IT-Sicherheit durch konsequente Aggregation von Analysewerkzeugen

Thomas Schwenkler*
Thomas.Schwenkler@iese.fhg.de

Stephan Groß†
Stephan.Gross@inf.tu-dresden.de

Kai Simon*
Kai.Simon@iese.fhg.de

Abstract: Bis heute wird die Sicherheit bei der Entwicklung von IT-Systemen eher stiefmütterlich behandelt. Die Hauptursache hierfür liegt in extrem kurzen Innovationszyklen: Unternehmen sind darauf angewiesen, einen neuen Markt so schnell wie möglich zu erschließen, um Konkurrenten den Markteintritt zu erschweren. Ein weiteres Problem für die Sicherheit heutiger IT-Systeme sind Fehler im Designprozess, bei der Entwicklung und Implementierung. Mit der Verwendung verfügbarer Analysewerkzeuge wird die Sicherheitsproblematik nicht ausreichend verbessert, sondern lediglich verlagert. Nach einem Überblick über vorhandene Werkzeuge und deren Eigenschaften beschreiben wir, wie sich durch die Aggregation unterschiedlicher Werkzeugtypen die Systemsicherheit verbessern lässt. Darüber hinaus beschreiben wir Ideen zum Entwurf von Synthesewerkzeugen, um zukünftig durch Fehlervermeidung bereits bei der Konfiguration und durch den Einsatz von Security-Engineering-Methoden bei der Implementierung die Systemsicherheit weiter zu erhöhen und damit heutige Analysewerkzeuge ganz oder zumindest teilweise abzulösen.

1 Sicherheitsproblematik heutiger IT-Systeme

Bis heute wird die Sicherheit bei der Entwicklung von IT-Systemen eher stiefmütterlich behandelt. Die Hauptursache hierfür liegt in den extrem kurzen Innovationszyklen der IT-Branche. Dies führt zu Effekten, die sich beispielsweise mit Hilfe von Konzepten aus der Mikroökonomie erklären lassen [And01]. Demnach sind Firmen darauf angewiesen, einen neuen Markt so schnell wie möglich zu erschließen, da jeder etablierte Konkurrent den Markteintritt erschwert, wenn nicht sogar unmöglich macht. Aus diesem Grund müssen vorhandene Systeme oftmals neue Aufgaben erfüllen, deren Anforderungen sie nicht oder nur teilweise gewachsen sind.

Ein weiteres Problem für die Sicherheit heutiger IT-Systeme sind Fehler bei Design und Entwicklung. Programmierfehler sind beispielsweise die Ursache für so genannte Buffer-Overflow-Angriffe. Quasi täglich werden neue Einbruchsmöglichkeiten dieser Art bekannt. Auch zeichnet sich ein Trend zu immer versierteren Angriffswerkzeugen ab, die

*Fraunhofer Institut Experimentelles Software Engineering, Sauerwiesen 6, 67661 Kaiserslautern
†Technische Universität Dresden, Fakultät Informatik, Institut für Systemarchitektur, 01062 Dresden

es selbst Laien ermöglichen, auf einfache Weise Systeme zu schädigen [CER02]. Administratoren sind daher gut beraten, ihre Systeme ständig auf dem aktuellsten Stand zu halten. Durch die hohe Komplexität ist das Einspielen neuer Security Patches jedoch ein äußerst fehlerträchtiger Prozess. So kann beispielsweise durch eine neue Programmversion die bestehende Konfiguration verändert werden. Ferner können durch Programm-Updates bisher unbekannte Wechselwirkungen mit anderen Systemteilen auftreten. Dies erfordert umfangreiche Verträglichkeitstests durch den Administrator.

Darüber hinaus gab es in der Vergangenheit auch immer wieder Fälle, in denen das Systemdesign selbst fehlerhaft war. Die relativ junge Disziplin Security Engineering soll dem entgegenwirken [Eck03]. Basierend auf dem aus dem Software Engineering bekannten Phasenmodell [Bal98] werden dazu Maßnahmen und Methoden definiert, um sicherheitsrelevante Systemeigenschaften zu erfassen, den konkreten Schutzbedarf für ein System zu ermitteln sowie adäquate Sicherheitsstrategien und -architekturen für den Schutz des Systems zu erstellen und umzusetzen.

Mit der Verwendung von Analysewerkzeugen wird die oben beschriebene Problematik jedoch nur verlagert. Statt einer Vielzahl komplexer Systeme zu warten, müssen jetzt mehrere Analysewerkzeuge konfiguriert werden. Deren Funktionsumfang ist typischerweise eng umgrenzt, sodass für spezielle Aufgaben jeweils eigene Werkzeuge eingesetzt werden müssen. Darüber hinaus fehlt eine einheitliche Bedienerschnittstelle; weder die Konfiguration noch die Ergebnisausgabe sind standardisiert. Es ist also weiter ein detailliertes Fachwissen notwendig, über das in der Regel nur ausgewiesene Experten verfügen.

Im folgenden beschreiben wir, wie sich diese Problematik beheben oder zumindest entschärfen lässt. In Kapitel 2 führen wir zunächst zwei Klassifikationen für Werkzeuge zur Sicherheitsanalyse ein und geben dann einen Überblick über solche Programme für den Unix- und Netzwerkbereich. In Kapitel 3 beschreiben wir, wie sich die Aussagekraft und Nutzbarkeit aktueller Analysewerkzeuge verbessern lässt. Grundidee unserer Überlegungen ist die Steigerung des Nutzens von Analysewerkzeugen durch die Vereinheitlichung von Ein- und Ausgabeschnittstellen und langfristig durch die Aggregation der Werkzeuge. Dabei greifen wir auf Erfahrungen zurück, die wir im Rahmen mehrerer Sicherheitsaudits als externe Revisoren gemacht haben [SG03]. In Kapitel 4 schließlich präsentieren wir einen Lösungsansatz, wie zukünftig die IT-Sicherheit durch Fehlervermeidung statt Fehlerbeseitigung gesteigert werden kann. Von diesem Ansatz versprechen wir uns einen Ausweg aus der bisher üblichen „Penetrate and Patch“ Methodik und der damit seit langem bekannten Probleme [And72]. Als wertvolle Hilfsmittel hierbei könnten sich Synthesewerkzeuge herauskristallisieren, für die wir Aufbau und Wirkungsweise beschreiben und Anforderungen aufzählen. Kapitel 5 enthält eine abschließende Bewertung.

2 Werkzeuge zur Überprüfung der Systemsicherheit

Im folgenden geben wir einen Überblick über gebräuchliche Analysewerkzeuge zur Untersuchung der Systemsicherheit. Dafür führen wir zunächst zwei Möglichkeiten der Klassifikation ein, bevor wir anschließend einige Beispiele solcher Werkzeuge aufzählen.

2.1 Klassifikation

In der Literatur werden Werkzeuge für Sicherheitsanalysen häufig hinsichtlich der von ihnen untersuchten Sicherheitsmerkmale unterschieden (siehe z.B. [Wat98] und [Kir00]). Dies führt zu folgender Einteilung:

- *Konfigurationsanalyse:* Diese Werkzeuge durchsuchen die Konfiguration eines Systems nach Sicherheitsschwachstellen. Dabei kann zwischen online und offline arbeitenden Werkzeugen unterschieden werden, je nachdem, ob es direkt auf dem untersuchten System ausgeführt oder ob die zu prüfende Konfiguration auf einem separaten Testrechner untersucht wird.
- *Integritätsüberprüfung:* Erste Anzeichen eines erfolgreichen Angriffs sind oftmals veränderte Systemdateien. Hierzu gehören beispielsweise der Austausch zentraler Systemprogramme durch Trojanische Pferde oder die Manipulation von Passwortdaten. Solche Änderungen können mit Hilfe kryptographischer Prüfsummen aufgedeckt werden. Programme zur Integritätsüberprüfung kontrollieren ein System regelmäßig auf nicht autorisierte Veränderungen des Dateisystems.
- *Netzwerk-Scanner:* Diese Werkzeuge untersuchen ganze Netzwerke, indem sie jedes einzelne System von außen auf Schwachstellen überprüfen, die einen unerlaubten Zugang ermöglichen.
- *Security Compliance Tools:* Die erforderlichen Maßnahmen zum Schutz eines Systems werden in einer *Sicherheitsstrategie* oder auch *Sicherheitspolitik* festgelegt. Hierzu gehören beispielsweise Aussagen zur Stärke von verwendeten Passwörtern. Mit Hilfe von Security Compliance Tools können Verstöße gegen diese Strategie aufgedeckt werden.
- *Loganalyse:* Werkzeuge zur Loganalyse machen die umfangreichen Logdaten überschaubar, die ein System im laufenden Betrieb generiert. Die Logdateien werden dafür nach relevanten Ereignissen durchsucht und das Untersuchungsergebnis in einem für den Administrator verständlichen Bericht zusammengefasst.
- *Intrusion Detection Systeme (IDS):* Die Aufgabe von IDS ist das Erkennen und (in eingeschränktem Maße) die Reaktion auf Angriffe. Hierfür vergleicht das IDS permanent den laufenden Netzverkehr mit bekannten Angriffsmustern, ähnlich der Suche eines Virenscanners nach Virenmustern in Programmdateien. Ein anderer Ansatz basiert auf der Generierung von typischen Verhaltensmustern des Systems. Abweichungen von diesen werden dann ebenfalls als potentieller Angriff gedeutet.

Darüber hinaus können Analysewerkzeuge auch nach ihrer Sicht auf das untersuchte System unterteilt werden [Eck03]. So untersucht beispielsweise ein Netzwerk-Scanner ein System von außen, ohne genaue Kenntnisse über die interne Systemstruktur zu besitzen. Dieser *Blackbox-Ansatz* simuliert damit das Verhalten eines externen Angreifers. Dem wird ein *Whitebox-Ansatz* entgegengesetzt, bei dem ein „interner Angriff“ durchgeführt wird; nach diesem Prinzip arbeiten z.B. die Werkzeuge zur Konfigurationsanalyse.

Beide Klassifikationsansätze weisen in der Praxis Vor- und Nachteile auf. Die Unterteilung nach untersuchten Sicherheitsmerkmalen vereinfacht die gezielte Auswahl eines bestimmten Werkzeugs. Jedoch können die Werkzeuge nicht immer eindeutig einer Kategorie zugeordnet werden. So können Werkzeuge zur Konfigurationsanalyse durchaus auch den Security Compliance Tools zugerechnet werden, ist die gewählte Systemkonfiguration doch ein Ausdruck der vereinbarten Sicherheitsstrategie. Auch die Unterscheidung nach Whitebzw. Blackbox-Ansatz bietet Vor- und Nachteile. So deckt der Whitebox-Ansatz nur die Konfiguration betreffende Schwächen auf. Damit ist es allerdings nicht möglich, Fehler in der Implementierung eines Systemdienstes aufzudecken. Dieses Problem stellt sich für den Blackbox-Ansatz nicht. Dafür müssen sich diese Werkzeuge erst mühsam Informationen über die Systemstruktur verschaffen, die beim Whitebox-Ansatz gegeben sind. In der Praxis wird daher idealerweise der Whitebox- mit dem Blackbox-Ansatz kombiniert.

2.2 Überblick

Die folgende Übersicht fasst die derzeit bekanntesten frei verfügbaren Werkzeuge für Sicherheitsanalysen im Bereich Unix- und Netzwerksysteme zusammen und ordnet diese den oben beschriebenen Klassen zu. Darüber hinaus existiert eine Vielzahl weiterer Werkzeuge, die jedoch meist nur eng abgegrenzte Aspekte der Systemsicherheit überprüfen. Eine vollständige Übersicht würde den Rahmen dieses Artikels bei weitem sprengen.

- **Konfigurationsanalyse:** Das *Computer Oracle and Password System* (COPS) ist ein 1990 von Daniel Farmer und Eugene H. Spafford entwickeltes Werkzeug, das die Konfiguration von Unix-Systemen hinsichtlich sicherheitsrelevanter Schwachstellen analysiert [FS94]. Auf dem untersuchten System werden dafür z.B. die Zugriffsrechte und die Konfiguration der angebotenen Systemdienste überprüft, in begrenztem Umfang auch die Integrität ausgesuchter Systemdateien. Realisiert ist COPS mit Hilfe diverser Bourne-Shell-Skripte und C-Programme. Der Quellcode ist frei verfügbar. Ähnliche Ansätze verfolgen USEIT [BSI00], TIGER [SSH93] und TITAN [FPA98], die ebenfalls Unix-Systeme untersuchen. Hauptproblem aller vier genannten Programme ist die relativ starke Bindung an den Unix-Dialekt, für den sie entwickelt wurden. Hier verspricht NIXE *(Non-Intrusive Unix Evaluation)* Abhilfe, das gezielt im Hinblick auf Portabilität entwickelt wurde und von den genannten Werkzeugen den größten Funktionsumfang besitzt [GS02]. Auch NIXE basiert auf einer Sammlung von Bourne-Shell-Skripten, verwendet aber nur Unix-Befehle, die auf jedem gängigen System verfügbar sind. Der modulare Aufbau und eine Vielzahl generischer Unterstützungsfunktionen ermöglichen eine flexible Erweiterung und Konfiguration nach eigenen Bedürfnissen. Zu den Konfigurationsanalysewerkzeugen kann auch CROCODILE *(Cisco Router Configuration Diligent Evaluator)* gezählt werden [GPS+03]. Auf einem gesonderten Rechner prüft es offline die Konfiguration von Cisco Routern. Es besteht aus einer Sammlung von PERL-Skripten, welche die Ergebnisse in einem interaktiven HTML-Bericht zusammenfassen. Auch bei CROCODILE ermöglichen ein modularer Aufbau und eine Basis von Unterstützungsfunktionen die einfache Erweiterung. Alle hier vorgestellten Werkzeuge zur

Konfigurationsanalyse gehen von einem Whitebox-Ansatz aus; sie haben vollen Zugriff auf die Systemkonfiguration und -dokumentation.

- **Integritätsüberprüfung:** Das wohl gängigste Werkzeug zur Überprüfung der Dateiintegrität ist *Tripwire* [Tria, Trib]. *Tripwire* speichert charakteristische Daten ausgewählter Systemdateien in einer Datenbank. Es verfolgt damit ebenfalls einen reinen Whitebox-Ansatz. Zu den protokollierten Charakteristika gehören neben der Dateigröße und dem Änderungsdatum auch kryptographische Hashwerte, mit denen Veränderungen zweifelsfrei nachgewiesen werden können. Durch regelmäßige Prüfung dieser Datenbank lassen sich Manipulationen des Systems sehr schnell aufspüren. *Tripwire* steht als frei verfügbare Version und als kommerzielles Produkt zur Verfügung. Neben einem größeren Funktionsumfang, unter anderem werden auch Windows-Rechner unterstützt, zeichnet sich die kommerzielle Version vor allem durch erweiterte kostenpflichtige Dienstleistungen aus.

- **Netzwerk-Scanner:** Netzwerk-Scanner sind typische Beispiele für Blackbox-Werkzeuge. Zu einem der ersten Vertreter dieser Gruppe gehört SATAN *(Security Administrator's Tool for Analyzing Networks)* von Dan Farmer und Wietse Venema [FV95]. SATAN analysiert Sicherheitslücken in TCP/IP-Netzwerken, wobei der Fokus auf Unix-Servern liegt. Es wird mit Hilfe eines HTML-Browsers bedient und setzt sich aus mehreren Modulen zusammen, die in PERL realisiert wurden. Der umfangreichste und aktuelleste frei verfügbare Netzwerk-Scanner dürfte zur Zeit jedoch *Nessus* sein [Nes]. Das modular in C++ realisierte System verfügt über ca. 1.200 Prüfkritierien, mit denen sowohl Windows- und Unix-Systeme als auch Netzwerkkomponenten untersucht werden können. Mit der *Nessus Attack Scripting Language (NASL)* wurde eine einfache Skriptsprache zur Verfügung gestellt, um die Untersuchungskriterien beliebig um neue Prüfmuster zu erweitern. *Nessus* basiert auf einer Client/Server-Architektur. Während der Server die eigentlichen Prüfläufe ausführt, erfolgt die Konfiguration über die graphische Benutzerschnittstelle der Client-Software. Für den Server wird ein Unix-System vorausgesetzt, Client-Implementierungen gibt es sowohl für Unix als auch für Windows.

- **Security Compliance Tools:** Zu den Security Compliance Tools kann beispielsweise der Passwort-Cracker *John the Ripper* gezählt werden [JtR]. Mit seiner Hilfe kann die Sicherheit der Benutzerpasswörter eines Systems untersucht werden. Dazu versucht *John*, mit Hilfe von speziellen Wörterbüchern und Heuristiken das Passwort eines oder mehrerer Systembenutzer zu „erraten". Wie bereits in Kapitel 2.1 erwähnt, lassen sich auch die Werkzeuge zur Konfigurationsanalyse zu den Security Compliance Tools zählen. Da *John* als Eingabe die verschlüsselten Passwortdaten benötigt, gehört auch er zu den Whitbox-basierten Werkzeugen.

- **Loganalyse:** Fast jeder Angriff hinterlässt Spuren in den Logdateien des angegriffenen Systems. Die Schwierigkeit besteht jedoch darin, die passenden Informationen aus der Fülle der gesammelten Logdaten herauszufiltern. Diese Aufgabe kann mit Programmen wie *Swatch* [HA93] oder SHARP *(Syslog Heuristic Analysis and Response Program)* [BE00] vereinfacht werden. Auf Basis eines frei konfigurierbaren Regelsatzes in Form regulärer Ausdrücke werden die Systemlogs durchsucht. Je

nach Ergebnis können nahezu beliebige Aktionen ausgeführt werden. Diese können vom Speichern des Analyseberichts über die Benachrichtigung des Administrators per Email bis hin zur Beendigung akut gefährdeter Systemdienste reichen. Im Gegensatz zu *Swatch* rettet SHARP außerdem den aktuellen Programmzustand hinüber zum nächsten Aufruf. Dies führt zu einer drastischen Reduktion redundanter Analyseergebnisse. Loganalysewerkzeuge sind praktisch Teil des Systems, d.h. sie verfolgen einen Whitebox-Ansatz.

- **Intrusion Detection Systeme:** *Snort* ist ein frei verfügbares Intrusion Detection System (IDS) [Roe99]. Sein Ziel ist das zeitnahe Aufspüren von Angriffen durch gezieltes Durchsuchen des Netzverkehrs zur Laufzeit nach vorkonfigurierten Angriffsmustern. Je nach Konfiguration wird dann eine Reaktion ausgeführt, beispielsweise der Administrator per Email alarmiert. Derzeit durchsucht *Snort* den Netzverkehr nach ca. 1.800 bekannten Angriffsmustern, die auch vom Benutzer beliebig erweitert werden können. Weitere bekannte freie IDS sind Bro [Pax99] und LIDS [LID], die jedoch insbesondere in puncto Bedienbarkeit nicht an *Snort* heranreichen. Mit Hilfe eines IDS lassen sich sowohl Innen- als auch Außentäter aufspüren. Es wird also sowohl ein Whitebox- als auch ein Blackbox-Ansatz verfolgt.

Dieser allgemeine Überblick verdeutlicht die Vielfalt vorhandener Analysewerkzeuge und deren Heterogenität. Dies führt beim Einsatz in der Praxis immer wieder zu Problemen. Im nächsten Abschnitt formulieren wir daher einige grundlegende Anforderungen, um diese Schwierigkeiten zu verringern.

3 Anforderungen an Audit-Werkzeuge

Die große Vielfalt an verschiedenen Werkzeugen, die sich zum Teil erheblich in puncto Funktionsumfang, Bedienung, Aktualität und Systemvoraussetzungen unterscheiden, verdeutlicht die Schwierigkeiten bei den heute verfügbaren Analysewerkzeugen. Auf Grund der hohen Komplexität und Mannigfaltigkeit aller zu berücksichtigenden sicherheitsrelevanten Aspekte kann keines der Werkzeuge den Anspruch auf Vollständigkeit erheben. Allerdings wird dadurch letztlich auch die Arbeit des Sicherheitsauditors erschwert, denn für verschiedene Problematiken sind auch verschiedene Werkzeuge mit teilweise verschiedenen Benutzeroberflächen und Ausgabeformaten von Nöten.

Um diesen und ähnlichen Widrigkeiten konstruktiv begegnen zu können, müssen an zukünftige Analysewerkzeuge strengere Anforderungen gestellt werden. Damit Sicherheit nicht nur ein Schlagwort bleibt, sondern auch verinnerlicht und gelebt wird, müssen Auditierungswerkzeuge einfacher verständlich und nachvollziehbar, intuitiver bedienbar sowie hilfreicher in der Anwenderunterstützung werden. Dies hat schließlich auch den Hintergrund, dass Sicherheit nicht mehr ausschließlich im Verantwortungsbereich von Spezialisten liegen darf, sondern auch auf den weniger versierten Anwender zumindest in Ansätzen übertragen werden muss. Dies erscheint um so wichtiger, da vom ökonomischen Standpunkt gesehen jeder einzelne für die Sicherheit seines Systems verantwortlich gemacht werden muss. Nur so lässt sich ein Anreiz für die Investition in Sicherheit schaffen

[Var00]. Hauptgedanke zur stärkeren Sensibilisierung auch des Einzelnen ist eine Auftrennung der Arbeitsschritte innerhalb einer Sicherheitsauditierung in drei Stufen: Konfiguration, Analyse und Reaktion. Nach wie vor bleibt es eine Aufgabe der Experten, die Auditierungswerkzeuge zu entwickeln und ihr Wissen dort zu integrieren. Gleichzeitig können bei der Analyse der für das Sicherheitskonzept entscheidenden Komponenten jedoch Aufgaben deligiert werden. Hier sind Sicherheitsspezialisten nicht mehr zwingend notwendig, zumindest solange bei der Auditierung keine schwerwiegenden Sicherheitslücken aufgedeckt werden. Erst in der dritten Stufe, der Reaktion auf entdeckte Probleme, sind Wissen und Fähigkeit der Experten wieder gefragt. So können sich die Spezialisten nicht nur besser auf die ständig erforderliche Aktualisierung der Sicherheitswerkzeuge konzentrieren, sondern ihr Handlungsbedarf bei erkannten Sicherheitsmängeln kann auf die akuteren Fälle eingegrenzt werden, während die weniger kritischen Vorfälle von untergeordneten Instanzen bearbeitet werden können. Positiver Nebeneffekt dieses Konzeptes ist ein Schärfen des Sicherheitsbewusstseins bei den mit der Analyse betrauten Benutzern.

Als unabdingbare Grundvoraussetzung für die Umsetzung dieses Konzeptes ist der Anwender der Werkzeuge in den Vordergrund zu stellen, was der Optimierung der Benutzbarkeit einen hohen Stellenwert verleiht. Im folgenden finden sich Ansätze und Lösungswege, mit denen sich dieses Ziel erreichen lässt.

Ein Schritt zur Verbesserung besteht in einer Vereinheitlichung auf technischer Ebene, denn es trägt nicht zur guten Benutzbarkeit bei, wenn zur umfassenden Sicherheitsüberprüfung mehrere, in ihrer Bedienung teilweise stark unterschiedliche Werkzeuge eingesetzt werden müssen [Fin99]. Eine deutliche Erhöhung der Akzeptanz auf Seiten der Anwender ließe sich dann erzielen, wenn zur umfassenden Sicherheitsanalyse möglichst wenige Werkzeuge zum Einsatz kommen müssen. Im Idealfall wird sogar nur ein einzelnes, flexibles Rahmenwerk benötigt, welches je nach Bedarf mit verschiedenen Prüfmodulen versehen werden kann, sich dabei dem Anwender aber mit einer einheitlichen und leicht verständlichen Oberfläche präsentiert. Ein derart modularisiertes Konzept ist nur mittels standardisierter und vereinheitlichter Ein- und Ausgabeschnittstellen realisierbar [PCW85]. Zu diesem Zweck müssen sich die Werkzeuge einander annähern, damit sie nicht nur technisch, sondern auch logisch dieselbe Sprache sprechen.

Die Vergleichbarkeit von Analyseergebnissen unterschiedlicher Werkzeuge kann durch eine einheitliche Klassifikation hinsichtlich der Kritikalität des gemachten Befunds erreicht werden. Hier sollte Wert auf eine Unterteilung in nur wenige, scharf gegeneinander abgegrenzte Abstufungen gelegt werden. Tabelle 1 zeigt, wie eine solche Klassifikation in fünf Stufen aussehen kann [GS02, GPS$^+$03].

Gleichzeitig darf sich ein Auditierungswerkzeug jedoch nicht auf die reine Einteilung der Analyseergebnisse in wenige Kategorien beschränken. Nur durch eine Erweiterung der Klassifikation um hilfreiche und erläuternde Kommentare sowie konstruktive und korrekte Lösungsvorschläge können Werkzeuge ihren Nutzen für den Anwender demonstrieren. Bei einem hohen Wert des integrierten Wissens kann die Bereitschaft zur Verwendung eines Werkzeugs signifikant erhöht werden [Gro03]. Dieses Verhalten muss außerdem einhergehen mit einer Transparenz der Arbeits- und Funktionsweise des Werkzeugs, denn nur so kann endgültig das unverzichtbare Vertrauen des Anwenders gewonnen werden.

Klasse	Erläuterung
INFO	Die Aussage ist rein informeller Natur und nicht sicherheitsrelevant.
OKAY	Der Prüfgegenstand ist als sicher zu bewerten.
CHECK	Der Prüfpunkt kann vom Werkzeug nicht einwandfrei klassifiziert werden. Eine manuelle Bewertung und Interpretation ist erforderlich.
WARN	Der analysierte Aspekt ist bezüglich der Sicherheit fragwürdig und sollte einer Revision unterzogen werden.
ALERT	Es wurde ein kritisches Sicherheitsproblem entdeckt.

Tabelle 1: Klassifikation von Analyseergebnissen

Damit der Benutzer seine Skepsis gegenüber dem Einsatz eines Werkzeugs verliert, darf dieses niemals das zu untersuchende System verändern. Die Aufgabe von Analysewerkzeugen sollte es immer sein, Informationen zu sammeln, zu verarbeiten, zu bewerten und abschließend zu kommentieren.

Weitere Kriterien, die entscheidenden Einfluss auf den Einsatz von Auditierungswerkzeugen haben können, finden sich oftmals auch in Details. Bereits ein geringfügig zu hoher Anspruch an die Systemvoraussetzungen kann die Verwendung eines Werkzeugs erheblich erschweren oder sogar verhindern. Hier sollte das Prinzip der minimalen Vorgaben verfolgt werden. Dadurch schafft man die Grundlagen für ein Werkzeug, welches im Idealfall keinerlei und im ungünstigsten Fall höchstens minimale Auswirkungen auf das zu untersuchende System besitzt. Vorteil eines so konzipierten Werkzeugs ist die grundlegende Fähigkeit, im Wirkbetrieb eines Systems störungsfrei eingesetzt werden zu können.

Schließlich besteht gegenüber Analysewerkzeugen aus Anwendersicht ein unerfüllbarer Anspruch an deren Vollständigkeit, dem nur durch Flexibilität und eine Offenheit für Aktualisierungen und Anpassungen entgegengewirkt werden kann. Letztendlich darf Sicherheit niemals in einen statischen Zustand gelangen, sondern muss immer als ein Prozess verstanden werden.

4 Vision

Für die Überwachung der Einhaltung der in der Sicherheitspolitik vereinbarten Vorgaben ist eine umfassende, regelmäßige und ständig an aktuelle Sicherheitsbedürfnisse angepasste Auditierung aller relevanten Netzwerkkomponenten zwar wichtig, jedoch können so nur Symptome behandelt werden. Um dieser Problematik vorbeugend begegnen zu können, muss ein vollständig anderer Ansatz gewählt werden. Zwar wäre es ebenso falsch, die Strategie „Funktionalität vor Sicherheit“ umzukehren, jedoch darf der Sicherheitsgedanke nicht erst nach der Implementierung von Systemen berücksichtigt werden. Vielmehr muss bereits bei der Konfiguration und Integration sicherheitskritischer Komponenten eine Fehlervermeidung angestrebt werden [VM02].

Langfristig zur Lösung dieses Problems beitragen könnten auf allen sicherheitsrelevanten

Informationen, Erfahrungen und Empfehlungen basierende Synthesewerkzeuge. Die Vorteile liegen auf der Hand: Sicherheit entsteht nicht mehr dadurch, dass viele Experten auf verschiedensten Wegen dieselben Probleme immer wieder neu lösen, sondern das gebündelte Wissen in den Synthesewerkzeugen wäre ohne tiefgehende Kenntnis der dahinter liegenden Sicherheitskonzepte direkt verfügbar. Auf diese Weise können Fehler schon im Ansatz vermieden werden, und die Sicherheit muss nicht erst nachträglich durch Analysen überprüft und bestätigt werden. Im Idealfall wären durch den problemvermeidenden Ansatz der Synthesewerkzeuge keine Auditierungen mehr notwendig, da alle Informationen, die in einem Analysewerkzeug enthalten wären, bereits bei der Synthese berücksichtigt würden. Bei neu auftretenden Sicherheitsproblemen würden dann auch nicht mehr die Auditierungswerkzeuge aktualisiert, damit sie danach die vorher unbekannten Schwachstellen aufspüren könnten, sondern die Konfiguration aller sicherheitsrelevanten Komponenten würde nach den aktuellen Sicherheitsaspekten neu synthetisiert. Sicherheitsanalysen würden in diesem Konzept nur noch der Überprüfung der Integrität der Komponenten dienen, wären aus demselben Grund allerdings auch niemals vollständig verzichtbar.

Die offensichtliche Problematik bei dieser Herangehensweise ist unübersehbar: Ein so geartetes Synthesewerkzeug ist nicht trivial zu entwickeln, da das Werkzeug bereits über alle sicherheitsrelevanten Informationen verfügen muss. Der Umfang des zu integrierenden Wissens ist derart hoch, dass die Realisierung innerhalb eines einzelnen Werkzeugs nur schwerlich durchführbar ist. Vielversprechender ist hier der Ansatz einer Aufteilung in Synthesemodule mit standardisierter Schnittstelle, die über ein gemeinsames Rahmenwerk miteinander verknüpft sind. So können die Module jeweils einen hohen Spezialisierungsgrad bezüglich einzelner Komponenten besitzen, ohne dass dabei die unumgängliche ganzheitliche Betrachtung der Sicherheitsproblematik aufgegeben werden muss.

Deutlich wird die Herausforderung auch unter Berücksichtigung der Tatsache, dass ein Synthesewerkzeug nicht wie ein Auditierungswerkzeug sukzessive durch Erweitern der betrachteten und analysierten Sicherheitsaspekte ergänzt werden kann, um ein ständig steigendes Sicherheitsniveau zu gewährleisten. Hier ist im Gegenteil eine vollständige Berücksichtigung aller Teilaspekte eines einzelnen Sicherheitspunktes unerlässlich. Um so mehr rückt auch die Forderung nach einer einheitlichen Schnittstelle der beteiligten Module bzw. Werkzeuge in den Vordergrund. Nur mit offenen und vordefinierten Anbindungspunkten lassen sich so verschiedenste Werkzeuge und Werkzeugarten fehlerfrei miteinander verbinden und sinnvoll kombinieren.

Die großen Hindernisse, die es auf dem Weg zu einem vollständigen Synthesewerkzeug noch zu überwinden gilt, haben in der Praxis bislang nur für eine unzureichende Verfolgung dieses Ansatzes gesorgt. Heute verfügbare Synthesewerkzeuge, wie z.B. CISCO CONFIGMAKER [CCM] oder Solsofts NP™[Sol], scheitern am Anspruch der Vollständigkeit und können bestenfalls für Teilaspekte hinreichend gute Lösungen bieten [CCM]. Zu diesem Missstand tragen hauptsächlich zwei Punkte bei: zum einen die wirtschaftlich begründete Ausrichtung primär auf Funktionalität statt auf Sicherheit und zum anderen die Probleme bei der technischen Umsetzung des ganzheitlichen Grundgedankens.

Mit NP™kann beispielsweise die Netzwerktopologie mit allen verwendeten Komponenten und Kommunikationsbeziehungen grafisch konfiguriert werden, wobei Standardkomponenten marktführender Hersteller bereits unterstützt werden. Anschließend können dann

Konfigurationsfragmente für die einzelnen Netzwerkkomponenten bezüglich Zugriffslisten, Netzwerkadressumsetzung, virtuellen privaten Netzwerken etc. erstellt werden. Diese Konfigurationsfragmente werden jedoch unstrukturiert und ohne eindeutigen Zusammenhang dargestellt, sodass die Übersicht über das Gesamtkonzept und somit die Transparenz für den Benutzer verloren gehen.

Eine Vielzahl von Werkzeugen, deren Aufgabenschwerpunkt die Synthese ist, verfolgen ihr Ziel mit Hilfe sogenannter Assistenten, die in einem Interviewmodus durch gezielte Fragen Informationen sammeln, die für die korrekte Konfiguration der dahinter liegenden Systeme notwendig sind. Wie in anderen Bereichen auch zeigt sich hier der Einfluss der Netzwerkökonomie auf das Sicherheitsbewusstsein bei der Implementierung und Konfiguration von Systemen [And01]: Hintergrund der Assistenten ist oftmals nicht das Erhöhen der Sicherheit sondern vielmehr die Schaffung und Verbesserung der Benutzerfreundlichkeit und damit die Steigerung des wirtschaftlichen Nutzens. Zum mangelnden Sicherheitsbewusstsein kommen erschwerend die Widrigkeiten bei der technischen Umsetzung hinzu. Die nur unzureichend ganzheitliche Auffassung von Sicherheit und das hochgradig komplexe Zusammenspiel der verschiedenen sicherheitsrelevanten Komponenten haben letztendlich erst zu den Problemen der heute verfügbaren Werkzeuge geführt.

Unser Lösungsansatz ist eine grundlegend abgewandelte Auffassung von Sicherheit, nicht als Kaleidoskop von Einzelaspekten, sondern als ein komponentenübergreifender Satz von Regeln, der einen Sicherheitsgedanken ganzheitlich beschreibt. Beispielsweise im Netzwerkbereich darf der Fokus nicht mehr auf die unabhängige Optimierung der einzelnen Komponenten gesetzt werden, sondern die Umsetzung der Sicherheitspolitik muss alle Elemente gleichzeitig berücksichtigen, da selbst eine einfache Verbindung zwischen zwei Knoten alle dazwischen liegenden Objekte gleichermaßen betrifft. Durch diese Vorgehensweise werden nicht nur typische Sicherheitsprobleme vermieden, sondern es wird auch das Sicherheitsbewußtsein geschärft und eine Sensibilisierung weg vom reduzierten Blick auf einzelne Elemente in Richtung der Wahrnehmung des Ganzen geschaffen.

Unser Synthesewerkzeug zur vollständig Konfiguration eines Netzwerks befindet sich derzeit in einer frühen Entwicklungsphase. Die Basis bildet ein Rahmenwerk von Servicefunktionen. Die eigentliche Funktionalität wird in einer Vielzahl generischer Basisklassen gekapselt, von denen für die einzelnen Netzelemente spezifische Instanzen abgeleitet werden. Das Hauptaugenmerk der Entwicklungsarbeiten liegt derzeit auf der Entwicklung eines ersten Demonstrators. Erklärtes Ziel ist die Bereitstellung einer Anwendung, die bei gegebener Sicherheitsstrategie und Netzstruktur automatisch das gesamte Netzwerk nach den Wünschen des Systemadministrators konfiguriert.

5 Zusammenfassung

Trotz langjähriger leidvoller Erfahrungen wird die Sicherheit bei der Entwicklung und dem Betrieb von IT-Systemen immer noch vernachlässigt. Die Komplexität und Undurchschaubarkeit heutiger Systeme tun ihr Übriges, sodass die Thematik selbst von versierten Administratoren immer noch als „Buch mit sieben Siegeln“ empfunden wird. Wir haben

mehrere Ansätze demonstriert, diese Probleme zu entschärfen und damit das allgemeine Sicherheitsniveau zu erhöhen. Eine Auftrennung des Sicherheitsauditierungsprozesses in die drei Schritte Konfiguration, Analyse und Reaktion sowie die teilweise Übertragung des Analyseschrittes auf den einzelnen Benutzer soll das Sicherheitsbewusstsein des Einzelnen erhöhen. Um dies zu ermöglichen, haben wir Vorschläge zur Zusammenführung der unterschiedlichen Analysewerkzeuge gemacht. In Verbindung mit den Methoden des Security Engineerings soll darüber hinaus das von uns vorgestellte Konzept eines Synthesewerkzeugs in Zukunft die Durchführung aufwändiger Sicherheitsrevisionen minimieren. Solche Werkzeuge können zur Fehlervermeidung bereits bei der Konfiguration eingesetzt werden. Die Herausforderung hierbei liegt allerdings in der Vollständigkeit der dem Werkzeug zur Verfügung gestellten Informationen. Neben der weiteren Entwicklung unseres Prototyps sind außerdem noch die Einbindung unserer Vorschläge in bestehende organisatorische Prozesse zu klären, damit Sicherheit letztendlich auch gelebt wird.

Literatur

[And72] James P. Anderson. Computer Security Technology Planning Study Volume II. Technical Report ESD-TR-73-51, Electronic Systems Division, US Air Force Systems Command, Hanscom Field, Bedford, MA, October 1972.

[And01] R. Anderson. Why Information Security is Hard - An Economic Perspective. In *17th Annual Computer Securtity Applications Conference*, December 2001.

[Bal98] Helmut Balzert. *Lehrbuch der Software-Technik, Teil 1: Software-Entwicklung*. Spektrum Akademischer Verlag, 1998.

[BE00] Matt Bing and Carl Erickson. Extending UNIX System Logging with SHARP. In *Proceedings of the 14th Systems Administration Conference (LISA 2000)*, New Orleans, Louisiana, USA, December 2000. USENIX Association.

[BSI00] Bundesamt für Sicherheit in der Informationstechnik (BSI), Bonn. *BSI-Tool Sichere UNIX-Administration (USEIT). Version 2.0. Anwender-Handbuch*, November 2000.

[CCM] CISCO CONFIGMAKER, Version 2.6. Cisco Systems Incorporated, http://www.cisco.com/warp/public/cc/pd/nemnsw/cm/index.shtml.

[CER02] Overview of Attack Trends. CERT (Computer Emergency Response Team) Coordination Centre, http://www.cert.org/archive/pdf/attack_trends.pdf, February 2002.

[Eck03] Claudia Eckert. *IT-Sicherheit – Konzepte, Verfahren, Protokolle*. Oldenbourg Verlag München, 2003.

[Fin99] Craig A. Finseth. *The Craft of Text Editing – Emacs for the Modern World*. http://www.finseth.com/ fin/craft, 1999.

[FPA98] Dan Farmer, Brad Powell, and Matthew Archibald. TITAN. In *Proceedings of the Twelth Systems Administration Conference (LISA '98)*, Boston, MA, December 1998. USENIX Association.

[FS94] Daniel Farmer and Eugene H. Spafford. The COPS Security Checker System. Technical Report CSD-TR-993, Department of Computer Science, Purdue University, January 1994. Originally published in the proceedings of the Summer Usenix Conference 1990, Anaheim, CA.

[FV95] Dan Farmer and Wietse Venema. SATAN: Security Administrator's Tool for Analyzing Networks. http://www.fish.com/zen/satan/satan.html, 1995.

[GPS+03] Stephan Groß, Holger Peine, Thomas Schwenkler, Reinhard Schwarz, and Kai Simon. CROCODILE *Benutzerhandbuch – Der Cisco Router Configuration Diligent Evaluator*. Fraunhofer IESE, Kaiserslautern, 2.1 edition, January 2003. IESE Report-Nr. 067.02/D.

[Gro03] Nielsen Norman Group. 233 Tips and Tricks for Recruiting Users as Participants in Usability Studies. http://www.nngroup.com/reports/tips/recruiting/index.html, 2003.

[GS02] Stephan Groß and Reinhard Schwarz. NIXE™ *– Ein Werkzeug für Unix Sicherheitsrevisionen. Anwenderhandbuch*. Fraunhofer IESE, Kaiserslautern, February 2002.

[HA93] Stephen E. Hansen and E. Todd Atkins. Automated System Monitoring and Notification With Swatch. In *Proceedings of the 7th System Administration Conference (LISA)*, Monterey, CA, November 1993. USENIX Association.

[JtR] Openwall Project: John the Ripper Password Cracker. http://www.openwall.com/john.

[Kir00] Ulhas Kirpekar. A Linux Based Integrated Security Monitoring Tool. Master's thesis, Department of Computer Science & Engineering, Indian Institute of Technology, Kanpur, India, February 2000.

[LID] Linux Intrusion Detection System (LIDS). http://www.lids.org.

[Nes] The Nessus Project. http://www.nessus.org/.

[Pax99] Vern Paxson. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks*, 31(23–24):2435–2463, December 1999.

[PCW85] D. L. Parnas, P. C. Clements, and D. M. Weiss. The Modular Structure of Complex Systems. *IEEE Transactions on Software Engineering*, SE-11(3):259–266, March 1985.

[Roe99] Martin Roesch. Snort – Lightweight Intrusion Detection for Networks. In *Proceedings of the 13th Large Installation System Administration Conference (LISA)*, pages 229–238, Seattle, Washington, November 1999. USENIX Association.

[SG03] Thomas Schwenkler and Stephan Groß. Die Umsetzung von Netzsicherheitskonzepten in heterogenen Organisationen. In *17. DFN-Arbeitstagung über Kommunikationsnetze*, LNI, Düsseldorf, June 2003.

[Sol] Solsoft NP Workstation. http://www.solsoft.com.

[SSH93] David R. Safford, Douglas Lee Schales, and David K. Hess. The TAMU Security Package: An Ongoing Response to Internet Intruders in an Academic Environment. In *Proceedings of the Fourth USENIX Security Symposium*. USENIX Association, October 1993.

[Tria] Tripwire. http://www.tripwire.org.

[Trib] Inc. Tripwire. Tripwire. http://www.tripwire.com.

[Var00] Hal R. Varian. Managing Online Security Risks. *Economic Science Column, The New York Times*, June 1 2000.

[VM02] J. Viega and G. McGraw. *Building secure Software*. Addison Wesley Professional, Boston, MA, 2002.

[Wat98] Bret Watson. Current Practices in Audit Automation. Presented at TACS-Asia, The Oriental Hotel, Singapore, June 1998.