Komplexe Systeme, heterogene Angreifer und vielfältige Abwehrmechanismen: Simulationsbasierte Entscheidungsunterstützung im IT-Sicherheitsmanagement*

Andreas Ekelhart, Bernhard Grill, Elmar Kiesling
SBA Research
Favoritenstraße 16
1040 Wien, Österreich
{aekelhart,bgrill,ekiesling}@sba-research.org

Christine Strauß
Universität Wien
Oskar-Morgenstern-Platz 1
1090 Wien, Österreich
christine.strauss@univie.ac.at

Christian Stummer
Universität Bielefeld
Universitätsstraße 25
33615 Bielefeld, Deutschland
christian.stummer@uni-bielefeld.de

Abstract: Dieser Beitrag beschreibt einen simulationsbasierten Entscheidungsunterstützungsansatz zur Analyse und Optimierung der Sicherheit komplexer Informationssysteme. Er stützt sich auf die konzeptuelle Modellierung von Sicherheitswissen, die Modellierung des Angreiferverhaltens, die Simulation von Angriffen sowie auf genetische Algorithmen zur Bestimmung effizienter Bündel von Sicherheitsmaßnahmen. Mittels Angriffssimulationen für unterschiedliche interne und externe Angreifertypen können Vertraulichkeits-, Integritäts- und Verfügbarkeitsrisiken erfasst und für die Ermittlung effizienter Kombinationen von Sicherheitsmaßnahmen hinsichtlich mehrerer Risiko-, Kosten- und Nutzen-Ziele genutzt werden. Wir beschreiben den entwickelten Ansatz sowie seine prototypische Implementierung und zeigen die Anwendung anhand von beispielhaften Szenarien.

1 Einführung

Der Schutz komplexer Informationssysteme stellt heute, nicht zuletzt angesichts der dynamischen und sich rasch verändernden Bedrohungslandschaft, eine erhebliche Herausforderung dar. In den vergangenen Jahren war eine deutliche Tendenz zu komplexen und zielgerichteten Angriffen beobachtbar. Im Gegensatz zu Schadsoftware, die automatisiert vordefinierte technische Schwachstellen ausnutzt, richten sich diese Angriffe gegen spezifisch und strategisch gewählte Ziele, die häufig über komplexe Angriffsketten erreicht werden. Die ausschließlich isolierte Betrachtung einzelner technischer Maßnahmen ist daher unzureichend und aus Sicht eines systematischen Risikomanagements wenig zielführend. Viel-

*Ein ähnlicher Beitrag erschien in englischer Sprache [KEG+14].

mehr sollte eine umfassende Betrachtungsweise verfolgt werden um adäquat berücksichtigen zu können, dass Angreifer zur Zielerreichung technische wie auch nicht-technische Mittel einsetzen und kombinieren (z.B. Netzwerk, Software, physische, soziale Angriffe). Der hier vorgestellte Ansatz unterstützt das IT-Sicherheitsmanagement durch eine dynamische Angriffssimulation bei der Bewertung der Sicherheit von Systemen sowie bei der Bestimmung effektiver Kombinationen von Sicherheitsmaßnahmen. Wir betrachten Sicherheit dabei als emergente Eigenschaft komplexer Informationssysteme.

Basierend auf einer Reihe von Formalismen und Methoden (Monte Carlo, ereignisorientierte Simulation, Petrinetze etc.) wurden in der Literatur unterschiedliche Ansätze zur Sicherheitsanalyse mittels simulierter Angriffe vorgestellt [Coh99, CPJL01, DW06, DMCR06]. Diese modellieren jedoch den Angreifer und sein Verhalten in der Regel nicht explizit. Im Kontext der Identifikation möglicher Angriffe setzt unser Ansatz auf dynamische Angriffsgraphen. Im Unterschied zu bestehenden Arbeiten [AWK02, OBM06, SO08] verfolgen wir aber nicht das – in komplexeren Umgebungen häufig nicht erreichbare – Ziel einer vollständigen Enumeration aller Pfade, sondern ermitteln Angriffsgraphen dynamisch im Zuge der Simulation von Angriffen, was es uns auch erlaubt, auf einschränkende Annahmen, wie etwa Monotonie (d.h. dass einmal erlangte Angriffsmöglichkeiten nicht – etwa durch Abstürzen eines Rechners aufgrund eines versuchten Angriffes – wieder verloren werden können), zu verzichten. Weiters beschränken wir uns in unserer Modellierung nicht auf den Aspekt der Netzwerksicherheit, sondern erfassen Wissen über mögliche physische, soziale, und technische Angriffe und deren Kausalzusammenhänge in einer formalen Wissensbasis. Dabei greifen wir auf bestehende Konzepte einer in [FE09] vorgestellten Ontologie zurück. Um die im Simulationsablauf häufigen Abfrage der Wissenbasis performant durchführen zu können erfolgt die Modellierung der Wissensbasis jedoch ähnlich wie in [OBM06] in Form von Prolog-Regeln. In Bezug auf Mehrzieloptimierung ist der vorgestellte Ansatz mit [EFN09] verwandt, hebt sich aber durch den simulationsbasierte Bewertung und die Berücksichtigung mehrstufiger Angriffe ab.

Die zentralen Forschungsbeiträge können wie folgt zusammengefasst werden: Wir modellieren abstrakte Angriffsmuster und entwickeln einen Mechanismus, um daraus konkrete Angriffspfade abzuleiten. Während bestehende Ansätze häufig darauf abzielen die Menge aller möglichen Angriffspfade aufzuzählen betrachten wir die Bestimmung dieser Pfade als einen dynamischen, durch das Verhalten des Angreifers gesteuerten Prozess. Wir modellieren Angreifer daher als Agenten, die sich strategisch verhalten und bewusste Entscheidungen treffen. Schließlich entwickeln wir eine ereignisorientierte Angriffssimulation und nutzen diese, um mit Hilfe genetischer Algorithmen unter mehrfacher Zielsetzung effektive Kombinationen (Portfolios) von Sicherheitsmaßnahmen zu bestimmen. Abbildung 1 stellt die Architektur des simulationsbasierten Optimierungsansatzes dar. Dieser baut auf einer Wissensbasis auf, welche Angriffsmechanismen, Gegenmaßnahmen und das zu schützende System formal abbildet (Kapitel 2). Auf Basis dieser Elemente können Angriffsmuster dynamisch verbunden und Angriffe simuliert werden (Kapitel 3). Die automatisierte Bestimmung effizienter Kombinationen von Sicherheitmaßnahmen erfolgt schließlich durch metaheuristische Optimierungstechniken (Kapitel 4). Entscheidungsträger können auf Basis der Optimierungsergebnisse effiziente Lösungen vergleichen, mehrere zueinander in Konflikt stehende Ziele abwägen und letzlich eine Kombina-

tion von zu implementierenden Sicherheitsmaßnahmen auswählen. Wir verdeutlichen die Anwendung der Methode anhand eines Beispiels (Kapitel 5).

2 Wissensbasis

Eine Wissensbasis, welche die benötigten Konzepte und Zusammenhänge aus der Sicherheitsdomäne erfasst, dient als Grundlage für die in Abschnitt 3 vorgestellte Angriffssimulation. Unsere Implementierung verwendet die deklarative, logische Programmiersprache *Prolog* zur Modellierung des sicherheits- und systemrelevanten Wissens sowie zur Erkennung möglicher Angriffspfade.

2.1 Sicherheitswissen

Das *attack and control model* liefert eine formale Beschreibung, wie Systeme angegriffen und abgesichert werden können. Es beschreibt Angriffsmuster, die sich auf einen bestimmten Kontext beziehen und nur unter bestimmten Voraussetzungen gültig sind. Wir nutzen das bestehende, öffentlich verfügbare CAPEC-Repository[1] (common attack pattern enumeration and classification), welches aktuell 400 Angriffsmuster aus dem Softwarebereich umfasst, die Sicherheitswissen aus Angreifersicht repräsentieren.

Um diese zumeist semi-strukturierte Information für unseren Simulationsansatz nutzen zu können, ist eine Übersetzung in eine formale Darstellung (auf Basis der CAPEC Abschnitte *Summary*, *Experiments*, *Outcomes* und *Attack Prerequisites*) erforderlich. Das folgende Beispiel beschreibt eine *SQL Injection* mit den nötigen Vorbedingungen. Das Muster liefert mittels der hinterlegten Datenbasis alle zulässigen Variablenbelegungen für *Attacker* und *DbServer*.

```
action_sqlInjection(Attacker, DbServer) :-
        technicalSkillLevel(Attacker, TechnicalSkillLevel),
```

[1] http://capec.mitre.org/, letzter Zugriff am 13. Februar 2014.

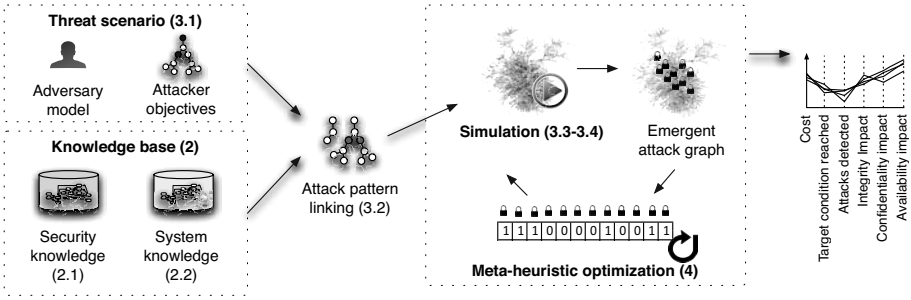

Abbildung 1: Architektur des simulationsbasierten Optimierungsansatzes

```
        TechnicalSkillLevel >= 1,
        connected(AttackHost, WebServer, httpProtocol, httpPort),
        connected(WebServer, DbServer, dbProtocol, dbPort),
        ...
```

Um mögliche Auswirkungen einer Angriffsaktion abzubilden, definieren wir Nachbedingungen, welche die in der Simulation auftretenden Zustandsveränderungen formalisieren. Die erfolgreiche Ausführung einer Angriffsaktion kann dem Angreifer beispielsweise Root-Rechte auf dem Zielserver sichern, während hingegen ein fehlgeschlagener Versuch den attackierten Server offline nehmen könnte. Formal definieren wir beide Typen als Regeln, welche eine Änderung in der Wissensbasis bewirken.

In CAPEC findet sich auch ein Abschnitt *CIA Impact*, welcher die Auswirkungen einer Angriffsaktion hinsichtlich Vertraulichkeit, Integrität und Verfügbarkeit auf einer Skala von *high*, *medium* und *low* bewertet. Obwohl diese Information einen nützlichen Anhaltspunkt hinsichtlich einer Bewertung liefert, wird auf den Kontext, in dem der Angriff erfolgt, keine Rücksicht genommen. Unser simulationsbasierter Ansatz hingegen trägt dem Umstand Rechnung, dass gleiche Angriffsaktionen, je nach Angriffsziel und Umgebung, unterschiedliche Auswirkungen haben können. Die Sicherheitswissensbasis definiert nur die betroffenen Sicherheitsattribute einer Angriffsaktion sowie Regeln, wie das Schadensausmaß erhoben wird. Das folgende Beispiel legt fest, dass sich *SQL Injection* auf das Sicherheitsattribut *Vertraulichkeit* auswirkt. Gelingt es einem Angreifer die Aktion erfolgreich durchzuführen, wird die Auswirkung durch die Vertraulichkeitsbewertung des betroffenen Servers bestimmt.

```
action_impact(action_sqlInjection, confidentiality).
impact_success_sqlInjection(Attacker, TargetHost,  Impact):-
        importance(TargetHost, confidentiality, Impact).
```

Zusätzlich erweitern wir das Sicherheitswissen um Maßnahmendefinitionen aus CAPEC. Die Abschnitte *Solutions*, *Mitigations* und *Relevant security requirements* liefern hierzu die Grundlage.

2.2 Systemwissen

Um den Sicherheitszustand eines Informationssystems bestimmen zu können, müssen dessen Komponenten erhoben und Zusammenhänge modelliert werden. Zu diesem Zweck erfassen wir – getrennt von dem bereits erläutertem abstrakten Angriffswissen – systemspezifisches Wissen. In diesem Modell werden materielle sowie immaterielle *Assets* wie z.B. Hardwarekomponenten, Netzwerkstrukturen, Daten und Mitarbeiter abgebildet. Das folgende Beispiel einer Intrusion Detection Maßnahme zeigt, wie *Assets* modelliert und Maßnahmen zugewiesen werden.

```
host(dbServers_host_1).
installed(dbServers_host_1, ids1).
stores(dbServers_host_1, projectDb1).
inHostGroup(dbServers_host_1, dbServerHosts).
hacl(workstationHosts, dbServerHosts, httpProtocol, httpPort).
```

Zur späteren Bewertung der Auswirkung von Bedrohungen werden modellierten Assets Kritikalitätswerte für Sicherheitsattribute zugewiesen. Diese Werte können aus existierenden Impact-Analysen oder Asset-Kritikalitätsanalysen übernommen werden.

```
criticality(projectDb1, confidentiality,2).
criticality(projectDb1, integrity, 3).
criticality(projectDb1, availability, 2).
```

3 Angriffssimulation

3.1 Bedrohungsprofil

In der Literatur werden Angreifer üblicherweise anhand natürlichsprachlicher Beschreibungen charakterisiert und klassifiziert [MRKS10]. Ein Vorteil unseres formalen Modells besteht darin, dass es die Definition spezifischer Profile unter Berücksichtigung von Fähigkeiten, Ressourcen und Verhaltensattributen des Angreifers ermöglicht. Diese Attribute bestimmen, welche Angriffsmöglichkeiten für einen Angreifer verfügbar sind, das Angriffsverhalten, sowie die Erfolgswahrscheinlichkeiten für Angriffsaktionen.

Die Angriffssimulation nutzt und verbindet das modellierte Wissen für die Ausführung von Angriffsszenarien. Die Entscheidungen der Angreifer für bestimmte Aktionsfolgen, das Ergebnis einzelner Aktionen sowie die Erkennung von Angriffen werden in der Simulation probabilistisch ermittelt. Um Variabilität und Unsicherheit abzubilden, werden für jedes Angriffsszenario mehrere Replikationen mit unterschiedlichen Startwerten des Zufallszahlengenerators ausgeführt.

Bei jedem Durchgang arbeitet die Simulation eine Liste von Ereignissen ab und sammelt Ergebnisvariablen (z.B. impacts auf Sicherheitsattribute), die aggregiert und zu Optimierungszwecken genutzt werden können. *Action Selection* Ereignisse ermitteln durch Verbindung von Angriffsmustern verfügbare Aktionen, wählen auf Basis des Angreifer-Verhaltensmodells Aktionen aus und fügen *Action Start* Ereignisse in den Simulationsablauf ein. Während der Ausführung wird die effektive Dauer dieser Aktionen (entsprechend der Schwierigkeit, Angreiferfähigkeiten und eingesetzten Präventivmaßnahmen) bestimmt und ein *Action End* Ereignis in den Simulationsablauf eingefügt. *Detection* werden werden ausgelöst, sobald Assets angegriffen werden, die mit detektierenden Maßnahmen versehen sind; anschließend kann ggf. ein *Attacker Stopped* Ereignis ausgelöst werden. Schließlich beendet ein *Target Condition Reached* Ereignis die Simulation, falls der Angreifer sein Ziel erreicht hat.

3.2 Verkettung von Angriffsmustern

Angriffsmuster werden automatisch aufgrund gemeinsamer Vor- und Nachbedingungen verbunden. Dieser Ansatz ist mit Angriffsgraph-Konzepten [Sch00] verwandt, insbeson-

dere mit fortgeschrittenen Formalismen und Methoden zur Generierung großer Bäume [AWK02, SHJ+02, SO08] sowie mit Erweiterungen, die Sicherheitsmaßnahmen einbeziehen [BFP06]. Allerdings beruhen diese Ansätze darauf, *alle* möglichen Angriffspfade zu identifizieren, um einen vollständigen Angriffsgraph zu erhalten. Dies ist aufgrund der exponentiellen Größe des Suchraums rechnerisch problematisch. Unser Ansatz begreift die Suche nach möglichen Angriffspfaden hingegen als dynamischen Prozess und ist daher in vieler Hinsicht allgemeiner und flexibler.

3.3 Auswahl und Ausführung von Aktionen

Wir implementieren Angreifer mit einem Verhaltensmodell, das Aktionen iterativ aufgrund von (i) individuellen Charakteristika, (ii) dem allgemeinen Wissen des Angreifers über mögliche Angriffspfade und (iii) dem Ergebnis vorausgegangener Angriffsaktionen auswählt. Dieses Modell berücksichtigt, dass Angreifer (in unterschiedlichem Ausmaß) über allgemeines Sicherheitswissen verfügen, aber typischerweise nur unvollständige Informationen über das angegriffene System haben. Das allgemeine Wissen über mögliche Angriffspfade wird durch einen abstrakten Angriffsgraph repräsentiert, der aus der Wissensbasis für einen bestimmten Angreifer und ein betrachtetes Ziel abgeleitet werden kann. Die Simulation berücksichtigt die gültigen Zuordnungen für alle Vorbedingungen einer abstrakten Aktion im Kontext eines modellierten Systems und generiert daraus eine Menge konkreter Instanzen von Angriffsaktionen, die gegen bestimmte Assets ausgeführt werden können.

Der im Anhang dargestellte Verhaltensalgorithmus beruht auf der Annahme, dass Angreifer wechselweise einer gewählten Strategie folgen (d.h. eine Sequenz von logisch zusammengehörigen Aktionen ausführen) und alternative Ansätze versuchen (d.h. Startpunkte für einen neuerlichen Angriffsversuch wählen). Wir unterstellen ferner, dass Angreifer den erwarteten Aufwand minimieren und daher Angriffe im abstrakten Graph möglichst "nahe" am Zielzustand beginnen. Diese Annahme wird in der Funktion $choose(\overline{A})$ umgesetzt, welche die relativen "Distanzen" (Anzahl von Schritten in Bezug auf die längste Distanz) zwischen einer abstrakten Aktion a und der Zielbedingung t im abstrakten Angriffsgraph für alle möglichen Aktionen berechnet. Schließlich ermittelt die Funktion Gewichtungen für jede mögliche konkrete Aktion $\overline{a}$ auf Basis der Erfolgs- und Detektionswahrscheinlichkeiten sowie der ermittelten relativen Distanzen zur Zielbedingung und der Präferenzen des Angreifers. Die Funktion $weightedChoice(\overline{A}, W)$ bestimmt anschließend die gewählte konkrete Aktion mit entsprechenden Wahrscheinlichkeiten.

Nachdem die erste Aktion ausgewählt wurde, sind alle weiteren Entscheidungen vom Ergebnis der zuvor gewählten Aktion p abhängig. Die Verhaltensparameter $p_{continueNew}$, $p_{alternativesNoNew}$, p_{retry}, $p_{AlternativesFailed}$ bestimmen dabei abhängig vom Ergebnis, ob der Angreifer (i) eine neu verfügbar gewordene Aktion, (ii) eine alternative Aktion auf gleicher Ebene (d.h., eine Aktion, die als Konsequenz der gleichen vorhergehenden Aktion verfügbar wurde) oder (iii) einen neuen Ansatz versucht und aus allen verfügbaren Aktionen wählt.

Nach Ausführung einer Aktion wird bestimmt, (i) ob die Angriffsaktion erfolgreich ist, (ii) ob die Zielbedingung erreicht wird, (iii) welche Aktionen als Konsequenz nicht mehr verfügbar sind, (iv) welche neuen Aktionen verfügbar werden und (v) welche Auswirkungen auf Sicherheitsattribute entstehen.

3.4 Auswirkung detektierender Maßnahmen

Für detektierende Maßnahmen können zwei mögliche Konsequenzen definiert werden, nämlich (i) *stop*, d.h. Simulationslauf beenden, oder (ii) einen Verweis auf einen definierten Nachbedingungspfad im abstrakten Graph, der ausgeführt werden soll. Letzteres kann Zustände im Systemmodell verändern. Wenn beispielsweise ein Intrusion Detection System die Quell-IP-Adresse des Angreifers blockiert, so verhindert das bestimmte Angriffe, ermöglicht aber potentiell auch neue Angriffsmuster (z. B. wenn dies genutzt werden kann, um legitime Zugriffe zu unterbinden).

4 Optimierung

Entscheidungsträger können durch Modellierung ihrer Systeme, Einsatz von Sicherheitsmaßnahmen und Simulation von Angriffen wertvolle Erkenntnisse über die Sicherheit eines Systems gewinnen. Manuelles Experimentieren mit Maßnahmen, bis zufriedenstellende Simulationsergebnisse erzielt werden, ist jedoch ein aufwändiger Prozess, der bei komplexeren Systemen praktisch nicht sinnvoll durchführbar ist.

Wir führen daher das Konzept von Sicherheitsmaßnahmen-Portfolios ein, die durch einen Genotyp beschrieben und automatisch optimiert werden. Jeder Genotyp setzt sich aus binären Variablen zusammen, die jeweils eine Maßnahmen-Asset-Kombination repräsentieren (z. B. *logPolicy1* implementiert auf *dbServerHosts*) und den Wert 1 annehmen, wenn die entsprechende Maßnahme auf das asset angewendet wird.

Zur Bewertung eines Maßnahmenportfolio wird das System zunächst entsprechend dem Genotyp initialisiert, anschließend werden eine Reihe von Angriffen mit unterschiedlichen Zufallsgenerator-Startwerten simuliert und die Ergebnisse aggregiert. Da wir Mehrfachzielsetzungen erlauben, wird in der Regel kein eindeutig “bestes” Maßnahmen-Portfolio gefunden, sondern eine Reihe von alternativen effizienten Lösungen. Jedes in dieser Menge enthaltene Portfolio ist nicht-dominiert, d.h. die Lösung enthält kein anderes Portfolio, das zumindest gleich gute Werte für alle Ziele und einen strikt besseren Wert für zumindest eines der Ziele aufweist.

Aufgrund des großen Entscheidungsraums und der “teuren” simulationsbasierten Evaluierungsprozedur handelt es sich um ein rechnerisch sehr aufwändiges Optimierungsproblem. Exakte Lösungen können daher nur für kleine Probleminstanzen durch vollständige Enumeration ermittelt werden. Für praxisrelevante Problemgrößen setzen wir daher genetische Algorithmen ein. Diese von natürlichen Evolutionsprozessen inspirierten Ansätze lösen Optimierungsprobleme implizit, indem sie eine Population von Individuen über mehrere

Generationen hinweg durch Bewertung ihrer „Fitness", Auswahl von „fitten" Individuen und Anwendung von Kreuzungs- und Mutationsoperatoren auf die ausgewählten Individuen entwickeln.

Konkret haben wir mit den Algorithmen NSGA-II [DPATM00] und SPEA-II [ZLT02] experimentiert und festgestellt, dass beide ein für unsere Zwecke zufriedenstellendes Leistungsverhalten zeigen. Für unser im nächsten Abschnitt vorgestelltes illustratives Beispiel haben wir Optimierungsläufe mit NSGA-II über 500 Generationen durchgeführt und typischerweise eine Konvergenz innerhalb der ersten 250 Generationen festgestellt.

5 Anwendungsbeispiel

5.1 Versuchsaufbau

Der Prototyp der Simulations- und Optimierungskomponenten wurden in Java implementiert. Die Simulation nutzt die Scheduling-Engine der Mason-Bibliothek [LCRPS04], die Wissensbasis wurde in SWI-Prolog [WSTL12] modelliert und der Zugriff darauf erfolgt über JPL [SDW]; die Optimierung mittels genetischer Algorithmen wurde mit Hilfe des Opt4j-Frameworks implementiert [LGRT11].

Für das Beispielszenario wurden 10 CAPEC-Angriffsmuster modelliert und um zusätzliche Muster ergänzt. Die im Anhang enthaltene Tabelle 3 fasst die modellierten Angriffsmuster zusammen. Außerdem wurden Abwehrmechanismen, wie etwa Antivirensoftware, Patches, Intrusion Detection-Systeme, Logging-Policies und Sicherheitstrainings, definiert. Die ebenfalls im Anhang enthaltene Tabelle 4 liefert eine Übersicht der modellierten Maßnahmen und deren Wirksamkeit in Bezug auf unterschiedliche Angriffe.

Schließlich wurde eine Organisation und deren IT-Infrastruktur durch Instanziierung von abstrakten Konzepten wie *host*, *subnet*, *data*, *user* etc. und deren Beziehungen (z.B. host *stores* data, host *uses* software) modelliert. Das Beispielszenario mit 30 Hosts, 5 Webservern, 5 Datenbankservern, 30 Angestellten und 3 Administratoren wurde automatisch von einem Generator erzeugt. Die im Anhang enthaltene Abbildung 4 stellt das modellierte System vereinfacht dar.

Wir simulieren fünf interne und externe Typen von Angreifern, die alle das Ziel verfolgen, Zugriff auf die Datenbank *db2* zu erlangen. Interne Angreifer verfügen bereits über (eingeschränkten) Zugang zum System; für externe Angreifer stellt die demilitarisierte Zone (DMZ) den zentralen Angriffspunkt dar. Ferner wird jeder Angreifer durch individuelle verhaltensbestimmende Attribute charakterisiert (siehe Tabellen 1, 2 und 3 im Anhang).

Für unsere Experimente wurden sechs Optimierungsziele definiert, nämlich Minimierung der Kosten, Minimierung erfolgreicher Angriffe, Maximierung der Entdeckung von Angriffen, und Minimierung des „Impacts" auf CIA-Attribute (Vertraulichkeit, Integrität, Verfügbarkeit) des Systems. Um die Beeinträchtigung der CIA-Attribute zu erfassen, wurden die Ausprägungen *low*, *medium* und *high* anhand einer lexikografischen Skala bemessen, d.h. lediglich die Anzahl der Ereignisse in der höchsten auftretenden Katego-

rie ist jeweils relevant. Insgesamt wurden 500 Generationen mit jeweils 30 Simulationsdurchgängen je Portfolio evaluiert.

5.2 Ergebnisse

Alle Berechnungen wurden auf einem 2x3GHz Xeon Prozessor mit einer Laufzeit zwischen 90 Minuten (*Admin*) und 50 Stunden (*Advanced Persistent Threat*) je Szenario durchgeführt. Der genetische Algorithmus identifizierte 251 effiziente Portfolios für das APT-Szenario, 306 für den versierten externen Angreifer, 104 für den unerfahrenen externen Angreifer, 58 für den Angestellten und 2 für den Administrator. Abbildung 2 zeigt die Zielwerte der effizienten Portfolios in einer Parallelkoordinaten-Darstellung. Die Y-Achse ist für die Impact-Attribute in die drei Kategorien *high* (oberhalb der roten Marke), *medium* (oberhalb der orangen Marke) und *low* (oberhalb der grünen Marke) untergliedert. Es wird jeweils nur die höchste Kategorie je Portfolio dargestellt.

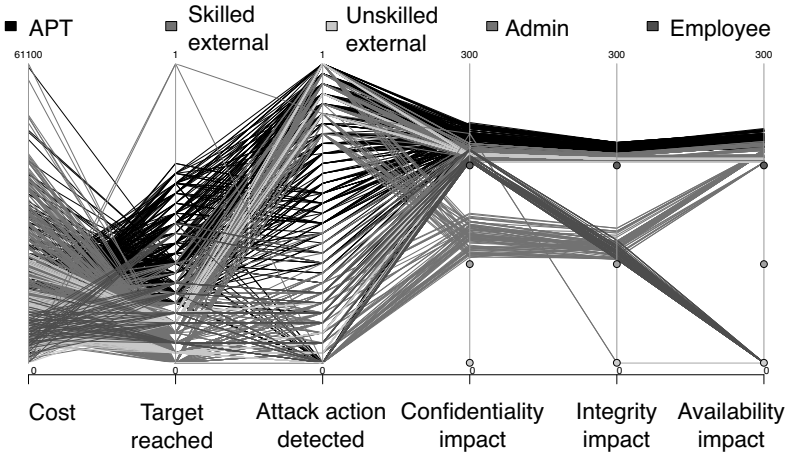

Abbildung 2: Zielwerte der effizienten Portfolios

Die im Anhang enthaltene Abbildung 5 gibt die Lösungsalternativen in einer Heatmap-Darstellung wieder. Die gefundenen effizienten Lösungen werden dabei zeilenweise dargestellt; der Genotypstring wird auf der linken Seite (blau) visualisiert und die Optimierungszielwerte für das Portfolio befinden sich rechts (rot). Jedes blaue Feld repräsentiert eine aktive Zuweisung einer Sicherheitsmaßnahme zu einem Asset. Die roten Ergebnisspalten für den Impact untergliedern sich in die Werte *low*, *medium* und *high*.

Die Ergebnisse dieses Beispiels zeigen, dass der Administrator als Angreifer das Ziel immer erreicht, da er bereits über die erforderlichen Rechte verfügt; eine Log-Policy auf *dbServerHost* erhöht aber zumindest die Entdeckungsrate. Ohne eingesetzte Sicherheitsmaßnahmen erreicht ein Angestellter das Ziel in etwa der Hälfte der Fälle. Die relativ hohe Erfolgsrate kann auf die Verfügbarkeit von effektiven sozialen Angriffsmöglichkeiten

zurückgeführt werden. Eine hohe Erfolgsrate von etwa zwei Drittel weist erwartungsgemäß der Angreifertyp *Advanced Persistent Threat* auf. Maßnahmenportfolios können den Anteil der erfolgreichen Attacken auf minimal 20% reduzieren bzw. die Entdeckungsrate bis auf 63% steigern. Der versierte externe Angreifer erreicht ähnliche Ergebnisse, wobei die Erfolgswahrscheinlichkeit etwas geringer ausfällt. Der durch einen unerfahrenen Angreifer verursachte Schaden ist weit weniger drastisch, und die Rate der erfolgreichen Angriffe kann mit geeigneten Sicherheitsmaßnahmen auf 3% reduziert werden. Gegen unerfahrene Angreifer zeigen Intrusion-Detection-Systeme eine sehr gute Wirkung. Bei erfahrenen Angreifern sind technische Kontrollen besonders oft in den Lösungsportfolios zu finden.

6 Zusammenfassung

Das vorgestellte Entscheidungsunterstützungssystem kann in komplexen Informationssystemen unter Bedachtnahme auf mehrfache Zielsetzung zur Erhöhung der IT-Sicherheit beitragen. Es greift auf eine Wissensbasis zurück, welche zu mehrstufigen Angriffen verkettbare Angriffsmuster modelliert und die Ermittlung möglicher Angriffspfade für unterschiedliche Angreifertypen ermöglicht. Darauf aufbauend kann eine Vielzahl von Angriffen für unterschiedliche Angreifertypen simuliert werden, um Systemkonfigurationen zu finden, die sich gegen diese Angreifer möglichst resistent zeigen. Die Ergebnisse unseres Experiments mit einer prototypischen Implementierung illustrieren den Nutzen für das IT-Sicherheitsmanagement.

Zukünftige Forschungsarbeit – aufbauend auf dem hier vorgestellten Ansatz – kann in mehrere Richtungen erfolgen. Die Entwicklung einer umfangreichen Sicherheitswissensbasis durch Erfassung weiterer Angriffsmuster ist ein naheliegender nächster Schritt. Diese Aufgabe ist keinesfalls trivial, erfordert sie doch die Entwicklung eines entsprechenden Vokabulars zur Spezifikation von Ursachen und Wirkungen über mehrere Abstraktionsebenen hinweg. Der Aufwand zur Abbildung formaler Angriffsmuster hängt von dem gewählten Detaillierungsgrad, sowie der Expertise des Modellierers ab. In diesem Beitrag wurde CAPEC zur Erstellung formaler Angriffsmuster herangezogen, es können aber natürlich auch andere Standards als Vorlage dienen. Entsprechend der Ausrichtung der Sicherheitsanalyse sollten technische, physische, und soziale Angriffsmuster einbezogen werden. Ferner wäre es interessant, unseren Ansatz in weiteren Szenarien mit strikten Sicherheitsanforderungen, wie beispielsweise im Bereich kritischer Infrastrukturen, zu testen. Schließlich könnte die Wissensbasis öffentlich zugänglich und von einer Community in einem gemeinsamen Repository weiterentwickelt werden. Organisationen müssten dann lediglich ihre eigene IT-Infrastruktur modellieren und könnten die gemeinsame Wissensbasis zur Optimierung ihres Systems nutzen. Da eine manuelle Modellierung und laufende Aktualisierung der IT-Infrastruktur sowohl fehleranfällig, als auch mit hohem Aufwand verbunden ist, sollten Ansätze und Werkzeuge zur automatisierten Inventarisierung betrachtet werden. Durch regelmäßige Updates könnten sie dieses außerdem laufend hinsichtlich der Widerstandsfähigkeit gegen neue Angriffstechniken testen, was den Informationsvorsprung zwischen Angreifer und Organisationen verringern würde.

ACKNOWLEDGMENTS

Die präsentierte Arbeit wurde im Rahmen des Forschungsprojektes "Moses3" vom Fonds zur Förderung der wissenschaftlichen Forschung (Austrian Science Fund FWF: P23122-N23) finanziert und bei Secure Business Austria, einem von der Österreichischen Forschungsförderungsgesellschaft FFG im Rahmen des COMET-Programm unterstützten K1 Kompetenzzentrums, umgesetzt.

Literatur

[AWK02] Paul Ammann, Duminda Wijesekera und Saket Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, Seiten 217–224. ACM, 2002.

[BFP06] Stefano Bistarelli, Fabio Fioravanti und Pamela Peretti. Defense trees for economic evaluation of security investments. In *Proceedings of the First International Conference on Availability, Reliability and Security (ARES 2006)*, Seiten 416–423. IEEE Computer Society, 2006.

[Coh99] Fred Cohen. Simulating cyber attacks, defences, and consequences. *Computers & Security*, 18(6):479–518, 1999.

[CPJL01] Sung-Do Chi, Jong Sou Park, Ki-Chan Jung und Jang-Se Lee. Network security modeling and cyber attack simulation methodology. In *Proceedings of 6th Australasian Conference (ACISP 2001)*, 2001.

[DMCR06] G. C. Dalton, R. F. Mills, J. M. Colombi und R. A. Raines. Analyzing attack trees using generalized stochastic petri nets. In *IEEE Information Assurance Workshop*, Seiten 116–123, 2006.

[DPATM00] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal und T. T. Meyarivan. A fast elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2000.

[DW06] Ole Martin Dahl und Stephen D. Wolthusen. Modeling and execution of complex attack scenarios using interval timed colored Petri nets. In *Proceedings of the Fourth IEEE International Workshop on Information Assurance*, 2006.

[EFN09] Andreas Ekelhart, Stefan Fenz und Thomas Neubauer. AURUM - A framework for information security risk management. In *Proceedings of the 42nd Hawaii International Conference on System Sciences (HICSS)*, Seiten 1–10, Hawaii, USA, 2009.

[FE09] Stefan Fenz und Andreas Ekelhart. Formalizing Information Security Knowledge. In *Proceedings of the 4th ACM Symposium on Information, Computer, and Communications Security*, Seiten 183–194. ACM, 2009.

[KEG+14] Elmar Kiesling, Andreas Ekelhart, Bernhard Grill, Christine Strauss und Christian Stummer. Evolving secure information systems through attack simulation. In *Proceedings of the 2014 Hawaii International Conference on System Sciences (HICSS-47)*, Seiten 4868–4877. IEEE Computer Society, 2014.

[LCRPS04] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait und Keith Sullivan. MASON: A new multi-agent simulation toolkit. In *2004 SwarmFest Workshop*, 2004.

[LGRT11] Martin Lukasiewycz, Michael Glaß, Felix Reimann und Jürgen Teich. Opt4J: A modular framework for meta-heuristic optimization. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, Seiten 1723–1730. ACM, 2011.

[MN98] Makoto Matsumoto und Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, 1998.

[MRKS10] Carol Muehrcke, Elizabeth Ruitenbeek, Ken Keefe und William Sanders. Characterizing the behavior of cyber adversaries: The means, motive, and opportunity of cyber-attacks. In *40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, Chicago, 2010.

[OBM06] Xinming Ou, Wayne F. Boyer und Miles A. McQueen. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, Seiten 336–345. ACM, 2006.

[Sch00] Bruce Schneier. *Secrets & Lies: Digital Security in a Networked World*. Wiley, New York, NY, 2000.

[SDW] Paul Singleton, Fred Dushin und Jan Wielemaker. JPL: A bidirectional Prolog/Java interface. Letzter Zugriff am 12. Februar 2014. `http://www.swi-prolog.org/packages/jpl/`.

[SHJ+02] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann und Jeanette M. Wing. Automated generation and analysis of attack graphs. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, Seiten 273– 284. IEEE, 2002.

[SO08] Reginald E. Sawilla und Xinming Ou. Identifying critical attack assets in dependency attack graphs. In *Proceedings of the 13th European Symposium on Research in Computer Security: Computer Security*, ESORICS '08, Seiten 18–34. Springer, 2008.

[WSTL12] Jan Wielemaker, Tom Schrijvers, Markus Triska und Torbjörn Lager. SWI-Prolog. *Theory and Practice of Logic Programming*, 12(Special Issue 1-2):67–96, 2012.

[ZLT02] Eckart Zitzler, Marco Laumanns und Lothar Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. In K. Giannakoglou, D. Tsahalis, K. Papailiou und T. Fogarty, Hrsg., *Evolutionary Methods for Design, Optimisation and Control*. International Center for Numerical Methods in Engineering, 2002.

A Simulationsablauf

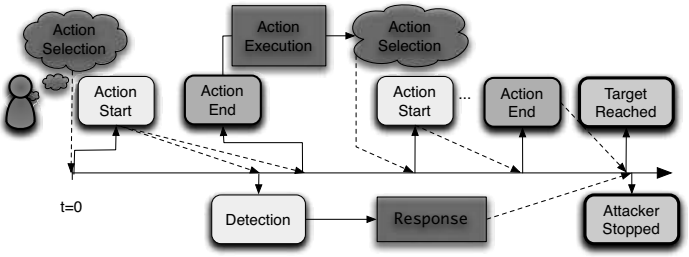

Abbildung 3: Ereignistypen und Simulationsablauf

B Angreiferprofile

Angreifertyp	Zeitlimit (Sek.)	w_{suc}	w_{det}	w_{dist}	Zugriff
Angestellter	150.000	0,45	0,25	0,30	interner Host
Administrator	300.000	0,50	0,20	0,30	alle Rechner
Versierter Externer	200.000	0,30	0,40	0,30	extern
Unerfahrener Externer	100.000	0,50	0,20	0,30	extern
Advanced Persistent Threat	1.000.000	0,50	0,20	0,30	extern

Einheitlich: $pContinueNew = 0,9$; $pRetry = 0,5$;
$pAlternativesNoNew = 0,7$; $pAlternativesFailed = 0,3$

Tabelle 1: Angreifertypen und Attribute

Angreifertyp	Verfügbare Angriffe
Angestellter (Skill-Level: 0)	shoulder surfing
Unerfahrener Externer (Skill-Level: 1)	spearfish attack
	sql injection
	social attack
	brute force
	email keylogger
	email backdoor
Versierter Externer (Skill-Level: 2)	(gleich wie Unerfahrener Externer)
	+ buffer overflow
	+ directory traversal
Administrator (Skill-Level: 2)	(gleich wie Versierter Externer)
Advanced Persistent Threat (Skill-Level: 3)	(gleich wie Versierter Externer)
	+ zero day

Tabelle 2: Verfügbare Angriffe

C Verhaltensmodell

Algorithm 1 Attacker behavior model

Input: available actions $\overline{A}$, previous action p, new actions $\overline{N}$
Output: selected action

```
1:  procedure SELECTACTION(Ā, p, N̄)
2:    for a ∈ Ā do
3:      if FULLFILLSTARGETCOND(ā) then
4:        return ā
5:      end if
6:    end for
7:    if p = null then
8:      CHOICE(Ā)
9:    else if WASSUCCESSFUL(p̄) then
10:     if |N| > 0 then
11:       if p_continueNew > RANDOM() then
12:         return CHOOSE(N̄)
13:       else
14:         return CHOOSE(Ā)
15:       end if
16:     else if p_altNoNew > RANDOM() then
17:       S̄ ← GETALTERNATIVES(p)
18:       return CHOOSE(S̄)
19:     else
20:       return CHOOSE(Ā)
21:     end if
22:   else if p_retry > RANDOM() then return p
23:   else if p_altFailed > RANDOM() then
24:     S̄ ← GETALTERNATIVES(p)
25:     return CHOOSE(S̄)
26:   else
27:     return CHOOSE(Ā)
28:   end if
29: end procedure
```

```
1: procedure CHOOSE(Ā)
2:   for ā ∈ Ā do
3:     d_ā^rel ← d(a,t) / (max(d(a,t))+1)
4:     W_ā ← p_success(ā)^w_success (1 − p_detection(ā))^w_detection (1 − d_ā^rel)^w_distance
5:   end for
6:   return WEIGHTEDCHOICE(Ā, W)
7: end procedure
```

D Angriffsmuster

Angriff	CAPEC (ID)
sql injection	SQL-Injektion (66)
social attack	Informationsgewinnung durch Social-Engineering (410)
brute force	Passwort Brute-Force Attacke (49)
spearfish attack	Informationsgewinnung durch gezielten persönlichen Angriff (407)
buffer overflow	Bufferüberlauf in API-Aufruf (8)
email backdoor	Email-Injektion (134)
zero day	Rechteausweitung (233)
directory traversal	Directory Traversal (213)
shoulder surfing	Informationsgewinnung über soziale Komponenten (404)
access data	legitimer Zugriff
access host	legitimer Zugriff

Tabelle 3: Modellierte Angriffsmuster

E Systemmodell

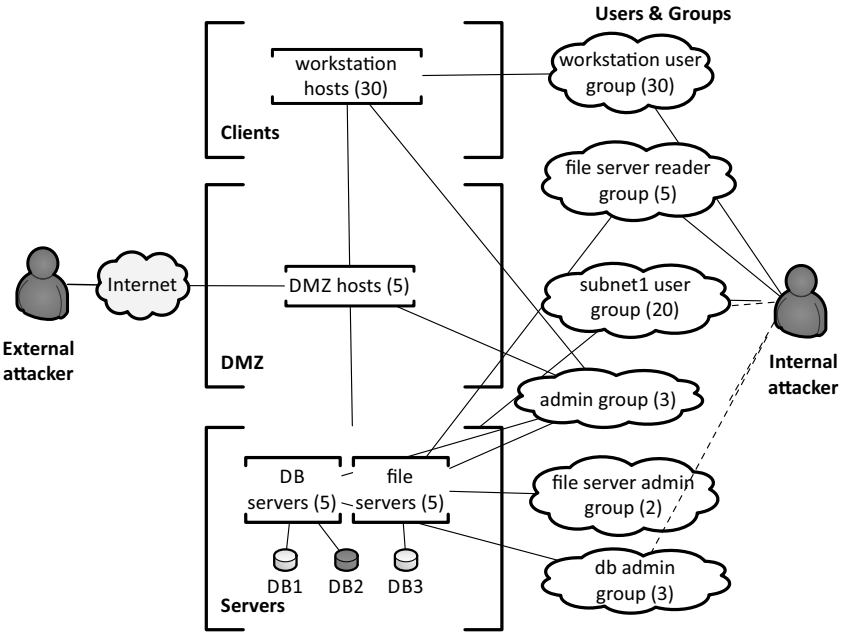

Abbildung 4: Systemmodell des Beispielszenarios

F Maßnahmen

Kategorie	Ausprägung	Angriff	Effektivität	Kosten
	AV1	buffer overflow	0.94	900
	AV1	email backdoor	0.70	900
	AV1	email keylogger	0.70	900
AV	AV1	zero day	0.05	900
	AV2	buffer overflow	0.98	1500
	AV2	email backdoor	0.85	1500
	AV2	email keylogger	0.85	1500
	AV2	zero day	0.08	1500
Code Review	Review1	sql injection	0.80	3200
Server Hardening	Hardening1	directory traversal	0.80	3000
	IDS1	buffer overflow	0.75	7500
	IDS1	sql injection	0.80	7500
IDS	IDS1	zero day	0.20	7500
	IDS2	buffer overflow	0.80	10000
	IDS2	sql injection	0.90	10000
	IDS2	zero day	0.25	10000
Log	Log	access data	0.80	2200
Patch	Patch	buffer overflow	0.98	400
	Train 1	email backdoor	0.30	1200
	Train 1	email keylogger	0.30	1200
	Train 1	shoulder surfing	0.30	1200
	Train 1	social attack	0.40	1200
	Train 1	spearfish attack	0.30	1200
	Train 2	email backdoor	0.70	1800
	Train 2	shoulder surfing	0.50	1800
Security Training	Train 2	email keylogger	0.70	1800
	Train 2	social attack	0.75	1800
	Train 2	spearfish attack	0.70	1800
	Train 3	email backdoor	0.80	4600
	Train 3	email keylogger	0.75	4600
	Train 3	shoulder surfing	0.75	4600
	Train 3	social attack	0.85	4600
	Train 3	spearfish attack	0.75	4600

Tabelle 4: Maßnahmen

G Simulationsergebnisse

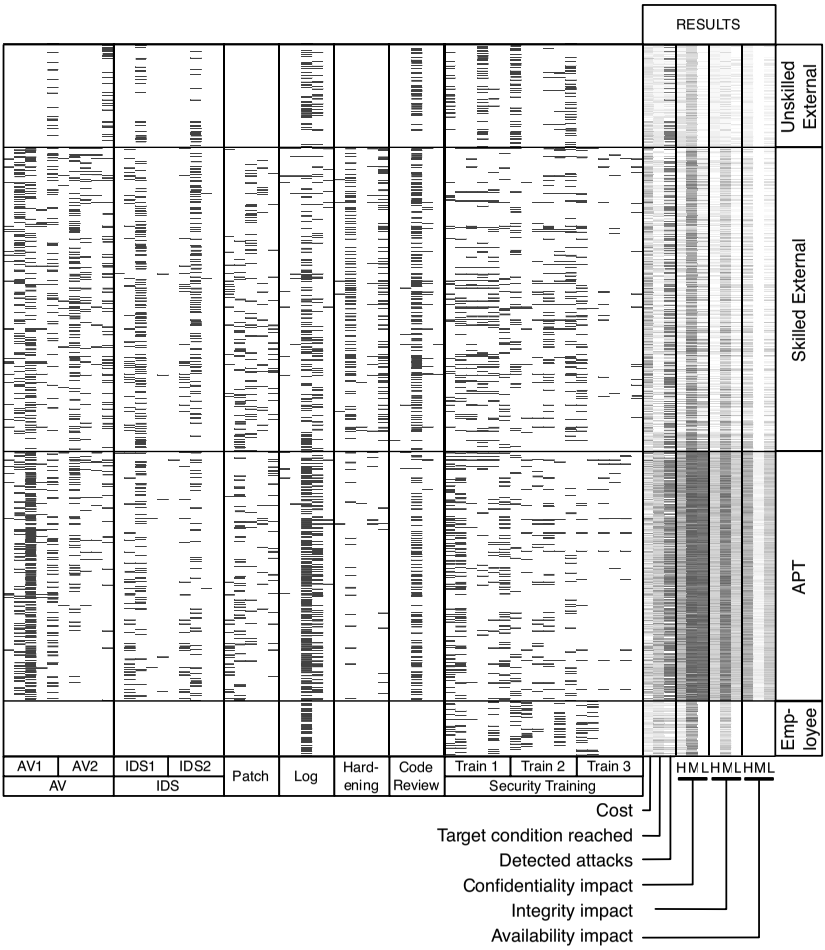

Abbildung 5: Eingesetzte Maßnahmen und Simulationsergebnisse