Seminararbeit Attack-Graphs

Lennart Steffin
B_WInf1026, Semester 6
21.07.2014

Seminar Wirtschaftsinformatik
Fachhochschule Wedel
Professor: Gerd Beuster
WS 2013 / 14

Inhaltsverzeichnis

Abkürzungen

Abb.	Abbildung
AG	Attack-Graph
BKA	Bundeskriminalamt
Bzw.	Beziehungsweise
G	Graph
I.d.R.	In der Regel
IDS	Intrusion Detection System
IP / IP-Adresse	Internet-Protokoll-Adresse
IT	Informationstechnik
o.g.	Oben genannte(s)
PAG	Probabilistischer Attack-Graph
Q#	Quelle Nr. # (siehe Quellen)
s.o.	Siehe oben
u.a.	Unter Anderem
Vergl.	Vergleiche
z.B.	Zum Beispiel

Abbildungen

Präambel

Dieses Dokument ist eine Betrachtung des wissenschaftlichen Artikels „Two Formal Analyses Of Attack Graphs“ von S. Jha (Computer Sciences Department, University of Wisconsin Madison), O. Sheyner und J. Wing (Computer Science Department, Carnegie Mellon University, Pittsburgh) [Q1] und wird im Folgenden als „das Paper“ bezeichnet. Sofern nicht anders zitiert, beziehen sich Fakten stets auf diesen Artikel.

Der Artikel gibt, nach einem Grundgerüst, zwei formale Analysen für Attack-Graphen wieder. Dies ist zum einem eine Minimierungsanalyse, die aufzeigt, wie Gegenmaßnahmen am effizientesten ausgewählt werden können. Zum anderen eine probabilistische Analyse, die versucht, die Wahrscheinlichkeit erfolgreicher Angriffe zu mindern.

Der Fokus dieser Betrachtung ist der generelle Einsatz und Nutzen von Attack-Graphen in Unternehmensnetzen. Um eine weitere Facette dieser Methode aufzuzeigen, wurde die Erweiterung von Attack-Graphs zu Ranked Attack-Graphs aus dem Artikel „Raking Attack Graphs“ von Vaibhav Mehta, Constantinos Bartzis, Haifeng Zhu, Edmund Clarke und Jeannette Wing (Carnegie Mellon University, Pittsburgh) [Q6] einbezogen.

Aktuelle Situation

Das Wachstum des Internets ging mit der Entstehung der Cyber-Kriminalität einher; viele Einzelpersonen und kriminelle Organisationen haben seither versucht, Unternehmensdaten über das Internet zu erbeuten und gewinnbringend zu verwenden.
Wie das BKA (Bundeskriminalamt) im Cybercrime Bundeslagebild 2012 resümiert wird *„der Bereich Cybercrime auch in den kommenden Jahren ein wachsendes Phänomen darstellen, dessen Bekämpfung die Sicherheitsbehörden sowohl präventiv als auch repressiv im Sinne eines ganzheitlichen Ansatzes fortsetzen müssen“*. [Q7]

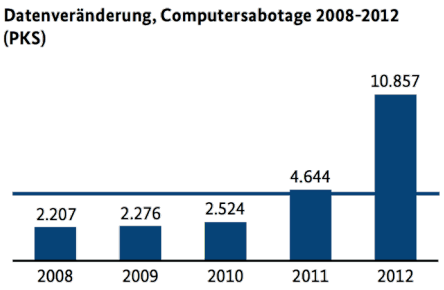
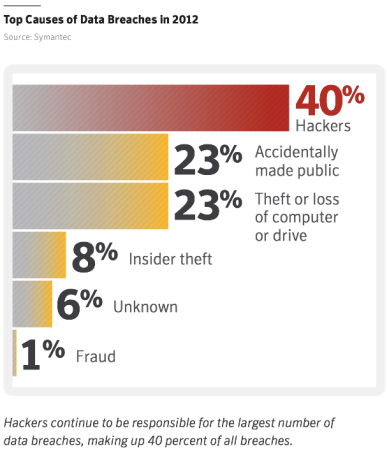


Abb.1: Datenveränderung, Computersabotage 2008-2012, Bundeskriminalamt Cyberkriminalität

Diese Aussage trifft auch den Kern zahlreicher Analysen im Bereich Cyberkriminalität, wie beispielsweise die des Anti-Viren-Softwareherstellers Symantec, die u.a. Hacker als die größte Gefahr – noch vor Mitarbeiterfehlern oder Geräteverlust/-diebstahl – skizzieren oder ThreatTrack Security, die das Volumen und die Komplexität der Schadsoftware im Internet (Malware) als größte Schwierigkeit einschätzen. [Q8]



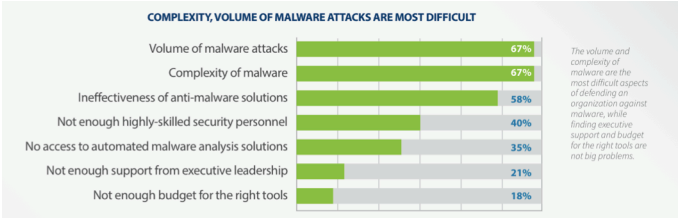


Abb.2 (links): Top Causes of Data Breaches 2012, Symantec
Abb.3 (rechts): Complexity, Volume of Malware Attacks, ThreatTrack

Prävention

Um sich vor Hacker-Angriffen zu schützen, feilen Unternehmen kontinuierlich an der Sicherheit ihrer Systeme und Netzwerke – diverse neue Berufsbilder und Spezialisierungen sind hieraus hervorgegangen: IT-Sicherheitsanalyst, Datenschützer, Sicherheitstechniker und viele mehr.
Um das unternehmensinterne Netz zu sichern, erstellen Analysten umfangreiche Karten, welche die Struktur und die Gefahren, bzw. Schwachstellen des Unternehmensnetzes aufzeigen, um diese kontinuierlich zu verbessern.

Anoop Singhal und Ximming Ou [Q3; S. iv] fassen zusammen, dass die Unternehmen heute raffinierten Angreifern, welche mehrere Sicherheitslücken nutzen um verheerenden Schaden anzurichten, gegenüberstehen. Um sich zu schützen, so Singhal und Ou, müssen Unternehmen verstehen, wie Schwachstellen erkannt und ausgebeutet werden können.
Der grundlegende Schritt für Unternehmen dieses zu erreichen, sind Attack-Graphs. Dies sind gerichtete Graphen, die Angriffssequenzen auf ein Netzwerk darstellen, die zu einem kritischen Zustand führen. Da Einzelangriffe meist nicht direkt kritisch sind, wird so analysiert, ob und wie Angreifer über eine Sequenz an Einzelangriffen die Chance haben, einen solchen Systemzustand zu erzeugen. Attack-Graphs können heutzutage durch einen Algorithmus erstellt werden, der aus dem Unternehmensnetz und den einzelnen Schwachstellen über eine gewisse Verkettung kritische Zustände folgert.
Während dies früher händisch von Red Teams – also Teams deren Aufgabe es ist, gezielt sensible Unternehmsteile wie Produkte, Strategien oder Sicherheit herauszufordern – gemacht wurde, ist die Generierung heute i.d.R. automatisiert, insbesondere da es impraktikabel ist, Netze heutiger Größen fehlerfrei zu übertragen.

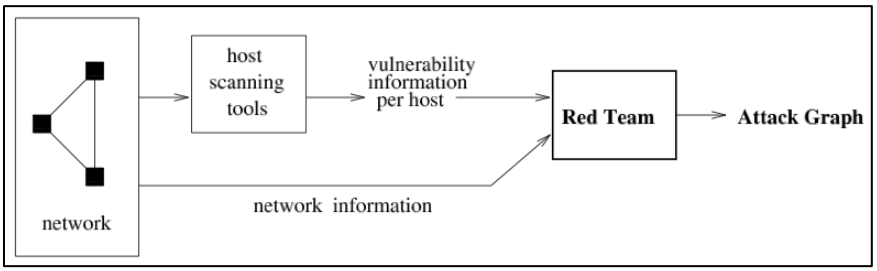


Abb.4: Händischer Prozess der Erstellung von Attack Graphs durch Red Teams

Am Ende der Generierung steht schließlich ein Graph, der alle Wege zu einem vorgegebenen kritischen Zustand aufzeigt und aus atomaren Angriffen besteht.
Atomare Attacken sind einzelne Angriffe, die durchgeführt werden können, um die Rechtestufe des Angreifers im Gesamtsystem zu erhöhen. Das bedeutet auch, dass keine Zyklen im Graphen entstehen können; dies erleichtert die Analyse des Graphen. Laufen Zustände wieder zu Einem zusammen, wurde der gleiche Zustand im Gesamtsystem erzeugt.
Die Endzustände (Zustände auf der untersten, finalen Ebene) im Graphen sind die zu erreichenden kritischen Zustände, welche vor Generierung festgelegt wurden – wie etwa Root-Rechte auf dem System mit der IP_X. Bei jedem Endzustand besteht ein anderer Rechtezustand im Gesamtsystem.

Beispiel: Ein Angreifer der erst A und dann B angreift, landet anschließend im gleichen Zustand, wie der, der erst B dann A (mit gleichen atomaren Attacken) angreift. Wer vorher C, dann A und B angreift, landet in einem neuen Zustand, da er zusätzlich Rechte auf C hat.

Intrusion Detection System

Hilfe gegen Angriffe bietet das Intrusion Detection System (IDS), ein Verteidigungsmechanismus, der von den Administratoren eines Netzwerks kontrolliert wird und Hacker-Angriffe aufspüren soll. Der Mechanismus reagiert durch gewisse Software-Sensorik im Netzwerk, analysiert, welche verdächtigen Aktivitäten begangen werden und löst ggf. Alarm aus.
Die Administratoren müssen dabei einen Trade-Off aus Verlässlichkeit und Irrtumshäufigkeit eingehen.
Um dieses Problem zu begrenzen, nutzen IDS oft einen großen Pool heterogener Sensoriken, welche durch heuristische Algorithmen die derzeit besten Ergebnisse verschaffen. Dabei helfen Attack-Graphs, die Fehlerquote zu reduzieren, indem Aktionen mit naheliegenden Folgeangriffen verglichen werden können. Dadurch kann die Trefferquote von Angriffen deutlich gesteigert werden. Neben dem Aufspüren von Angriffen, dient das IDS auch zur Verteidigung und zur forensischen Analyse von Angriffen.
Bei der Verteidigung können Attack-Graphen helfen, vorbeugende Sicherungen einzubauen, sodass kritische Punkte gegen Angriffe abgesichert werden. Nach Angriffen können forensische Analysen die Spuren des Eindringlings aufdecken und somit weitere Schwachstellen im Netzwerk identifizieren.

Der Attack-Graph

Model Checking Algorithmus

Der Attack-Graph Algorithmus ist ein Model Checking Algorithmus, der mittels einer vollständigen Suche durch den Baum alle validen Endzustand sucht.

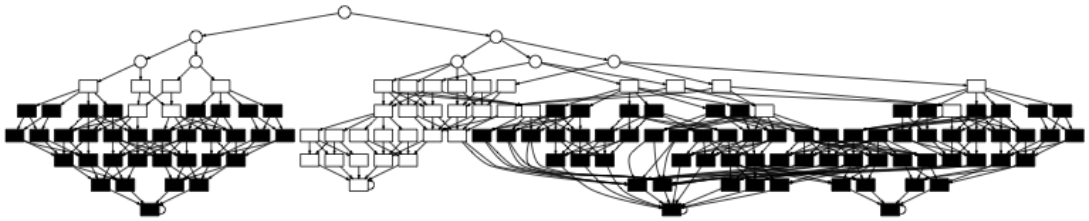


Abb.5: Visualisierung eines Attack-Graphs mit schwarzen und weißen Knoten

"Model checking is an automatic technique for verifying correctness properties of safety-critical reactive systems" [Q4; S. 1637–1790]

Formal: $M \models \Phi$
(„Ist Systembeschreibung M ein Modell für die logische Eigenschaft Φ?")
Erfüllt unser Graph die Eigenschaft, dass bestimmte Sicherheitseigenschaften immer gewährleistet sind?
Um diese Eigenschaft zu definieren, existiert eine Sicherheitsrestriktion (p) in Form einer aussagenlogischen Formel. Diese Logik wird durch CTL beschrieben – eine Temporallogik, welche im Gegensatz zu Aussagenlogiken temporale Aussagen beinhalten kann und somit Ausdrucksstärker, aber weiterhin entscheidbar ist. CTL ist, wie die *Lineare temporale Logik* (LTL) eine Teilmenge der Obermenge CTL*.
Dieser Prozess nennt sich Model Checking und ist somit ein automatisiertes Verfahren, um die Korrektheit eines solchen Systems zu testen.

Bedingungen

Die Konstruktion des Attack-Graphs untersteht drei Lemmata, die für die Generierung zu erfüllen sind:

a) *erschöpfbar*: Die Ausführung ***e*** des formalen Eingangsmodells $\boldsymbol{M=(S, R, S_0, L)}$ verstößt gegen die Sicherheitsrestriktion $\boldsymbol{p=AG(\neg unsafe)}$ genau dann, wenn ***e*** ein Angriff im Attack-Graph $\boldsymbol{G=(S_{unsafe}, R_p, S_{0p}, S_{sp}, L)}$ ist.
b) *Prägnant im Hinblick auf die Zustand*: Ein Zustand ***s*** des Eingangsmodells $\boldsymbol{M=(S, R, S_0, L)}$ ist im Attack-Graph ***G*** genau dann vorhanden, wenn es einen Angriff in ***G*** gibt, der ***s*** beinhaltet.
c) *Prägnant im Hinblick auf die Kanten:* Eine Kante (auch *Übergang*) $\boldsymbol{t=(s_1, s_2)}$ des Eingangsmodells ist im Attack-Graph ***G*** genau dann vorhanden, wenn es einen Angriff in ***G*** gibt, der ***t*** beinhaltet.

Attack Graph Algorithmus

Input

Der Attack-Graph wird aus einem 5-Tupel erstellt, das sich aus

S – der Menge aller Zustände -,
R – aller Verbindungen / Übergänge –,
S_0 – aller Initialzustand -,
L – der Kennzeichnung aller Zustände mit aussagenlogischen Formeln
p – der Sicherheits-Restriktion
bildet.

Output

Der Output der Generierung ist der Attack-Graph G_p, der alle Zustände und Übergänge enthält, die auf dem ‚Weg' zu einem unsicheren Endzustand zu durchlaufen sind. G_P wird aus einem 5-Tupel gebildet, dieses Tupel besteht aus der Menge der gefundenen unsicheren Zustände S_{unsafe}, der Verbindungen der Zustände R^p, der Initialzustand $S_0{}^p$, der Endzustand $S_S{}^p$ und der Aussagenformeln L.

Algorithmus

Der skizzierte Algorithmus (hier *GenerateAttackGraph* genannt) mit den genannten Eingangsgrößen geht nach einem einfachen Schema vor. Zunächst wird ein Model Check (s.o.) über den Graph durchgeführt. Die möglichen Verbindungen der resultierenden unsicheren Zustände (S_{unsafe}) werden dann R^p zugewiesen. Alle Initialzustände, die auch zu den unsicheren Zuständen gehören, werden $S_0{}^p$ zugewiesen. Alle Erfolgszustände werden $S_s{}^p$ zugewiesen und das 5-Tupel wird von der Funktion zurückgegeben.

Input:
S – set of states
$R \subseteq S \times S$ – transition relation
$S_0 \subseteq S$ – set of initial states
$L : S \rightarrow 2^{AP}$ – labeling of states with propositional formulas
$p = \mathbf{AG}(\neg unsafe)$ (a safety property)
Output:
attack graph $G_p = (S_{unsafe}, R^p, S_0^p, S_s^p, L)$
Algorithm: $GenerateAttackGraph(S, R, S_0, L, p)$
(* Use model checking to find the set of states S_{unsafe} that violate the safety property $\mathbf{AG}(\neg unsafe)$. *)
$S_{unsafe} = modelCheck(S, R, S_0, L, p)$.
(* Restrict the transition relation R to states in the set S_{unsafe} *)
$R^p = R \cap (S_{unsafe} \times S_{unsafe})$.
$S_0^p = S_0 \cap S_{unsafe}$.
$S_s^p = \{s | s \in S_{unsafe} \wedge s \models unsafe\}$.
$return(S_{unsafe}, R^p, S_0^p, S_s^p, L)$.

Abb.6: Formale Definition des Attack-Graph Algorithmus

Exemplarischer Durchlauf

Um die theoretische Darstellung des Algorithmus eingängiger zu erklären, soll folgendes Beispiel dienen.

Grundaufstellung

Es existiert folgendes Unternehmensnetz:

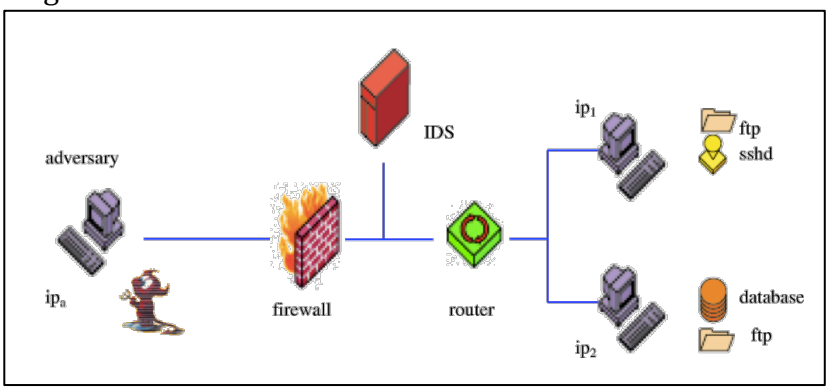


Abb.7: Attack-Graph Beispiel – Das Unternehmensnetz

In diesem Szenario sind zwei Computer (ip_1, ip_2) an einen Router angeschlossen. Computer ip_1 benutzt einen FTP-Zugriff und einen ssh-Dienst. Computer ip_2 hat eine Datenbank und ebenfalls einen FTP-Zugriff.
Das Unternehmensnetz ist durch eine Firewall zur Außenwelt getrennt. Ein angreifendes System (adversary / ip_a) steht außerhalb der Firewall. Zwischen Firewall und Router ist ein IDS angeschlossen.

Zustand des Endlichen Automaten-Modells

Der Model Checking-Algorithmus basiert auf einem endlichen Automaten. Damit Zwischenstände gespeichert werden können, wird eine Tabelle zu Hilfe gezogen. Insgesamt werden Daten über Services von Hosts, Angriffsmöglichkeiten von Hosts, Verbindungen unter den Hosts und dem Vertrauensverhältnis von Hosts für remote-logins benötigt.

Zugriffe

Um die Zugriffsmöglichkeiten während des Durchlaufens zu speichern, wird für jede mögliche Verbindung ein Tripel aus Boolean-Werten in einer Tabelle gespeichert. In diesem Tripel ist vermerkt, ob die beiden Hosts a) physikalisch verbunden sind b) sich per FTP oder c) sich über den sshd-Port verbinden können.

R	ip_a	ip_1	ip_2
ip_a	y,n,n	y,y,y	y,y,n
ip_1	y,n,n	y,y,y	y,y,n
ip_2	y,n,n	y,y,y	y,y,n

Abb.7: Aufbau des Beispiels – Darstellung der Zugriffsmöglichkeiten

Angriffe

In diesem Beispiel gibt es vier verschiedene, atomare Angriffe:

0. sshd buffer overflow (dieser remote-to-root Angriff gibt dem Angreifer sofortigen root shell-Zugriff auf der Zielmaschine),

1. ftp .rhosts (hier wird eine Verwundbarkeit genutzt, damit eine .rhosts Datei im FTP-Verzeichnis manipuliert werden kann, um als vertrauenswürdig zu gelten),
2. remote login (der Angreifer übernimmt eine existente Remote-Verbindung zweier Maschinen um – ohne Passworteingabe – Zugriff zum verbundenen System zu erhalten), sowie
3. local buffer overflow (der Angreifer hat die user shell erobert. Er kann anschließend durch einen *Local Buffer Overflow* mittels *setuid* root-Rechte der Maschine erbeuten).

Das Ziel des Angreifers ist die Zerstörung der Funktionalität der Datenbank. Dafür benötigt der Angreifer root-Rechte auf der Datenbank von Host ip_2.

Im System gibt es drei Rechtestufen. In aufsteigender Form: none, user, root – von keinen Rechten, über Benutzerrechte bis administrative Rechte.

Instrusion Detection System

Angriffe werden in *aufspürbar* (detectable) und *unsichtbar* (stealthy) unterschieden. Bei Ersteren wird das IDS einen Alarm auslösen.

Generierung des Attack-Graphs

Aus den genannten Daten und Methoden kann schließlich der Attack-Graph generiert werden. Dieser Graph besteht in oberster Ebene aus den Einstiegsknoten, welche über mehrere Ebenen mit den jeweils möglichen atomaren Angriffen verknüpft sind, bis die kritischen Endknoten erreicht sind, welche die Sicherheitsrestriktion verletzen. Führt kein Weg zu einem solchen Endknoten, ist der Attack-Graph leer – das Netzwerk ist somit nach aktuellem Kenntnisstand ‚sicher'.
Als Beispiel wird im Paper ein Attack-Graph angeführt, welcher aus drei Initialzustand zu zwei Endzuständen gelangt. Folgender farbig markierter Weg soll hier exemplarisch der Erklärung dienen:

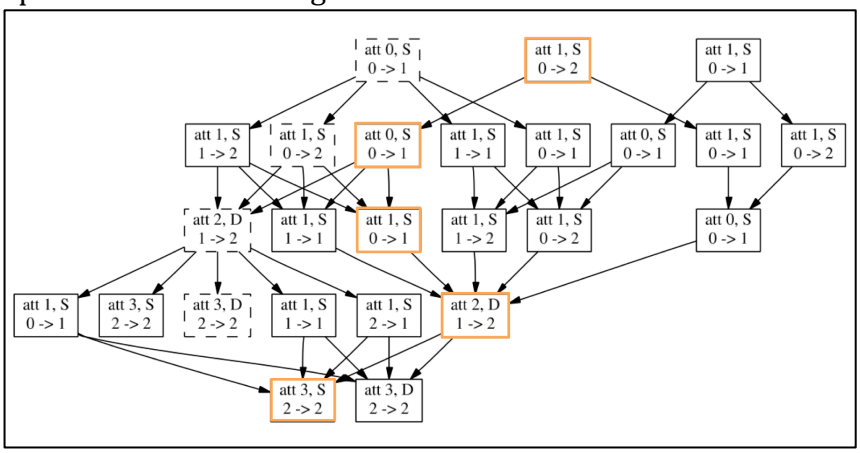


Abb.9: Attack-Graph mit exemplarischem Durchlaufweg in orange

- Als Einstiegspunkt wählt der Angreifer ip_a hier einen *FTP .rhosts* Angriff (vergl. *„att 1"*) auf ip_2 (vergl. *„0->2"*), welcher *stealthy* (vergl. *„S"*) durchgeführt werden kann. Der Angreifer gilt im FTP-System von ip_2 nun als vertrauenswürdig.

- Als nächstes nutzt der Angreifer einen *sshd buffer overflow* (vergl. *„att 0“*) gegen ip_1 (vergl. *„0->1“*), welcher auch *stealthy* durchgeführt werden kann. Der Angreifer verfügt nun über root shell-Rechte im Opfersystem.
- Identisch zum ersten Angriff kann ip_a hier einen *FTP .rhosts* Angriff (vergl. *„att 1“*) auf ip_1 (vergl. *„0->1“*) *stealthy* (vergl. *„S“*) durchführen. Der Angreifer gilt nun auch im FTP-System von ip_1 als vertrauenswürdig.
- Auf Grundlage der root shell-Rechte auf ip_1 und die Vertrauensbeziehung von ip_1 und ip_2 kann der Angreifende nun den Remote-Login (vergl. *„att 2“*) auf ip_2 (vergl. *„1->2“*) nutzen. detectable (vergl. *„D“*), hat aber dabei Zugriff auf das System von ip_2.
- Im letzten Schritt nutzt der Angreifer die Verbindung der Systeme um einen local buffer overflow (vergl. *„att 3“*) zu erzeugen und root-Rechte für ip_2 (vergl. *„2->2“*) zu bekommen. Dieser Schritt ist stealthy, der Angreifer hat nun sein Ziel erreicht.

Minimierungsanalyse

Im Rahmen der Minimierungsanalyse werden Optionen zur Sicherung des Netzwerkes analysiert und verglichen. Eine denkbare Menge an implementierbaren Maßnahmen wird hier als M' abgekürzt.
Im Paper werden folgende drei Fragestellungen hervorgehoben:

- Welche erfolgreichen Attacken werden vom IDS nicht erkannt?
- Wird ein Netzwerk sicher(er), wenn alle Maßnahmen aus M' implementiert werden?
- Welche Maßnahmen müssen ergriffen werden, um das Netzwerk sicher zu machen?

Im Paper wird dies ausführlich geschildert und in vertiefende Paper verlinkt. An dieser Stelle sollen die Inhalte nur grob skizziert werden, da der Fokus auf einer zusammenfassenden Darstellung liegt.

Welche erfolgreichen Attacken werden vom IDS nicht erkannt?

Zur Beantwortung wird ein Attack-Graph erstellt, der zwischen weißen und schwarzen Knoten unterscheidet. Weiße Knoten sind Zustände, in denen sich der Angreifer vom IDS unentdeckt bewegt. Schwarze Knoten werden vom IDS entdeckt und ein Alarm wird ausgelöst, was aber die Zielerreichung des Angreifers nicht verhindert. Weiße Knoten sind somit für den Angreifer vorzuziehen.

Wird ein Netzwerk sicher(er), wenn alle Maßnahmen aus M' implementiert werden?

Diese Frage ist interessant für Administratoren, die durch Maßnahmen ihr Netzwerk sichern wollen.
Zur Analyse müssen alle atomaren Attacken (Verbindungen) zwischen Knoten entfernt werden, die durch die Maßnahmen wegfielen. Ist der Endzustand S_S von S_0 aus nicht mehr zu erreichen, ist das Netzwerk sicher geworden.
Diese Frage kann in linearer Zeit zur Größe des Graphs beantwortet werden.

Welche Maßnahmen müssen ergriffen werden, um das Netzwerk sicher zu machen?

Es gibt eine Menge an Maßnahmen (M), aus denen heraus zu suchen ist, was die kleinste Teilmenge (M') ist, die das Netzwerk sicher macht.
Dies ist ein NP-Vollständiges Problem, das über die Prüfung mittels Frage 2 (s.o.) von jedem Element der Potenzmenge von M gelöst werden kann.
Im Paper wird daher eine Approximationsmethode vorgestellt, die nach eigener Aussage gute Ergebnisse liefern soll und durch zwei Schritte passiert:

1. Finden einer Menge von atomarer Angriffe, deren Entfernung das Netzwerk sicher machen würde.
2. Unter der Voraussetzung, dass eine Menge unter 1. gefunden wurde und Maßnahmen bestehen, welche diese atomaren Angriffe entfernen, kann nun eine minimale Menge von Maßnahmen gesucht werden, die diese Attacken abdeckten.

Das Problem ist bekannt als das Hitting Set-Problem und gehört zu Karps 21-NP vollständigen Problemen. Für dieses Problem wird im Paper ein Greedy-

Algorithmus zur Approximation vorgestellt, welcher nicht Inhalt dieser Zusammenfassung sein soll.

Probabilistische Analyse

Um den Graph weiter zu verbessern oder Maßnahmen zu priorisieren, können durch die o.g. Mittel quantitative Daten erhoben werden, welche die Häufigkeit von Angriffsrouten beziffern. So kann ein Zustand im Graph, der zwei Ausgänge hat, beispielsweise eine Verteilung von 25% zu 75% aufweisen, die als Indikator dienen kann.
Bei der Betrachtung eines Probablilistischen Attack-Graphs (kurz PAG) interessiert laut dem Paper insbesondere die Zuverlässigkeit des Graphs. Konkret bedeutet das, wie wahrscheinlich es ist, dass ein Angreifer in einen Endzustand gelangt.

Rank-Attack

In dem Paper „Ranking Attack Graphs“ wird eine Möglichkeit vorgestellt, den kennengelernten Attack-Graph zu erweitern: Die Zustände des Graph werden in eine Rangfolge der Wichtigkeit gebracht, beispielsweise der Wahrscheinlichkeit, dass der Angreifer diesen Zustand erreicht [Q6; S.3].
Im Anschluss wird dafür u.a. der von Google verwendete Page-Rank-Algorithmus als Mittel zur Bewertung aufgezeigt.

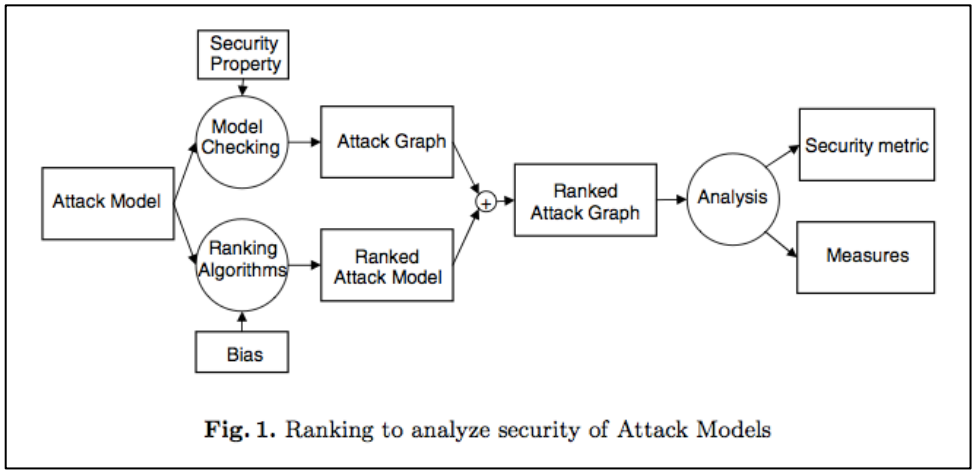


Abb.10: Attack Graph und Ranking Attack Graph

Laut V. Mehta et al. kann der vorgestellte Algorithmus insbesondere durch folgende vier Vorteile punkten [Q6; S.34]:

1. *Komfort und Flexibilität*: Probabilistische Werte müssen nicht zur Gänze vorliegen und Tendenzen können leicht angepasst werden.
2. *Skalierbarkeit*: Die Verwandtschaft zum Page-Rank-Algorithmus erlaubt eine hohe Beschleunigung bei der Berechnung.
3. *Anwendungsbreite*: Wie der Attack-Graph auch, ist der Ranking Attack-Graph ebenso dynamisch anpassbar und anwendbar auf verschiedenste Situationen.
4. *Analysierbarkeit*: Der Algorithmus priorisiert die Zustände mittels der Ränge, sodass sich Administratoren direkt den notwendigsten Schwachstellen widmen können.

Für Sicherheitsanalysten ist der Ranking Attack-Graph ein weiterer Indikator, welche Schwachstellen im Unternehmensnetz am kritischsten sind: Durch Sicherheitsmetriken können Zustände ab gewissen Schwellwerten als sicher eingestuft werden, Maßnahmen lassen sich anhand der Rangfolge priorisieren, laufende Angriffe – wie Vireninfektionen – sind analysierbar und Infektionen lassen sich isolieren. Zudem kann ein Fokus über die Zustände gesetzt werden, um kritische Schwachstellen konzentriert zu behandeln ohne sich von der Anzahl überwältigen zu lassen [Q6; S. 9, 10].

Fazit und Bewertung des Autors

Der Attack-Graph hat sich als Standardmittel im Kampf gegen Angriffe in Unternehmensnetze etabliert. Die Analysemethode ist ein Basisgerüst für die Analyse, bedarf allerdings detaillierter Daten über die Beschaffenheit des Systems. Folglich ist auch die Ergebnisqualität stark abhängig von der Qualität der Grundlagedaten.

Da es kaum möglich ist, sämtliche Angriffspunkte für Hacker abzustellen – seien es kostentechnische Gründe, wie z.B. teurere Serversysteme oder Gründe für Mitarbeiterzufriedenheit (wie Internet-Zugriff) – ist der Attack-Graph ein gutes Mittel, um Angreifern den Zugriff auf kritische Elemente effizient abzuschneiden.

Der Attack-Graph dient zur Identifikation von kritischen Zuständen durch Verkettung mehrerer Angriffe – nicht zur Analyse von Angriffspunkten auf Peripherie.

Um die Ergebnisqualität des Attack-Graphs zu erhöhen, gibt es verschiedene Ansätze. Drei formale Methoden wurden hier grob skizziert: Eine Minimierungsanalyse, die versucht, mit minimalem Aufwand das Netz zu sichern; eine probabilistische Analyse, um wahrscheinliche Wege des Angreifers priorisiert zu sichern und eine Rangfolge-Analyse, die eine Priorisierung der Zustände anhand Sicherheitsmetriken vornimmt.

Quellen

Q1	S. Jha, O. Sheyner and J. Wing Two Formal Analysis of Attack Graphs / In Proceedings Of The 15th Computer Security Foundation Workshop http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.61.2087&rep=rep1&type=pdf
Q2	Lingyu Wang, Sushil Jajodia, and Anoop Singhal Measuring the Overall Security of Network Configurations Using Attack Graphs / Druck nicht bekannt http://users.encs.concordia.ca/~wang/papers/dbsec07.pdf
Q3	Anoop Singhal, Ximming Ou, NIST Interagency Report 7788 Security Risk Analysis of Enterprise Networks Using Probabilistic Attack Graphs / Druck nicht bekannt http://csrc.nist.gov/publications/nistir/ir7788/NISTIR-7788.pdf
Q4	Clarke, Schlingloff Model Checking, in Handbook of Automated Reasoning, Band II, S. 1637–1790, 2001
Q6	Vaibhav Mehta, Constantinos Bartzis, Haifeng Zhu, Edmund Clarke, and Jeannette Wing Ranking Attack Graphs / Proceedings of Recent Advances in Intrusion Detection http://www.cs.cmu.edu/~scenariograph/mehta-wing06.pdf
Q7	Datenveränderung, Computersabotage 2008-2012, Bundeskriminalamt Cyberkriminalität http://www.bka.de/nn_205994/SharedDocs/Downloads/DE/Publikationen/JahresberichteUndLagebilder/Cybercrime/cybercrimeBundeslagebild2012,templateId=raw,property=publicationFile.pdf/cybercrimeBundeslagebild2012.pdf
Q8	Top Causes of Data Breaches 2012, Symantec (inzwischen aktualisiert auf 2013: http://www.auditnorth.co.uk/documents/Symantec%20Website%20Security%20Threat%20Report%202013.pdf)