.* für MulVAL verfeinert [Hoal13].

3 Konzept

Das hier vorgeschlagene Verfahren zur Netzwerkanalyse besteht aus mehreren Schritten, auf die im späteren Verlauf genauer eingegangen wird. Zunächst wird das Netzwerk inventarisiert. Die

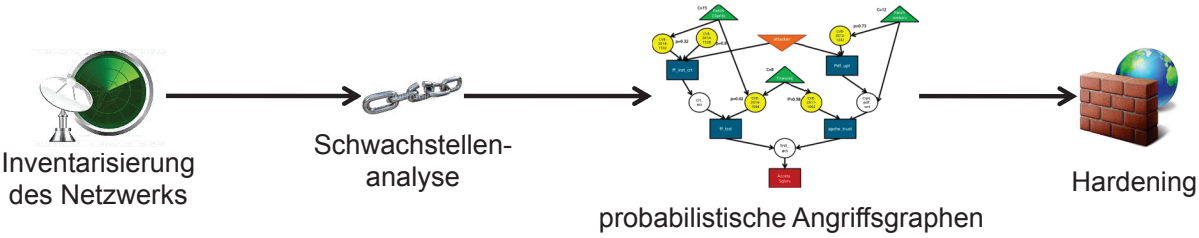

Abb. 1: Ablauf der Netzwerkanalyse

ermittelten Daten werden verwendet, um Schwachstellen und Exploits zu finden. Anschließend werden mögliche mehrschrittigen Angriffe mit Hilfe von probabilistischen Angriffsgraphen visualisiert. Der Angriffsgraph wird abschließend analysiert, um wichtige Systeme effektiv zu schützen.

3.1 Inventarisierung des Netzwerks

Bei der Inventarisierung des Netzwerks wird aktiv die Netzwerktopologie untersucht. Mit Hilfe des Netzwerkscanners NMap [Lyon, Lyon09, Wolf02] werden alle Netzwerkteilnehmer ermittelt und mit einem OS-Fingerprint in Windows- und Linux-Betriebssystemen unterteilt. Anschließend wird bei Computern mit Windows-Betriebssystem das Programm PSInfo verwendet, um die Namen, Versionsnummer und den Speicherort der installierten Software zu erhalten, sowie die installierten Patches auszulesen. PSInfo kann Remote auf die Computer zugreifen, benötigt aber Zugangsdaten zu den Windows-Systemen. Bei Domänensystemen wird ein Domänenadmin benötigt. Bei Linux-Betriebssystemen wird per SSH auf die Systeme zugegriffen und über den Paketmanager die Software mit Versionsnummer usw. ausgelesen. Der Netzwerkplan kann exportiert und dadurch mit MS Visio visuell dargestellt werden.

3.2 Identifizieren von Schwachstellen und Exploits

Nachdem die installierte Software von den einzelnen Netzwerkteilnehmern ermittelt wurde, kann diese mit Informationen über Schwachstellen angereichert werden.

NVD Die National Vulnerability Database - NVD enthält eine Vielzahl von bekannten Schwachstellen. Jede Schwachstelle besitzt eine eindeutige CVE-ID, sowie eine Beschreibung. Die Beschreibungen sind sehr ähnlich aufgebaut und können deswegen mit Hilfe von regulären Ausdrücken analysiert werden [Frie06]. Das Ergebnis der Analyse enthält folgende Punkte: Softwarename, Versionsnummer, Vorbedingungen, Nachbedingungen. Diese Daten werden in eine MySQL-Datenbank gespeichert, so dass die Analyse im späteren Verlauf nur auf neue NVD-Einträge angewandt werden muss. Durch den ermittelten Softwarenamen und der Versionsnummer kann eine Schwachstelle der installierten Software zugewiesen werden. Mit Hilfe der Vor- und Nachbedingungen können Exploits definiert werden. Ein Exploit ist eine Schadsoftware, durch welche ein Angreifer Privilegien auf einen Computer erlangt. Diese Informationen werden der Netzwerktopologie hinzugefügt.

Generische Exploits Neben den bekannten und veröffentlichten Exploits gibt es auch unbekannte Exploits von denen eine Gefahr ausgeht. Wir betrachten diese Gefahr auf verschiedenen Wegen. Einer davon ist die Verwendung von generischen Exploits. Es gibt Schwachstellen die in Software mit bestimmten Eigenschaften häufiger auftreten. Zum Beispiel sind bei C/C++ Programmen häufig Buffer-Overflows zu beobachten, während bei Webanwendung SQL-Injection ein Problem darstellt. Mit CAPEC und CWE steht eine Datenbank zur Verfügung, aus der solche generischen Exploits semi-automatisch abgeleitet werden können. In diesem Ansatz hat jeder Exploit zu einer Vor- und Nachbedingung auch eine Existenzwahrscheinlichkeit. Bei bekannten Expoits aus der NVD ist diese Wahrscheinlichkeit 1. Im Fall der generischen Exploits wird diese aus Softwareeigenschaften abgeleitet, die zuvor in einer Datenbank für verschiedene Softwareprogramme mit einer Wahrscheinlichkeit festgelegt wurden. Softwareeigenschaften können voneinander abhängen, so dass diese als bedingte Wahrscheinlichkeiten in die Berechnung eingehen. Daraus ergibt sich ein bayesisches Netz zur Berechnung. Ein Beispiel: Für die Webanwendung ERP ist die Eigenschaft „verwendet SQL-Befehle“ „sehr wahrscheinlich“ und die Eigenschaft „Filtert Eingabetexte nicht korrekt“„wahrscheinlich“. Die zweite Eigenschaft hängt von der ersten ab.

Predicted Exploits Eine weitere Möglichkeit unbekannte Exploits mit einer zugehörigen Existenzwahrscheinlichkeit aufzudecken ist es, Exploits aus der bekannten Historie vorherzusagen. Die in der NVD enthaltenen Einträge werden genutzt, um für jede Software ein Vulnerability Discovery Models (VDM) [NgMa12, Alal05, Resc05] zu trainieren. Diese basieren im Wesentlichen auf dem Einbetten sigmoider Funktionen. So kann für eine Software vorhergesagt werden, wie sich die Entdeckung der Schwachstellen durch die Internetgemeinde entwickelt. Für einen Angreifer der mehr Erfahrung hat oder großen Aufwand betreibt, könnte die Wahrscheinlichkeit zu gering eingeschätzt werden. Dennoch stellt dies ein Maß für mögliche Schwachstellen dar.

Konfigurationsfehler Eine andere Quelle von Schwachstellen, durch die Angreifer oder Schadsoftware eindringen kann, sind Konfigurationsfehler. Im SCAP-Standard werden Konfigurationsstellen für viele wichtige Programme definiert. Zudem wird mit dem USGCB eine Standardbelegung vorgeschlagen, die für Behörden in den Vereinigten Staaten verbindlich sind. Allerdings müssen Vor- und Nachbedingung manuell in die Datenbank eingetragen werden. Es gibt eine rudimentäre Implementierung eines SCAP-Interpreters zum Analysieren eines Computers. Dieser wird verwendet, muss allerdings stark erweitert werden.

Zero-Day-Exploits Trotz der vielen zusätzlichen Informationen kann es passieren, dass vollständig neuartige Schwachstellen oder Exploits von Angreifern verwendet werden. Um auch diesen Punkt nachvollziehen zu können, werden Zero-Day-Exploits eingesetzt. Dabei wird nach Wang *et al.* [WJSC+13] zwischen zwei Arten unterschieden: „Remote“ und „Privilege-Escalation“. Für einen Remote-Zero-Day-Exploit wird eine Software, eine Netzwerkverbindung zwischen Angreifer und Opfer, sowie die Kontrolle über den angreifenden Computer als Vorbedingung definiert. Der Angreifer erhält als Nachbedingung die Privilegien der Software. Die Privilege-Escalation ist ein lokaler Zero-Day-Exploit. Dieser braucht als Vorbedingung eine Software und generiert als Nachbedingung jegliche Privilegien auf dem Computer.

3.3 Generierung des Angriffsgraphen

Wenn die Exploits und die Topologie des Netzwerkes gegeben sind, kann daraus der Angriffsgraph generiert werden. Als Grundlage wird der Ansatz von Ammann *et al.* verwendet

[Amal02]. Für dieses Vorgehen müssen die vorliegenden generischen Exploits und Bedingungen instanziiert werden. D.h. sie werden einem bestimmten Computer im Netzwerk zugeordnet. Diese Instanziierung wird von Ammann *et al.* allerdings nicht beschrieben, sondern als gegeben angenommen. Deswegen wurde ein eigenes Verfahren entwickelt.

In Abbildung 2 ist auf der linken Seite ein Netzwerk dargestellt. Es besteht aus drei Computern, die mit den Namen 0, 1 und 2 bezeichnet werden. Alle drei Computer sind miteinander über einen Router vernetzt, allerdings wird die Verbindung zwischen Computer 0 und 2 durch eine Firewall blockiert. Der Angreifer will Root-Rechte auf Computer 2 erlangen. Sein Angriff startet von Computer 0. Auf Computer 1 ist Linux und ein FTP-Server installiert. Auf Computer 2 ist ebenfalls Linux sowie RSH installiert. In diesem Beispiel wird davon ausgegangen, dass Linux einen lokalen Exploit im Kernel hat. Dieser kann mit User-Rechten verwendet werden und gibt dem Angreifer anschließend Root-Rechte. Außerdem können die Programme RSH und FTP durch einen Remote-Exploit angegriffen werden, wodurch der Angreifer User-Rechte bekommt.

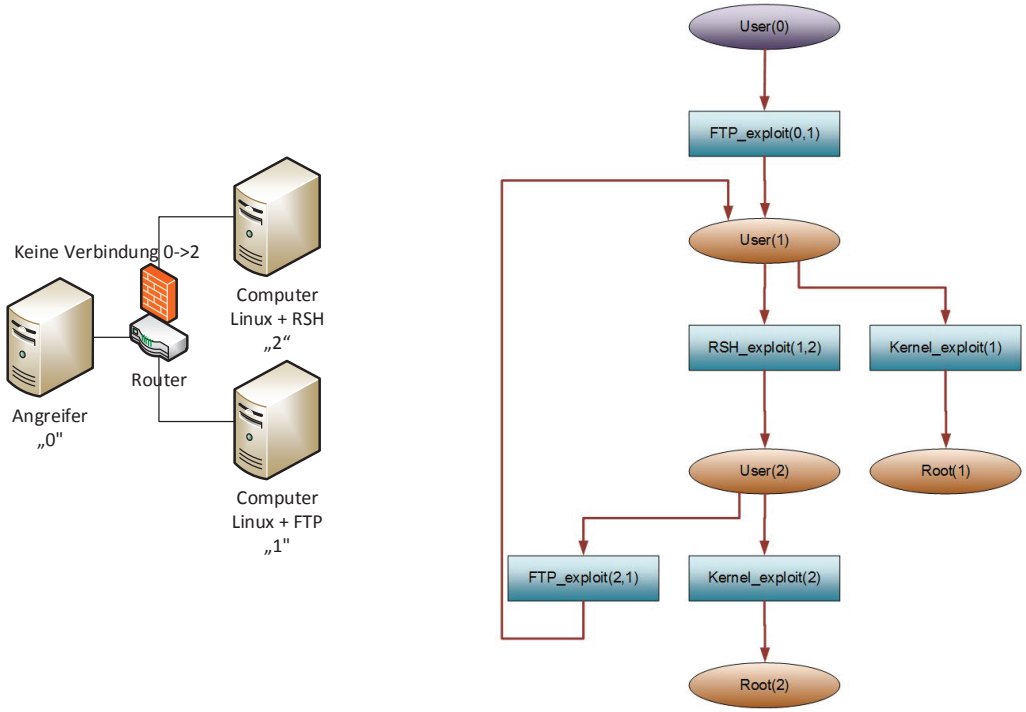

Abb. 2: Beispiel eines Netzwerks mit passendem Angriffsgraphen

Als initiale Startbedingung werden die User-Rechte auf dem Computer des Angreifers definiert: $User(0)$. Der Knoten ist im Angriffsgraphen ganz oben dargestellt. Der Algorithmus startet auf Computer 0 und analysiert die Software auf lokale Exploits. Da in diesem Beispiel keine Software auf dem Computer 0 installiert ist, wird als nächstes die Software auf den per Netzwerkverbindung angeschlossenen Computern analysiert. Der Remote-Exploit für der FTP Software kann verwendet werden, um User-Rechte auf Computer 1 zu erlangen. Dieser Pfad wird in den Graphen eingefügt. Die Analyse startet von vorne, mit den Computern auf denen in der vorherigen Runde neue Knoten eingefügt wurden. In diesem Fall Computer 1. Erst wird der lokale Exploit für den Kernel ($Kernel\_exploit(1)$) in den Graphen eingefügt, anschließend der Remote-Exploit für RSH ($RSH\_exploit(1,2)$). Dieses Verfahren wird fortgesetzt, bis keine neuen Bedingungen gefunden werden.

3.4 Angriffsgraph analysieren

Nachdem der Angriffsgraph vollständig erstellt wurde, kann die erste Analyse ausgeführt werden. Dabei werden überflüssige Pfade entfernt und eine aussagenlogische Formel für das spätere Netzwerk-Hardening generiert. Außerdem kann eine probabilistische Analyse durchgeführt werden, um die Werte von den generischen und predicted Exploits zu berücksichtigen.

3.4.1 Netzwerk-Hardening per DNF

Das Netzwerk-Hardening soll herausfinden, welche Bedingungsknoten gehärtet werden müssen, um das Angriffsziel vollständig zu schützen. Dafür wird zwischen zwei Knotentypen unterschieden, den Zwischenknoten und den Existenzknoten. Als Zwischenknoten werden alle Knoten bezeichnet, welche sich zwischen zwei Exploits befinden. Sie sind nur abhängig von den vorherigen Exploits und können nicht von außerhalb verhindert bzw. gehärtet werden. Die einzige Möglichkeit ist, die Existenzknoten zu härten und somit den Exploit zu verhindern. Der Exploit existiert aufgrund der Software und deren Schwachstelle. Bei Remote-Exploits zählt zusätzlich die Netzwerkverbindung als Existenzbedingung.

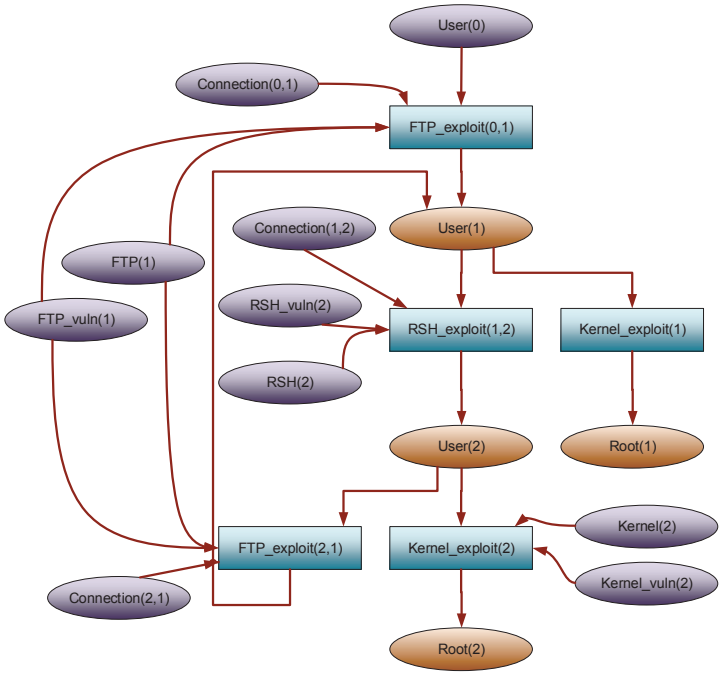

Abb. 3: Angriffsgraph mit Existenzbedingungen

In Abbildung 3 ist der Angriffsgraph aus Abbildung 2 mit den dazugehörigen Existenzknoten dargestellt. Pro Exploit sind die 2 bzw. 3 Existenzbedingungen eingefügt worden. Gleiche Existenzbedingungen werden zusammengefügt, da sie z.B. von der selben Vulnerability, Software oder der selben Netzwerkverbindung abhängen.

Um alle möglichen Kombinationen von Existenzbedingungen zu finden, welche ein Erreichen des Ziels verhindern, wird eine Methode von Wang et. al verwendet [WaAJ14]. Die Analyse für das Netzwerk-Hardening startet bei den Zielknoten und durchläuft den Angriffsgraphen rückwärts zu den Startknoten. Dabei wird eine aussagenlogische Formel entwickelt, wobei die Knoten des Graphen die Atome der Formel darstellen. Da der Angriff verhindert werden soll,

wird mit der Negation des Zielknoten gestartet. Beim Angriffsgraphen aus Abbildung 2 sieht der Start der Analyse wie folgt aus:

$$L = \neg Root(2)$$

Von dort läuft der Algorithmus an den Kanten rückwärts durch den Graphen. Er verhält sich dabei wie eine Breitensuche. Jeder Knoten wird durch seine Vorgänger in der Formel substituiert. Am vorherigen Beispiel wird dementsprechend wie folgt weiter gerechnet:

$$L = \neg Root(2) = \neg Kernel\_exploit(2)$$

Bei mehreren Vorgängern werden diese miteinander kombiniert. Die Art der Kombination hängt vom zu ersetzenden Knoten ab. Wenn es ein Exploitknoten ist, ist dieser ein UND-Knoten. Deswegen werden alle seine Vorgänger in einer Konjunktion zusammengefasst. Ein Bedingungsknoten ist ein ODER-Knoten, so dass dessen Vorgänger in einer Disjunktion zusammengefügt werden. Am Beispiel wird $\neg Kernel\_exploit(2)$, dementsprechend durch seine drei Vorgänger $[User(2) \wedge Kernel(2) \wedge Kernel\_vuln(2)]$ ersetzt. Die Substitution wird anschließend mit $User(2)$ weitergeführt. Durch den im Graphen enthaltenen Zyklus gelangt der Algorithmus über $FTP(2,1)$ erneut zu $User(2)$. Der Algorithmus merkt sich den gegangen Pfad und stellt so den Zyklus fest. Um diesen aufzubrechen, wird beim Ersetzen $FALSE$ statt $User(2)$ an dieser Stelle in die Formel eingesetzt. Da der Term eine Konjunktion ist, kann im nachfolgenden Schritt der ganze Term entfernt werden. Die gesamte Formel zu dem Beispiel lautet:

$$L = \neg[[User(0) \wedge Connection(2,1) \wedge FTP(1) \wedge FTP\_vuln(1)] \wedge Connection(1,2) \wedge RSH\_vuln(2) \wedge RSH(2)] \wedge Kernel(2) \wedge Kernel\_vuln(2)]$$

Diese Formel wird anschließend in eine Disjunktive Normalform (DNF) gebracht. Jede Konjunktion steht für eine Möglichkeit des Netzwerk-Hardening. Wenn die entsprechenden Existenzbedingungen gehärtet werden, kann das Angriffsziel nicht erreicht werden. Am Beispiel wird die Formel zuerst in die Negation Normalform gebracht, wodurch alle Aussagen negiert und ODER-verknüpft werden. Die Formel ist damit bereits vollständig umgewandelt. Es reicht eine der aufgeführten Existenzbedingung zu härten, um das Ziel zu schützen.

Abschließend werden alle nicht besuchten Knoten und Kanten aus dem Graphen entfernt. Sie führen nicht zum Angriffsziel und sind deswegen überflüssig.

3.4.2 Probabilistische Analyse

Die klassische TVA betrachtet die Netzwerksicherheit nur binär. Das Netzwerk ist entsprechend sicher oder unsicher. Durch die generischen Exploits und Konfigurationsfehler werden allerdings Exploits mit probabilistischen Werten in den Angriffsgraphen eingefügt.

Abhilfe schaffen Lingyu Wang und Sushil Jajodia mit ihrer probabilistischen Analyse [Waal08]. Dort erweitern sie die TVA zu einer probabilistischen Variante, in welcher jeder Exploitknoten einen probabilistischer Wert erhält. Aus diesen einzelnen Wahrscheinlichkeiten wird berechnet, wie wahrscheinlich das Erreichen des Knoten innerhalb des Netzwerkes ist.

Den Exploits *e* und Bedingungen *c* werden jeweils zwei probabilistische Werte zugewiesen. Zum einen der *individuelle Wert p(e)* bzw. *p(c)* und zum anderen der *Gesamtwert P(e)* bzw. *P(c)*. Der *individuelle Wert* ist die Wahrscheinlichkeit, dass der Exploit existiert. Bei Bedingungen wird der *individuelle Wert* als 1 angenommen.

Um den *Gesamtwert* der einzelnen Knoten zu berechnen werden die *individuellen Werte* zusammengerechnet. Die Berechnung folgt der Definition 1 und ist in Abbildung 4 dargestellt.

Die linke Darstellung repräsentiert den ersten, die mittlere den zweiten und die rechte Darstellung ist eine Kombination aus beiden Definitionsstichpunkten.

Definition 1 (vgl. [Waal08, S. 286])**.** *Gegeben ist ein azyklischer Graph $G(E \cap C, R_r \cup R_i)$, und jeweils ein individueller Wert p : $E \cup C \rightarrow [0,1]$, die Gesamtwertfunktion P : $E \cup C \rightarrow [0,1]$ ist definiert als*

- *P(e) = p(e)* $\cdot \prod_{c \in R_r(e)} P(c)$
- *P(c) = p(c), wenn* $R_i(c) = \phi$*; sonst,* $P(c) = p(c) \cdot \oplus_{e \in R_i(c)}$ *P(e) wobei der Operator* $\oplus$ *rekursiv definiert ist als* $\oplus P(e) = P(e)$ *für jedes* $e \in E$ *und* $\oplus(S_1 \cup S_2) = \oplus S_1 + \oplus S_2 - \oplus S1 \cdot \oplus S_2$ *für jeden disjunkten oder nicht leeren Set* $S_1 \subseteq E$ *und* $S_2 \subseteq E$*.*

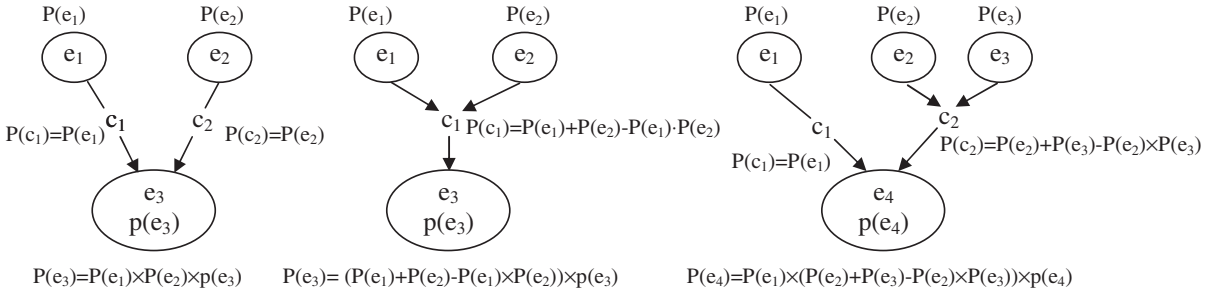

Abb. 4: Berechnung der probabilistischen Gesamtwerte [Waal08]

Die bis hierhin vorgestellten Berechnungen betrachten einfache azyklische umgedrehte Bäume. Der Angriffsgraph ist allerdings ein ggf. zyklischer Graph. Deswegen können zusammenführende Pfade gemeinsame Abhängigkeiten besitzen und es können Zyklen vorhanden sein. Dies wird in den nächsten beiden Abschnitten betrachtet.

D-Separation

Die gemeinsamen Abhängigkeiten werden in der probabilistischen Analyse von Homer *et al.* berücksichtigt [Hoal13]. In dem Artikel wird die Idee der D-Separation aus den bayesischen Netzen adaptiert. Diese ermöglicht eine bedingte Unabhängigkeit zwischen zwei Knotenmengen herzustellen, welche eine gemeinsame Abhängigkeit besitzen. Die Definition der D-Separation wird für diesen Zweck an den Angriffsgraphen angepasst. G_N ist die Menge aller Knoten des Angriffsgraphen.

Definition 2 ([Hoal13, S. 9])**.** *Zwei disjunkte Knotenmengen S_1 und S_2 sind in einem Angriffsgraphen durch eine Knotenmenge $V \subseteq G_N$ (disjunkt zu S_1 und S_2) d-separiert, wenn entlang jedes divergierenden Pfades zwischen S_1 und S_2 ein $v \in V$ existiert, so dass v der Punkt der Divergenz ist.*

Am Beispiel von Abbildung 5 existiert ein divergierender Pfad über P_1 zu A_2 und A_3. Die gemeinsame Abhängigkeit und damit der Punkt der Divergenz ist P_1. Also d-separiert P_1 die beiden Knoten A_2 und A_3. Werden die beiden Knoten bei einem gegebenen P_1 betrachtet, sind sie bedingt unabhängig. Daraus lässt sich ein Theorem erstellen.
Theorem 1 ([Hoal13, S. 9])**.** *Lasse die Knotenmengen D und N so sein, dass D jedes Paar von*

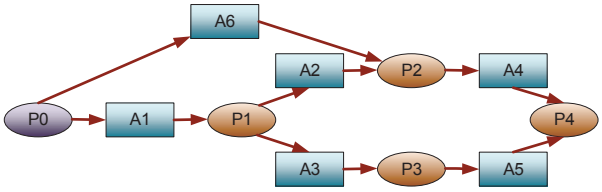

Abb. 5: Ein Angriffsgraph mit D-Separation [Hoal13, S. 8]

Knoten in N d-separiert. Dann:

$$P[N] = \sum_{D}(\prod_{n \in N} P[n|D]) * P[D]$$

Angewandt wird dieses Theorem um z.B. P_2 zu berechnen:

$$\begin{aligned} P[P2] &= 1 - P[\overline{A_2}, \overline{A_6}] \\ &= 1 - \sum_{P_0} P[\overline{A_2}|P_0] * P[\overline{A_6}|P_0] * P[P_0] \\ &= 1 - ((1 - p(A_1) * p(A_2))(1 - p(A_6))(1) + (1)(1)(0)) \end{aligned}$$

Probabilistische Analyse bei Zyklen

Die probabilistische Analyse von Angriffsgraphen mit D-Separation ist mit den im vorherigen Abschnitt vorgestellten Theorem möglich. Allerdings sind Angriffsgraphen keine Bäume, sondern Graphen und können somit Zyklen beinhalten. Diese Zyklen müssen separat betrachtet werden, da der Algorithmus diese unendlich oft durchlaufen würde.

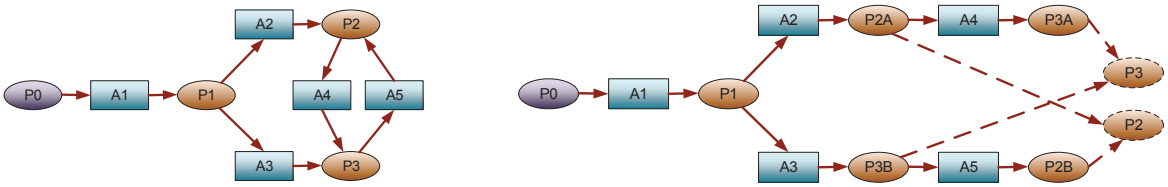

Abb. 6: Ein Angriffsgraph mit Zyklus (links), sowie die entfaltete Version (rechts) [Hoal13, 16]

In der Abbildung 6 ist links ein Angriffsgraph dargestellt, welcher einen Zyklus beinhaltet. Die probabilistische Analyse aus den vorherigen Abschnitten wird zunächst unverändert verwendet. D.h. der Angriffsgraph wird von P_0 bis zu den Knoten P_2 und P_3 analysiert. Für diese beiden Knoten stellt die Analyse fest, dass Teile ihrer Vorgänger nicht berechnet wurden.

Auf der rechten Seite der Abbildung 6 ist der gleiche Angriffsgraph dargestellt, allerdings entfaltet. Er beinhaltet alle einzigartigen Pfade, die auch die linke Version besitzt. Zum Beispiel wird Knoten P_3 durch zwei Pfade erreicht. Zum einen über $A_3 \rightarrow (P_{3B} \rightarrow)P_3$ und zum anderen $A_2 \rightarrow (P_{2A} \rightarrow)A_4 \rightarrow (P_{3A} \rightarrow)P_3$. Die eingeklammerten Knoten sind neu eingefügte Knoten. Sie können als partielle Werte angesehen werden. Wenn die partiellen Werte wahr werden, wird auch der dazugehörige Knoten wahr. Im rechten Graphen wird dies durch gestrichelte Linien dargestellt. Würden im Beispiel P_{3A} oder P_{3B} wahr, wird dadurch auch P_3 wahr.

Das Entfalten des Graphen wird von der probabilistischen Analyse durch eine Datenflussanalyse durchgeführt. Dabei ist zu beachten, dass der Angriffsgraph nur theoretisch entfalten wird und nicht permanent. Durch die ermittelten Pfade aus der Datenflussanalyse können die Eingangsknoten berechnet werden. Dafür werden alle Pfade analysiert deren letztes Element der Eingangsknoten ist. Für P_2 sind dies die beiden Pfade $A_3 \rightarrow P_3 \rightarrow A_5 \rightarrow P_2$ und $A_2 \rightarrow P_2$. Für die Berechnung werden die Formeln aus den vorherigen Abschnitten verwendet.

Probabilistisches Netzwerk-Hardening

Die in Abschnitt 3.4 vorgestellte DNF-Berechnung zum Auffinden von Hardening-Strategien berücksichtigt nicht die probabilistischen Werte, sondern betrachtet die Sicherheit ausschließlich binär. Ein Systemadministrator will sein System allerdings nicht zwangsläufig vollständig sichern, da einige Angriffe zu unwahrscheinlich oder die Kosten zum Verhindern des Angriffs zu teuer sind.

Um dem Systemadministrator auch in diesem Fall eine sinnvolle Auswahl von Hardening-Strategien zu präsentieren, werden alle Kombinationen von Existenzbedingungen durchlaufen. Jede Kombination stellt eine Hardening-Strategie dar. Der probabilistische Wert der jeweiligen Knoten wird von 1 auf 0 gesetzt und die probabilistische Analyse wird jeweils durchgeführt. Für jede Kombination wird anschließend der berechnete probabilistische Wert des Angriffsziel gespeichert. Dieser dient als Vergleichswert, um eine Rangliste von Hardening-Strategien zu erstellen. Je kleiner der Wert ist, desto besser ist das Netzwerk gesichert.

Das vorgestellte Verfahren ist rechenintensiv und kann optimiert werden. Die mit Hilfe der DNF gefundenen Kombinationen härten das Netzwerk vollständig vor einem Angriff. Der Wert des Angriffsziel und damit auch der Kombination ist 0. Deswegen brauchen diese Kombinationen nicht durch die probabilistische Analyse berechnet werden. Auch jede Erweiterung der Kombinationen durch zusätzliche Existenzbedingungen ergibt 0.

4 Software

Das vorgestellte Konzept wurde in einem Java-Programm umgesetzt. In Abbildung 7 ist auf der linken Seite das Java-Programm dargestellt. Auf der rechten Seite ist ein mit dem Programm inventarisiertes Netzwerk abgebildet, welches mit Hilfe von MS Visio visualisiert wird.

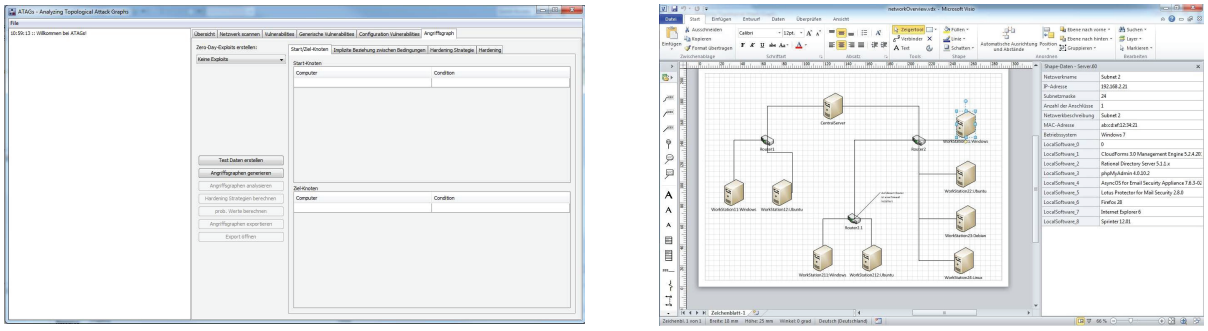

Abb. 7: Entwickeltes Java-Programm (links) und damit erstelltes Netzwerkdiagramm in Visio (rechts)

5 Fazit

In neuerer Zeit kommen zunehmend mehrschrittige Angriffe zum Einsatz unter Verwendung von unbekannten Schwachstellen. In dieser Arbeit wurde ein ganzheitliches semi-automatisches Verfahren vorgestellt, um Netzwerke auf diese mehrschrittigen Angriffe zu untersuchen und zu härten. Insbesondere wurde der Fokus auf die Berücksichtigung von unbekannten Schwachstellen gelegt und dazu wurde die TVA um eine probabilistische Analyse erweitert.

Das Verfahren beginnt mit der automatischen Inventarisierung eines realen Netzwerks und findet automatisch die bekannten Schwachstellen sowie wahrscheinliche Schwachstellen. Unter Berücksichtigung aller möglichen Angriffspfade sowie deren Existenzwahrscheinlichkeit berechnet der Ansatz mögliche Hardening-Strategien. Dadurch kann das Netzwerk individuell vor mehrschrittigen Angriffen gesichert werden.

Zukünftig sollen die Kosten für das Härten von Netzwerkelementen mit berücksichtigt werden. Daraus soll ein Gesamtkonzept entstehen, welches die Kosten des Netzwerk-Hardening mit der probabilistischen Analyse verbindet. Außerdem soll ein semi-automatisches Verfahren entwickelt werden, um die Kosten für das Härten abzuschätzen.

Literatur

[Alal05] O. H. Alhazmi, et al.: Quantitative vulnerability assessment of systems software. *In: Proc. annual reliability and maintainability symposium* (2005), 615–620.

[Amal02] P. Ammann, et al.: Scalable, graph-based network vulnerability analysis. *In: Proceedings of the 9th ACM CCS*, ACM (2002), 217–224.

[Artz02] M. L. Artz: Netspa: A network security planning architecture. Dissertation, Massachusetts Institute of Technology (2002).

[bey] beyondtrust – Retina: http://www.beyondtrust.com/Products/RetinaNetworkSecurity Scanner, aufgerufen am 06.08.2014.

[Chal10] M. Chu, et al.: Visualizing attack graphs, reachability, and trust relationships with NAVIGATOR. *In: Proceedings of the 7th ACM VizSec* (2010), 22–33.

[CoS] Core Security: CoreImpact, http://www.coresecurity.com/core-impact-pro, aufgerufen am 06.08.2014.

[FaMC11] N. Falliere, L. O. Murchu, E. Chien: W32. stuxnet dossier. In: *White paper, Symantec Corp., Security Response* (2011).

[Frie06] J. Friedl: Mastering Regular Expressions. O'REILLY, Sebastopol (2006), s. 38ff.

[Hoal13] J. Homer, et al.: Aggregating vulnerability metrics in enterprise networks using attack graphs. In: *Journal of Computer Security*, 21, 4 (2013), 561–597.

[JaNo10] S. Jajodia, S. Noel: Topological vulnerability analysis. *In: Cyber Situational Awareness*, Springer (2010), 139–154.

[Koal14] B. Kordy, et al.: DAG-Based Attack and Defense Modeling: Don't Miss the Forest for the Attack Trees. In: *Computer science review*, 13 (2014), 1–38.

[Lyon] G. F. Lyon: Nmap http://nmap.org, aufgerufen am 15.04.2014.

[Lyon09] G. F. Lyon: Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Insecure (2009).

[MaCr] S. MacGuire, R.-A. Croteau: BigBrother, http://www.bb4.org, aufgerufen am 06.08.2014.

[NgMa12] V. H. Nguyen, F. Massacci: An independent validation of vulnerability discovery models. *In: Proceedings of the 7th ASIACCS*, ACM (2012), 6–7.

[Ope15] Open Vulnerability Assessment System: http://www.openvas.org/about.html, aufgerufen am 21.02.2015 (2015).

[OuAp05] X. Ou, A. W. Appel: A logic-programming approach to network security analysis. Princeton University Princeton (2005).

[OuBM06] X. Ou, W. F. Boyer, M. A. McQueen: A scalable approach to attack graph generation. *In: Proceedings of the 13th ACM CCS*, ACM (2006), 336–345.

[OuGA05] X. Ou, S. Govindavajhala, A. W. Appel: MulVAL: A Logic-based Network Security Analyzer. *In: USENIX Security* (2005).

[PhSw98] C. Phillips, L. P. Swiler: A graph-based system for network-vulnerability analysis. *In: Proceedings of the 1998 workshop on New security paradigms*, ACM (1998), 71–79.

[QuS] Qualys Solutions, https://www.qualys.com, aufgerufen am 06.08.2014.

[Rapi] Rapid7: NexPose, https://www.rapid7.com/de/products/nexpose, aufgerufen am 06.08.2014.

[Resc05] E. Rescorla: Is finding security holes a good idea? In: *Security & Privacy, IEEE*, 3, 1 (2005), 14–19.

[Sec] Secunia ApS: Corporate Software Inspector, https://secunia.com/vulnerability_scann ing, aufgerufen am 25.01.2015.

[Ten] Tenable Network Security: Nessus, http://www.tenable.com/products/nessus-vulnerability-scanner, aufgerufen am 21.02.2015.

[WaAJ14] L. Wang, M. Albanese, S. Jajodia: Network Hardening: An Automated Approach to Improving Network Security. SpringerBriefs in Computer Science (2014).

[Waal08] L. Wang, et al.: An attack graph-based probabilistic security metric. *In: Data and applications security XXII*, Springer (2008), 283–296.

[Waal10] L. Wang, et al.: k-zero day safety: Measuring the security risk of networks against unknown attacks. *In: Computer Security–ESORICS 2010* (2010), 573–587.

[WiLI08] L. Williams, R. Lippmann, K. Ingols: GARNET: A graphical attack graph and reachability network evaluation tool. Springer (2008).

[WJSC+13] L. Wang, S. Jajodia, A. Singhal, P. Cheng, S. Noel: k-Zero day safety: A network security metric for measuring the risk of unknown vulnerabilities. In: (2013).

[Wolf02] M. Wolfgang: Host Discovery with nmap. In: *Exploring nmap's default behavior*, 1 (2002), 16.