CROCODILE — Ein Werkzeug zur sichtenbasierten Sicherheitsprüfung von Router-Konfigurationen

Holger Peine, Reinhard Schwarz

Fraunhofer Institut Experimentelles Software Engineering (IESE)
Sauerwiesen 6
67661 Kaiserslautern
{peine, schwarz}@iese.fraunhofer.de

Abstract: Router stellen eine kritische Komponente in IP-Netzen dar, für die bisher kaum Werkzeuge zur Sicherheitsüberprüfung existieren. Mit CROCODILE haben wir ein solches Werkzeug entwickelt, das die Sicherheit einer Router-Konfiguration unter verschiedenen, modular gegliederten Aspekten prüft und seine Ergebnisse in einer sichtenbasierten Darstellung präsentiert. Dadurch kann der Benutzer sich auf einzelne Aspekte konzentrieren und schrittweise in immer speziellere Details hinein navigieren. CROCODILE erfasst die Sicherheitsimplikationen verschiedener, potentiell weit verstreuter, aber aufeinander wirkender Konfigurationsklauseln, statt wie bisherige Werkzeuge nur einzelne Klauseln isoliert zu betrachten.

1 Sicherheitsüberprüfung von Router-Konfigurationen

1.1 Router und Routing

Alle größeren Unternehmen sind heute an das Internet angeschlossen und setzen die Internet-Technik auch intern als Intranet ein. Wesentlichen Einfluss auf die Sicherheit solcher IP-Netze haben die Router, mit deren Hilfe IP-Pakete zwischen Ursprungs- und Ziel-Rechnern vermittelt werden. Moderne Router übernehmen nicht nur Vermittlungsfunktionen, sondern dienen zugleich auch der Filterung, Separierung, Verschlüsselung und Überwachung von Datenströmen. Zur Erfüllung ihrer Kernaufgaben benötigen Router etliche Zusatzfunktionen. Sie tauschen zum Beispiel mit anderen Netzwerkkomponenten dynamische Wegewahl- und Zeitinformationen aus. Außerdem stellen sie diverse Management-Schnittstellen für Konfiguration, (Fern-)Wartung und Überwachung bereit. Die hier skizzierten Aufgaben eines Routers sind allesamt sicherheitskritisch, denn sie beeinflussen die Verfügbarkeit, Integrität und Vertraulichkeit von Datenverbindungen.

Das Verhalten eines Routers wird durch seine Konfiguration bestimmt. Die Erstellung einer Routerkonfiguration ist jedoch eine schwierige Aufgabe. Hier kann Werkzeugunterstützung erheblich dazu beitragen, verborgene Konfigurationsschwächen aufzuspüren, und Sicherheitsanalysen vollständiger, schneller, und objektiver zu gestalten.

1.2 Ein Werkzeug zur Sicherheitsanalyse von IOS Routern

Mit CROCODILE haben wir mit Unterstützung der Deutschen Telekom ein solches Prüfwerkzeug für Router des Marktführers Cisco Systems entwickelt [GS01], die unter dem Betriebssystem IOS betrieben werden [Ci99]. Dank der modularen, IOS-unabhängigen Struktur des Werkzeugs ist es möglich, durch Austausch der betroffenen Module auch andere Gerätetypen in gleicher Weise zu überprüfen.

Die korrekte Benutzung von IOS stellt den Netzwerkadministrator vor eine schwierige Aufgabe, obwohl typische Konfigurationen oft weniger als 1000 Zeilen umfassen. Die Möglichkeiten dieses Betriebssystems sind zahlreich, kompliziert und oft schlecht dokumentiert. Überdies ist die IOS-Syntax ausdrucksschwach, wenig intuitiv und mitunter mehrdeutig. Entsprechend hoch ist das Risiko unbemerkter Konfigurationsfehler.

CROCODILE liest IOS-Konfigurationen ein, analysiert diese auf potentielle Mängel, und erzeugt eine Hypertext-Ausgabe mit den Befunden. Dabei können eigentlich triviale Mängel, die aber dem menschlichen Auge leicht entgehen, automatisch erkannt werden — wie etwa Syntaxfehler und Inkonsistenzen aufgrund von Tipp- oder Flüchtigkeitsfehlern, die allesamt vom Router selbst *nicht* angemahnt würden. Das Werkzeug geht aber weit über eine solche rein syntaktische Analyse hinaus:

- CROCODILE unternimmt auch eine *inhaltliche Analyse* der Konfiguration, die zum Beispiel auch die Reihenfolge oder das Fehlen von Klauseln sowie deren Bedeutung im Kontext einbezieht. Dies ist nur möglich, weil das Werkzeug die Konfigurationsbeschreibung nicht nur zeilenweise, sondern auch als Ganzes betrachtet.

- Die Prüfergebnisse werden in verschiedene *Analyse-Sichten* geordnet, wobei eine solche Sicht die Befunde zu einem spezifischen Konfigurationsaspekt zusammenfasst (z.B. zur Authentisierung oder zum Logging). Der Anwender kann Sichten frei definieren. Die einer Sicht zugrunde liegenden Klauseln können dabei innerhalb der Konfigurationsbeschreibung weit verstreut sein, und eine Klausel kann in mehrere Sichten eingehen. Die sichtenbasierte Darstellung verdeutlicht die hinter der Konfiguration stehenden Überlegungen und lässt Fehler darin klarer hervor treten.

- Die oft verwirrend vielfältigen Informationen werden in einer Reihe von *aufgabenspezifischen Darstellungen* angeboten. So bietet das Werkzeug eine kommentierte Gesamtübersicht, eine differentielle Darstellung relativ zu früheren Konfigurationen, Detailanalysen, statistische Daten, und bequeme Hyperlinks zu Hersteller-Informationen. Einige besonders innovative Detailanalysen beruhen auf der Berechnung der von der Zugriffskontrollliste eines Router-Interface effektiv zugelassenen und abgewiesenen Pakete (siehe Abschnitt 0).

CROCODILE entlastet somit von mühsamer Kleinarbeit und hebt die Router-Konfiguration auf eine logische Ebene, wo der menschliche Experte angemessen unterstützt wird. Keines der uns bekannten Werkzeuge für Router-Konfigurationen (siehe Abschnitt 4) hat auch nur annähernd diese Fähigkeit.

2 CROCODILE-Entwurfsprinzipien und -Architektur

Abgesehen von funktionalen Anforderungen wie der sichtenbasierten Befunddarstellung waren Modularität, Erweiterbarkeit und Portierbarkeit wichtige Entwurfsziele. CROCODILE ist als erweiterbares Framework realisiert, das alle Funktionen für eine allgemeine, textbasierte Sicherheitsprüfung einschließlich der Befundausgabe bietet, während spezifische Prüfaufgaben in unabhängigen Prüfmodulen gekapselt sind. Prüfmodule sind zwecks Anpassung an lokale Gegebenheiten und Sicherheitsanforderungen konfigurierbar, und durch eine Implementierung in Perl ohne Benutzung besonderer Zusatzbibliotheken ist CROCODILE unter den gängigen Betriebssystemen sofort lauffähig.

2.1 Komponenten

Das Werkzeug gliedert sich grob in drei Arten von Komponenten: ein Mustererkennungsmodul (Parser), eine beliebige Anzahl von Prüfmodulen sowie ein Darstellungsmodul zur Ergebnisaufbereitung. Ein Prüfmodul realisiert eine in sich abgeschlossene Prüfaufgabe mit allen logischen Prüf- und Interpretationsschritten. Der Parser ruft die Prüfmodule mit Eingabedaten zur Analyse auf. Die Darstellungskomponente erzeugt die grafische Präsentation des Befunds. Abbildung 1 zeigt die Komponenten im Überblick.

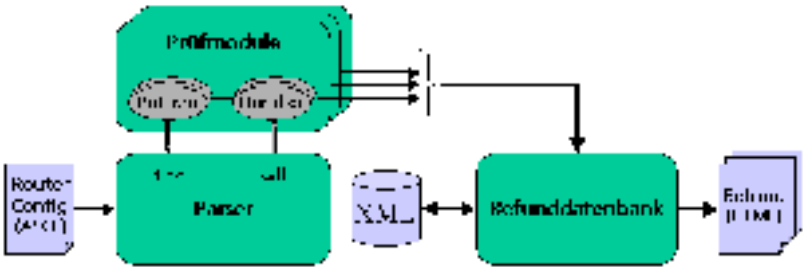

Abbildung 1: Struktur und Funktionsweise von CROCODILE

2.2 Arbeitsweise

Der Parser fragt zunächst alle Module nach den Textmustern, an denen sie interessiert sind. Diese Muster (Patterns) sind in Backus-Naur-Form beschrieben, ähnlich dem Format der Cisco-Dokumentation [Ci99], und bieten die üblichen Operatoren Sequenz, Alternative, Wiederholung, und Verneinung. Zur Vereinfachung können auch Pattern-Makros definiert werden. Danach liest der Parser eine IOS-Konfigurationsdatei im Textformat ein. Er vergleicht Textzeile für Textzeile mit den angemeldeten Textmustern. Sobald eines der Textmuster im Konfigurationstext erkannt ist, benachrichtigt der Parser alle Prüfmodule, die ihr Interesse für das betreffende Muster angemeldet hatten; er übergibt ihnen die Textzeile in einer dem Muster entsprechenden Zerlegung zur Auswertung.

Erhält ein Prüfmodul eine Benachrichtigung, so führt es eine dem Textmuster zugeordnete Prüfroutine (Pattern Handler) aus. Der Handler ist eine Methode des Prüfmoduls und umfasst alle erforderlichen Prüfschritte beim Auftreten des Textmusters. Dabei kann das Prüfmodul durchaus auf Informationen aus früheren Aufrufen (auch anderer Module) zurückgreifen und so eine Zeilen übergreifende Sicht gewinnen. Der Handler liefert ein Prüfergebnis und gegebenenfalls ergänzende Kommentare dazu. Alle Befunde werden in eine interne Datenbank eingespeist.

Sind alle Textzeilen der Eingabedatei in dieser Weise verarbeitet worden, so wird reihum jedes Prüfmodul per Methodenaufruf noch einmal aufgefordert, eine abschließende Gesamtwertung seines jeweiligen Prüfbefunds zurückzumelden. Dies bietet den Modulen die Gelegenheit, aus der Summe aller ihnen gemeldeten Textzeilen eine integrierte Sicht auf den zu prüfenden Sicherheitsaspekt zu konstruieren.

Die Rückmeldungen aller Module ergänzen die in der Datenbank gesammelten Befunde. Diese Rohdaten werden genutzt, um daraus verschiedene HTML-Darstellungen zu generieren, die sich mit einem Standard-Web-Browser darstellen lassen.

2.3 Konstruktion eigener Prüfmodule

Mit der Schnittstelle zwischen dem Parser und den Prüfmodulen kommt der Anwender normalerweise nicht in Berührung, selbst wenn er eigene Prüfmodule konstruiert. Das Zusammenspiel ergibt sich automatisch aufgrund der Vererbungsmechanismen, da alle Prüfmodule von einer gemeinsamen Basisklasse `Module` abgeleitet sind. Im Kern schrumpft damit die Konstruktion neuer Prüfmodule auf drei Punkte zusammen:

1) Spezifiziere die Patterns, die analysiert werden sollen.

2) Schreibe pro Pattern einen Handler, der die gefundenen Textfragmente analysiert.

3) Falls nötig, schreibe einen Nachbearbeitungs-Handler, der alle Befunde des Moduls einer abschließenden Gesamtanalyse unterzieht.

CROCODILE wird bereits mit vordefinierten Prüfmodulen für die wichtigsten Prüfbereiche ausgeliefert; zwei davon werden in Abschnitt 0 beispielhaft vorgestellt.

2.4 Befund-Datenbank

Alle Prüfmodule speichern ihre Befunde in strukturierter Form in einer Datenbank ab. Die wichtigsten Konzepte in diesem Zusammenhang sind Anmerkungen und Sichten.

Anmerkungen sind Kommentare zu einer Konfigurationseigenschaft. Jede Anmerkung ist mit einer *Gefahreneinstufung* versehen (`OKAY`, `INFO`, `CHECK`, `WARN` oder `ALERT`). Die Einstufung gibt an, ob es sich bei der Anmerkung um einen positiven, neutralen, unklaren, negativen oder gar kritischen Befund handelt. Die Einstufung eines Befundes wird in der Ausgabe durch entsprechende Färbung kenntlich gemacht.

Sichten fassen alle Befunde zu einem gemeinsamen Konfigurationsaspekt zusammen. Der Handler-Programmierer kann den Namen und die Bedeutung einer solchen Sicht selbst festlegen und jeden Befund einer Sicht zuordnen. Sichten ermöglichen es später, die Ergebnisdarstellung auf einen ausgewählten Aspekt zu fokussieren. Beispiele für mögliche Sichten sind etwa 'Benutzerauthentisierung', 'Logging' oder auch 'Accounting'.

Neben Anmerkungen und Sichten können auch *Verbesserungsvorschläge*, *Verweise* zu anderen Befunddaten oder auch zu externen Quellen sowie Daten-Abzüge (*Dumps*) als Befunde abgelegt werden. Für alle diese Befundtypen steht dem Handler-Programmierer eine einfache Programmierschnittstelle zur Verfügung.

Alle Befundarten können jeweils mit unterschiedlichem Geltungsbereich in der Datenbank abgelegt werden: Befunde zu einzelnen Konfigurationsklauseln, zu einem IOS-Klausel-Kontext (Configuration Mode), oder zu einer ganzen Sicht.

3 Praktischer Einsatz des Prüfwerkzeugs

3.1 Aufgabenspezifische Darstellungen

CROCODILE liefert verschiedene Darstellungen des Prüfergebnisses, sowohl Übersichten und Statistiken als auch themenbezogene Ausschnitte in Form der beschriebenen Analyse-Sichten. Alle Darstellungen können mit Hyperlinks zu verwandten Darstellungen und weiterführenden Informationen angereichert werden, etwa einem Verweis auf die passende Seite der Cisco IOS Command Reference im Internet. Ausgehend von der Übersicht kann der Anwender immer tiefer in spezifische Aspekte hinein navigieren, wobei sich der Ausschnitt verkleinert und der Detailreichtum zunimmt. Abbildung 2 zeigt zwei Beispiele für solche Darstellungen, wobei sich die untere auf die Konfiguration als Ganzes und die obere auf eine spezifische Zeile bezieht (hier: Zeile 90).

3.2 Blackset- und Whiteset-Analysen

Eine Besonderheit von CROCODILE ist seine Fähigkeit, Access-Control-Listen (ACLs) symbolisch darzustellen und weiterzuverarbeiten. Unter IOS werden ACLs als geordnete Folgen von ACL-Klauseln spezifiziert — ein flexibler und kompakter, aber leider recht fehleranfälliger Formalismus: Zwar lässt sich für ein gegebenes Paket leicht feststellen, ob es von der ACL abgewiesen oder akzeptiert wird. Betrachtet man aber alle denkbaren Pakete, so scheitert eine fallweise Betrachtung an der Unzahl der möglichen Attributkombinationen, wie etwa Protokoll, Ursprung, Ziel und Verbindungsstatus. In der Praxis geht so der Überblick über die Durchlässigkeit der konfigurierten ACLs leicht verloren.

Eine symbolische ACL-Beschreibung ermöglicht es nun, die Menge aller möglichen Datenpakete zu charakterisieren, die von einer ACL abgewiesen (Blackset) bzw. akzeptiert (Whiteset) werden. Mittels der symbolischen Blackset- und Whiteset-Charakterisierungen kann die Filterwirkung der ACL übersichtlich dargestellt und wi-

dersprüchliche oder redundante („tote“) Klauseln erkannt werden. Damit lässt sich die Einhaltung von Vorgaben für die „Durchlässigkeit“ des Routers formal beweisen, wobei die tatsächliche Filterwirkung unabhängig von der Struktur der beteiligten ACLs symbolisch berechnet werden kann. Es findet also keine Prüfung auf bloße syntaktische Übereinstimmungen statt, sondern eine genaue inhaltliche Analyse der Konfigurationsvorgaben. Details zum Berechnungsverfahren finden sich in [PS03a].

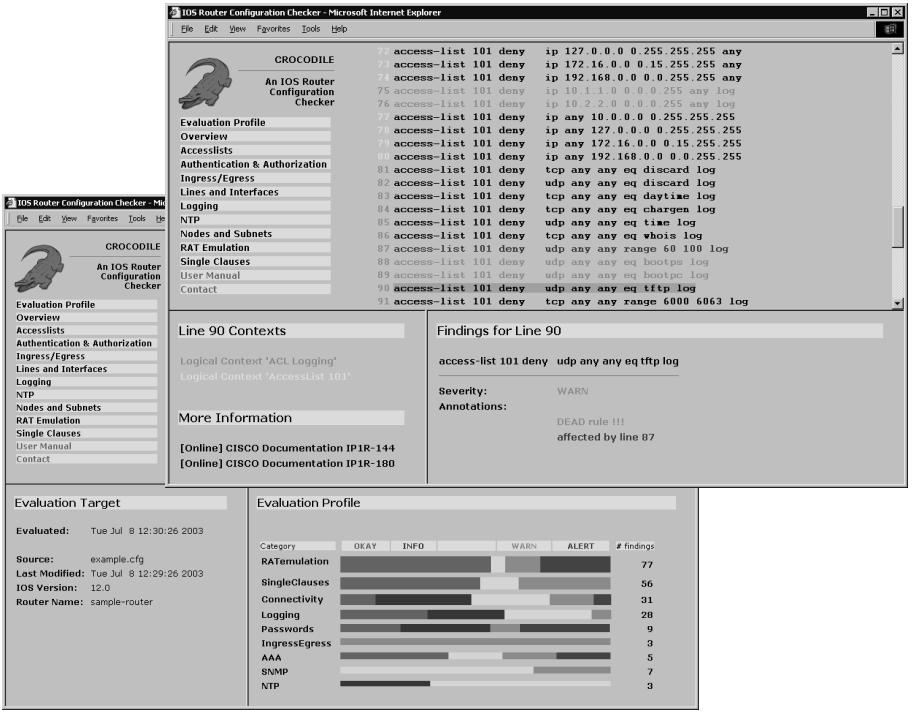

Abbildung 2: Zwei CROCODILE-Darstellungen verschiedener Detaillierung

3.3 Typische aufgedeckte Schwachstellen

Das Werkzeug findet mit den derzeitigen Prüfmodulen u.a. folgende typische Schwachstellen (siehe dazu auch die Online-Demonstration auf der CROCODILE Homepage):

- potentiell gefährliche Dienste, die eingeschaltet sind, und nützliche Sicherheitsfunktionen, die ausgeschaltet sind (z.B. Passwortschutz, Authentisierung)

- schwach verschlüsselte Passwörter, die auch stark verschlüsselt werden und so die starke Verschlüsselung ungewollt offen legen

- unvollständige Konfiguration komplexer Router-Funktionen (z.B. Authentisierung über RADIUS, ohne auch einen RADIUS Server anzugeben)

- Schutz eines Interface durch eine undefinierte ACL (so dass das Interface effektiv ungeschützt ist)

- ACLs, die zwar definiert, aber nie benutzt werden (wahrscheinlich als Folge eines Tippfehlers; möglicherweise existiert die falsch benannte ACL tatsächlich, bietet aber nicht den hier passenden Schutz)

- ACLs, die wirkungslose „tote" Klauseln enthalten (keine unmittelbare Gefahr, aber ein Hinweis, dass der Router-Administrator nicht alle Implikationen der Konfiguration verstanden hat)

- ein Ingress/Egress-Interface (d.h. eine Verbindung zu einem fremden, nicht vertrauenswürdigen Netz), das die vom Benutzer definierten minimalen Durchlässigkeits- (Whiteset) und Filterungsanforderungen (Blackset) verletzt

- ein unangemessen hoher Schwellenwert für das Logging (z.B. so hoch, dass unbedingt beachtenswerte ACL-Ereignisse nicht geloggt werden)

Die meisten dieser Prüfbefunde erfordern die Analyse der kombinierten Wirkung mehrerer Konfigurationsklauseln. Dies kann nicht durch bloßes zeilenweises Pattern-Matching (also eine Prüfung der strukturellen Äquivalenz zu Klausel-Vorgaben auf syntaktischer Ebene) erreicht werden, sondern erfordert eine Analyse der Konfiguration als Ganzes.

3.4 Differentielle Darstellung

Eine besondere Form der Befundaufbereitung ist die differentielle Darstellung zweier Prüfbefunde. Sie dient dazu, bei der Wiederholungsprüfung einer modifizierten Router-Konfiguration in den verschiedenen Sichten jene Befunde auszublenden, die seit der letzten Prüfung unverändert geblieben sind. Eine differentielle Darstellung ist für die Praxis besonders wichtig, denn ein Prüfer wird oft nicht auf alle Befunde des Prüfwerkzeugs reagieren, sondern einige Konfigurationsmerkmale des Routers trotz Befundmeldungen so belassen, wie sie sind. Bei erneuter Prüfung wäre es dann äußerst lästig, immer wieder mit den gleichen, irrelevanten Prüfbefunden behelligt zu werden. Beispiele für differentielle Analysesichten finden sich auf der CROCODILE Homepage. Der Anwender erkennt durch farbliche Hervorhebung sofort, wo die Konfiguration in der neuen Version geändert wurde und ob die Änderungen zum Guten oder zum Schlechten waren.

3.5 Verfügbare Prüfmodule

Mit den derzeit verfügbaren Prüfmodulen kann bereits eine beachtliche Testabdeckung erzielt werden. Zum einen decken die vorhandenen Module Prüfbereiche von hohem allgemeinen Interesse ab, so etwa die Logging-Einstellungen, die Vorgaben für Authentisierung, Autorisierung und Accounting (AAA) oder die Einstellungen für das Network

Time Protocol[1]. Zum anderen verfügt CROCODILE aber auch über zwei Prüfmodule mit frei konfigurierbaren Prüfkriterien: `IngressEgress` und `SingleClauses`.

Das Prüfmodul `IngressEgress` verifiziert benutzerdefinierte Vorgaben für die beabsichtigte Filterwirkung von ACLs oder Interfaces in ausgehender oder eingehender Übertragungsrichtung. Dazu können die Datenpakettypen deklariert werden, die den Filter mindestens passieren können müssen sowie jene Datenpakettypen, die mindestens zurückgehalten werden müssen (Whiteset und Blackset im Sinne von Abschnitt 0). Das Prüfmodul analysiert, welcher Teil des Whitesets die Filtermechanismen von ACLs oder Interfaces tatsächlich überwindet, und welcher Teil des Blacksets tatsächlich ausgefiltert wird. Diskrepanzen zwischen Vorgabe und Befund werden genau protokolliert.

Das Prüfmodul `SingleClauses` ermöglicht benutzerdefinierte Prüfvorgaben für das Vorhandensein von IOS-Konfigurationsklauseln. Es können beliebige Patterns hinterlegt werden, und für jedes Pattern wird vorgegeben, wie (1) das Auftreten des Patterns, (2) sein Fehlen bzw. (3) sein Auftreten in negierter Form in der Konfiguration zu bewerten ist. `SingleClauses` ist in der Lage, die IOS-Konfigurationsmodi zu berücksichtigen und Patterns nur im Kontext frei wählbarer Modi — zum Beispiel nur in Bezug auf Interface-Deklarationen des Typs 'Ethernet' — zu beachten. Der Nachteil dieses Moduls besteht darin, dass aufgrund der Prüfvorgaben nur isolierte Klauseln betrachtet werden können; klauselübergreifende Zusammenhänge lassen sich damit nicht analysieren. Der große Vorteil dieser Beschränkung liegt in der Einfachheit, mit der sich (einfache) Prüfvorgaben deklarieren lassen: Ohne Programmieren können weite Prüfbereiche erschlossen werden. Wie die Erfahrung zeigt, kann `SingleClauses` in der Praxis erheblich zu einer hohen Testabdeckung beitragen. Allerdings benötigt man für eine gründliche Analyse von Konfigurationszusammenhängen darüber hinaus noch tiefergehende Prüfmodule, wie CROCODILE sie zum Beispiel mit `IngressEgress`, `SNMP` oder `AAA` bietet.

4 Vergleichbare Prüfwerkzeuge

Für die Sicherheitsanalyse von Router-Konfigurationen existiert bisher nur wenig Software-Unterstützung. Bemerkenswerterweise lassen auch die Router-Hersteller den Anwender hier allein. Das bisher wichtigste Werkzeug auf diesem Gebiet ist RAT („Router Audit Tool“); ein wenig bekanntes Werkzeug von Equant Communications kommt noch hinzu, sowie zwei Werkzeuge, die eigentlich der Analyse von Firewalls dienen.

4.1 RAT

RAT Version 2.0 wurde von mehreren Partnern — darunter die U.S. National Security Agency (NSA), UUNET und das SANS Institute — gemeinsam entwickelt. Das Werkzeug wird kostenlos vom Center for Internet Security im Internet angeboten [Ci03].

[1] Eine genauere Beschreibung aller Prüfmodule findet sich in [PS03b].

RAT überprüft Router-Konfigurationen anhand konfigurierbarer Prüfregeln gemäß NSA-Empfehlungen [Ns03]. Eine RAT-Prüfregel besteht aus einem Pattern, das entweder als vorgeschrieben oder als verboten deklariert ist. Je nachdem, ob ein entsprechendes Pattern im Konfigurationstext auftritt oder nicht, gilt die Prüfregel als bestanden (`pass`) oder als nicht bestanden (`fail`). Wie bei CROCODILE können Patterns auch relativ zum Kontext bestimmter IOS-Konfigurationsmodi überprüft werden. Allerdings verzichtet RAT darauf, Patterns auch einen individuellen Pattern Handler zuzuordnen.[2] Anders als bei CROCODILE sind daher Prüfregeln, die sich auf das Zusammenspiel mehrerer Router-Konfigurationsklauseln beziehen, nicht darstellbar.

Die Analysefähigkeiten von RAT entsprechen somit genau der Funktionalität des CROCODILE-Prüfmoduls `SingleClauses`. Allerdings sind RAT-Prüfregeln mit zusätzlichen Attributen versehen, zum Beispiel einem Regelgewicht, einer Klartextbeschreibung und Erläuterungen sowie gegebenenfalls einem Korrekturvorschlag. `SingleClauses` verzichtet bewusst auf eine solche Attributierung seiner Prüfregeln, um den Aufwand zum Spezifizieren benutzerdefinierter Prüfanforderungen gering und die Regelsyntax einfach zu halten. CROCODILE verfügt aber über ein RAT Emulationsmodul und profitiert damit uneingeschränkt von RATs aufwendig dokumentierten Regelmengen, insbesondere von den dort eingebetteten Links auf weiterführende Informationen: Das Prüfmodul `RAT-emulation` liest RAT-Prüfregelsätze und erzeugt daraus Original-RAT-Prüfreports als zusätzliche CROCODILE-Sichten. Dies bietet einen Migrationspfad für Anwender, die bisher RAT verwendet haben. Allerdings steigern die von RAT beigesteuerten Prüfregeln nicht die Prüftiefe von CROCODILE, sondern ergänzen nur die „More Information"-Abschnitte in den vorhandenen CROCODILE-Sichten.

4.2 Equant Router-Werkzeug

Es gibt nur spärliche Informationen[3] über das Werkzeug von Equant [VL02], das auch nicht extern verfügbar zu sein scheint. Das Tool bietet zahlreiche kleinere Tests, die in Umfang und Tiefe mit dem `SingleClauses`-Modul in CROCODILE vergleichbar sind; seine bemerkenswerteste Fähigkeit ist aber die Analyse der Wechselwirkungen der Klauseln innerhalb einer ACL. In dieser Hinsicht geht es über die Einzelzeilen-Analyse von RAT hinaus: Innerhalb einer ACL vergleicht das Werkzeug alle möglichen Klauselpaare und prüft, ob sie widersprüchlich sind oder ob eine der Klauseln redundant ist. Eine vollständige Berechnung des Blacksets und Whitesets einer ACL, d.h. eine Betrachtung über Paare von ACL-Klauseln hinaus, erfolgt allerdings nicht. Deshalb sind Analysen wie in Abschnitt 0 beschrieben auch nicht möglich. Anders als CROCODILE erkennt das Equant-Tool nur solche Widersprüche und Redundanzen, die sich aus nur zwei Klauseln ergeben und nicht erst durch größere Klauselmengen entstehen.

---

[2] An Stelle eines Patterns kann allerdings ein sogenanntes Callout — d.h. eine beliebige Perl-Prozedur — angegeben werden, um eine pass/fail-Entscheidung zu treffen. RAT umfasst jedoch nur rudimentäre Callouts, und Callouts können generell auch keine über pass/fail hinausgehende Analyseergebnisse melden.

[3] Wir haben versucht, mit den Autoren Kontakt aufzunehmen, aber keine Antwort erhalten.

4.3 Firewall-Prüfwerkzeuge

Firewall-Prüfwerkzeuge befassen sich mit der Analyse von ACLs und Routing-Tabellen, einem Aspekt, der nur eine unter vielen Facetten des CROCODILE Tools ausmacht.

Der Lumeta Firewall Analyzer [Wo01] simuliert das Filter-Verhalten einer Firewall für alle IP-Pakete und erzeugt eine HTML-Ausgabe, die zeigt, welche Pakete eine gegebene Firewall-Konfiguration zulässt oder nicht. Die LFA-Ausgabe zeigt die zugelassenen und abgewiesenen Pakete nach Diensten, Rechnern, oder Rechner-Gruppen aufgeschlüsselt an, jeweils in eingehender und ausgehender Richtung. Das ist konzeptuell recht ähnlich zu CROCODILEs ACL-Analysefunktionen, aber noch ansprechender präsentiert.

Während CROCODILE Blackset und Whiteset explizit konstruiert, scheint LFA sein Bild vom Filterverhalten durch fortgesetztes Probieren verschiedener Kombinationen von Quelle, Ziel und Dienst aufzubauen, wobei auch „Jokerzeichen" verwendet werden. Tote Klauseln wie in CROCODILE können so allerdings nicht entdeckt werden. Ein detaillierter Vergleich der beiden Ansätze, vor allem hinsichtlich Geschwindigkeit, Vollständigkeit und Allgemeingültigkeit ist auf Grundlage der bisher verfügbaren Informationen leider nicht möglich.[4] Die Verfügbarkeit von LFA ist unklar – das Werkzeug wird auf der Lumeta Website nur erwähnt, aber in keiner Weise vermarktet.

Ein weiteres Werkzeug zur Analyse von ACLs beruht auf der Übersetzung von ACL-Klauseln in logische Formeln einer Prolog-ähnlichen Sprache, auf deren Grundlage dann eine Inferenzmaschine Fragen zur Filterwirkung dieser ACLs beantwortet [EZ01]. Die aus den ACLs gewonnenen Formeln werden noch mit allgemeinen logischen Regeln angereichert, die typische Routing-Konzepte wie die Zugänglichkeit bestimmter Dienste oder Subnetze in Logik codieren, so dass Anfragen auch mit diesen Konzepten formuliert werden können. Der Anwender kann auch selbst zusätzliche logische Regeln programmieren, womit das Werkzeug ähnlich wie CROCODILE erweiterbar ist.

Auch wenn das Werkzeug insgesamt noch recht unreif erscheint und bisher offenbar keine Erfahrungen mit komplexen ACLs aus der Praxis vorliegen, erscheint der Ansatz doch konzeptuell sehr interessant. Die Formulierung anwendungsspezifischer Regeln in einer Prolog-artigen Sprache mag für manche Probleme natürlicher als in Perl sein, ist aber auch bedeutend weniger flexibel, da die Regeln nicht mehr auf der ursprünglichen Konfigurationsbeschreibung operieren, sondern auf vorgegebenen logischen Atomen. Kritisch ist allerdings die Performance eines inferenzbasierten Ansatzes zu hinterfragen; die Autoren machen hierzu keinerlei Angaben, und offenbar haben sie das Werkzeug bisher nur mit vergleichsweise einfachen ACLs (aus 10-20 Klauseln) getestet. Auch die nur kurz erwähnte Zerlegung überlappender ACL-Klauseln in disjunkte Teile wirft Fragen nach der Performance auf (siehe auch [PS03a]).

[Ha99] stellt ein prototypisches Werkzeug für interaktive Anfragen zur Filterwirkung einer ACL vor. Auch mit diesem Werkzeug können typische Fragen nach zugelassenen oder abgewiesenen IP-Paketen beantwortet werden. Redundante (nicht aber wider-

[4] Eine Anfrage an den Entwickler von LFA blieb bisher unbeantwortet.

sprüchliche) Klauseln werden ebenfalls angezeigt. Weiterhin kann das differentielle Blackset und Whiteset zweier Versionen einer ACL berechnet werden. Eine Erweiterung durch den Anwender auf höher abstrahierte Konzepte wie in [EZ01] oder CROCODILE ist nicht möglich. Die verwendete Bitvektor-basierte Repräsentation von ACLs macht die Handhabung von Anfragen und Ergebnissen kompliziert; möglicherweise ist dieser Ansatz daher eher interessant für die Kompilation von ACL-Klauseln durch einen Router in eine effizienter auswertbare Form, als für einen menschlichen Benutzer, der eher auf der Ebene von Subnetzen und Diensten als auf der von Bitvektoren denkt.

5 Zusammenfassung und Ausblick

Mit CROCODILE haben wir ein einfach erweiterbares, flexibles Framework vorgelegt, das den Experten bei der Sicherheitsanalyse von Router-Konfigurationen unterstützt. Es entlastet von mühsamer Kleinarbeit und versieht alle Befunde mit Einschätzungen ihrer Sicherheitsrelevanz sowie vielen weiteren nützlichen Informationen. Das Werkzeug bereitet die Vielfalt der Informationen zu themenspezifischen Auswahlen auf.

Wir haben bisher eine Reihe von Konfigurationen untersucht, die von zwei externen Anwender-Organisationen (beide mit speziell geschultem Personal) stammen. In den meisten dieser Konfigurationen fand CROCODILE zumindest Ungereimtheiten — oft sogar gravierende Fehler wie das Referenzieren undefinierter ACLs! Offenbar sind Konfigurationsmängel eher die Regel als die Ausnahme, und Sicherheitslücken bleiben oft über Monate unentdeckt. Dass wir bereits mit wenigen Prüfmodulen Schwachstellen entdecken, die Fachpersonal übersehen hat, bestärkt uns in unserem Ansatz.

CROCODILE enthält einige innovative Elemente, insbesondere bei den Funktionen, die auf der Gesamtsicht der Router-Konfiguration beruhen, sowie bei der Blackset/Whiteset-Analyse. CROCODILE kann den Experten aber nicht ersetzen — allein die Anpassung der konfigurierbaren Prüfregeln an lokale Gegebenheiten erfordert Wissen, das ein Werkzeug nicht mitbringen kann. Die Interpretation des Prüfbefunds wird immer den individuellen Sachverstand eines Netzwerkspezialisten erfordern. Vollautomatische Korrekturen waren daher nie ein Entwicklungsziel.

CROCODILE wird laufend fortentwickelt, einschließlich der Entwicklung weiterer Prüfmodule. Der Ausbau erfolgt bedarfsgesteuert. Wichtiger als ein umfangreicher, festgelegter Prüfmodul-Fundus ist die durch Vererbungsmechanismen unterstützte, leichte Erweiterbarkeit des CROCODILE Frameworks. CROCODILEs Anwendungsprogrammierschnittstelle unterstützt die Erweiterung der Prüffähigkeiten u.a. durch

- eine saubere „Plug-In“-Schnittstelle für neue Prüfmodule, die jedem Modul nur die relevanten Informationen anbietet, durch BNF Patterns spezifiziert;

- eine interne Datenbank für Prüfresultate, Querverweise, Korrekturvorschläge, und andere nützliche Informationen in strukturiertem Format, mit einer einfachen, aber flexiblen Schnittstelle zu den Prüfmodulen;

- einen Hypertext-Reportgenerator, der den Inhalt der Datenbank übersichtlich präsentiert und zusammengehörige Befunde in logische Sichten gruppiert;

- eine differentielle Darstellung, die automatisch für jedes Modul verfügbar ist – sogar für das RAT-Emulationsmodul mit seinen artfremden Prüfregeln.

Durch die Definition geeigneter Pattern und Handler lässt sich der CROCODILE-Ansatz auch auf die Prüfung anderer textbasierter Konfigurationsbeschreibungen ausdehnen – auf Router anderer Hersteller, Firewalls oder auch auf allgemeine Netzdienste.

Die Homepage unter www.iese.fraunhofer.de/crocodile zeigt den aktuellen Entwicklungsstand. Dort findet sich auch ein ausführliches Handbuch sowie eine Online-Demonstration.

Literaturverzeichnis

[Ci99] Cisco Systems Inc.: Cisco IOS 12.0 Configuration Fundamentals. Cisco Press, 1999

[Ci03] The Center for Internet Security: Router Audit Tool. März 2003
http://www.cisecurity.org/

[EZ01] Eronen, P.; Zitting, J.: An expert system for analyzing firewall rules. Proc. 6th Nordic Workshop on Secure IT Systems (NordSec2001), Copenhagen, Denmark, pp. 100-107, November 2001
http://www.cs.hut.fi/~peronen/publications

[GS01] Groß, S; Schwarz, R.: Ein Werkzeug zur Sicherheitsüberprüfung von Cisco IOS Routerkonfigurationen. IESE Report Nr. 078/01D, Dezember 2001

[Ha99] Hazelhurst, S.: Algorithms for Analysing Firewall and Router Access Lists. Tech. Report TR-WitsCS -1999-5, Dep. of Computer Science, University of the Witwatersrand, South Africa, July 1999
http://citeseer.nj.nec.com/hazelhurst99algorithm.html

[Ns03] National Security Agency: Router Security Configuration Guide. Februar 2003
http://www.nsa.gov/snac/cisco/index.html

[PS03a] Peine, H.; Schwarz, R.: A Router Security Tool with Powerful Analysis of Blacksets and Whitesets. Zur Veröffentlichung eingereicht, Mai 2003

[PS03b] Peine, H.; Schwarz, R.: A Multi-View Tool for Checking the Security Semantics of Router Configurations. Zur Veröffentlichung eingereicht, Juni 2003

[Wo01] Wool, A.: Architecting the Lumeta Firewall Analyzer. 10th USENIX Security Symposium, Washington, D.C., August 13-17, 2001
http://www.usenix.org/events/sec01/wool.html

[VL02] Valois, D.; Llorens, C. : Identification of Security Holes in Router Configurations. 14th Annual Computer Security Incident Handling Conference (FIRST 2002), Hilton Waikoloa Village, Hawaii, June 24-28, 2002
http://www.first.org/events/progconf/ 2002/d4-04-valois-paper.pdf