Schutz verteilter Intrusion-Detection-Systeme gegen Denial-of-Service-Angriffe[1]

Marko Jahnke
Forschungsgesellschaft für angewandte Naturwissenschaften e.V. (FGAN)
Neuenahrer Str. 20
53343 D-Wachtberg
jahnke@fgan.de

1 Einführung

Ein geschütztes System kann immer höchstens so sicher sein wie das Schutzsystem selbst. Dies gilt in Zeiten immer häufiger auftretender Angriffe gegen Computersysteme auch und besonders für Intrusion-Detection-Systeme (*IDS*), deren Aufgabe darin besteht, Anzeichen für schadhafte Vorgänge in Computersystemen und –netzwerken aufzuspüren und entsprechende Gegenmaßnahmen einzuleiten. Wird das IDS an der bestimmungsgemäßen Ausführung seiner Aufgabe gehindert, so spricht man von einem Denial-of-Service-Angriff (*DoS*).

Um diese Aufgabe erfüllen zu können, sammeln IDS mit Hilfe von *Sensoren* Messdaten, die notwendig sind, um Zustandsparameter des zu überwachenden Systems zu bestimmen. Diese Daten können unterschiedlichster Natur sein; es kann sich beispielsweise um die Inhalte von Protokollpaketen auf Netzwerkverbindungen handeln, aber auch um numerische Parameter, wie die CPU-Auslastung oder die Anzahl der angemeldeten Benutzer eines Systems. Aus Gründen der Datenreduktion sowie der fehlenden Uniformität der Datenmodelle geht man im allgemeinen von sogenannten *Ereignismeldungs-generierenden* Sensoren aus, die nicht die rohen Sensordaten, sondern Ereignismeldungen (Events) an andere Systemkomponenten übermitteln, wenn bestimmte Bedingungen für die Daten eingetreten sind (z.B. wenn numerische Parameter in bestimmten Intervallgrenzen liegen).

Viele verteilte IDS verwenden eine zentralisierte Architektur mit verschiedenen Arten von Sensoren, wobei lokale Einheiten (meist als *Relays* oder *Agenten* bezeichnet) die auf einem Computersystem oder einer Netzwerkkomponente auftretenden Sensor-Ereignisse sammeln, vorverarbeiten und über das Netzwerk an eine zentrale Einheit (*Manager* oder *Konsole*) versenden. Diese analysiert und speichert die empfangenen Informationen, die optional – entsprechend aufbereitet – dem Sicherheitsadministrator zur visuellen Inspektion präsentiert werden können. Auch das zentrale Management und die Fernkonfiguration der verteilten Komponenten sind meist in die Konsoleneinheit integriert.

Die meisten verfügbaren Implementationen sind Software-basiert; nur eine geringe Anzahl von IDS-Produkten besteht dagegen aus vorkonfigurierter, dedizierter Hardware. Im weiteren

1 Erschienen in: Tagungsband des 10. DFNCERT/PCA-Workshops „Sicherheit in vernetzten Systemen“, Februar 2003

Verlauf wird im allgemeinen von Software-implementierten Agenten und Sensoren ausgegangen, die ihren Dienst ressourcenschonend im Hintergrund auf zu überwachenden Endsystemen (Desktop- oder Serversysteme) verrichten.

Der Rest dieses Beitrages gestaltet sich wie folgt. Abschnitt 2 beschreibt einige verwandte Ansätze, um IDS gegen bestimmte Arten von Angriffen zu schützen. In Abschnitt 3 werden verschiedene Aspekte von DoS-Angriffen beschrieben und die Konsequenzen für das zu schützende System diskutiert. Vorschläge für Gegenmaßnahmen werden in Abschnitt 4 gemacht. Die Realisierung einiger dieser Gegenmaßnahmen im Rahmen einer prototypischen Implementierung einer verteilten IDS-Infrastruktur wird in Abschnitt 5 beschrieben.

2 Verwandte Arbeiten

Schutzmechanismen für IDS wurden bereits in zahlreichen Publikationen behandelt. Takada und Koike [16] schlugen versteckte Kopien für Audit-Logfiles vor, um nichtautorisierten Zugriff von Angreifern zu verhindern. Allerdings bestehen zahlreiche Einwände dagegen, das Verstecken von Informationen in Schutzsystemen einzusetzen.

Im Rahmen der Diskussion um mobile Agenten wurde vorgeschlagen, dass IDS-Agenten zwischen den zu überwachenden Systemen „wandern“ sollten, um einer Erkennung und Zerstörung durch einen Angreifer zu entgehen (z.B. in [10], [1]). Auch wenn dies in der Theorie eine durchaus interessante Idee darstellt, sind funktionstüchtige Implementationen, die robust genug für einen industriellen Einsatz sind, kurzfristig nicht zu erwarten.

Holz et al. [8] beschreiben eine Strategie, nach der Agenten die Analyse von Ereignismeldungen in Überlastsituationen an andere Agenten delegieren können. Die Erkennung und Behandlung solcher Situationen ist aber insbesondere für den oft wünschenswerten integrierten Einsatz von Dritthersteller-Software (*Third-Party-Komponenten*) im IDS nicht verwendbar.

Anstatt auf DoS-Angriffe zu reagieren, verwenden verschiedene Ansätze (so z.B. der von Wolthusen [19], [20]) Präventivmaßnahmen, die durch die Modifikation Betriebssystem-interner Funktionalitäten die Einhaltung der Sicherheitspolicies erzwingen können. Privilegierte Operationen - wie der Zugriff auf bestimmte Dateien, auf Netzwerkressourcen oder Prozesse - können nur nach einer zusätzlichen Authentifizierung erfolgen; Verstöße werden direkt an den Sicherheitsadministrator gemeldet. Mittel- und langfristig sind derartige Ansätze sicherlich wegen des umfassenden Sicherheitskonzeptes das Mittel der Wahl, aber kurzfristig wegen des beträchtlichen Aufwandes nicht großflächig zu realisieren. Es existiert eine Vielzahl bereits im Einsatz befindlicher, mehr oder minder unsicherer Systeme, die es schnell bestmöglich zu schützen gilt.

3 Denial-of-Service gegen verteilte IDS

Wie eingangs erwähnt, ist die Überlebensfähigkeit eines Schutzsystems von entscheidender Bedeutung. Möglichkeiten zur Eigensicherung müssen bereits in der Entwurfsphase integraler Bestandteil der Architektur werden. Eine möglichst systematische Aufarbeitung von Aspekten, die bei Denial-of-Service-Angriffen eine Rolle spielen, trägt wesentlich zu einer effizienten Entwicklung von Gegenmaßnahmen bei.

Man kann bei DoS auf verteilte Systeme zunächst bezüglich des Ortes unterscheiden, an dem der Angriff Wirkung zeigt. Zum einen können die zentralen Komponenten des Systems (beispielsweise ein Konsolen-Server und seine Unterkomponenten) betroffen sein, zum

anderen aber auch die auf den zu überwachenden Plattformen verteilten Komponenten (z.B. Agenten, Relays, Sensoren). Als nicht zu vernachlässigender Angriffspunkt gelten aber auch die Kommunikationsverbindungen der Komponenten untereinander; hier ist wiederum zwischen der Netzwerk- und der Inter-Prozess-Kommunikation zu unterscheiden.

Auch die Methoden für DoS gegen ein IDS sind vielfältig. Die *Manipulation* sowie das *Mithören von Kommunikationsverbindungen* stellen eine Möglichkeit dar, das IDS an seiner bestimmungsgemäßen Ausführung seiner Aufgabe zu hindern (Streng genommen fällt das Mithören zwar nicht in diese Kategorie, trägt aber zur effektiveren Durchführung von DoS-Angriffen durch Ausgleich des Informationsdefizites bei).

Desweiteren ist es möglich, einzelne oder die Gesamtheit der Prozesse eines IDS zu terminieren, was die Beendigung der Ausführung der entsprechenden IDS-Komponenten zur Folge hat (*Prozessterminierung*).

Eine Herbeiführung von *Überlastsituation*en für die IDS-Komponenten hat ebenfalls zur Folge, dass die bestimmungsgemäße Aufgabe nicht oder nur verzögert erfüllt werden kann. Besonders schwierig zu handhaben ist dies, wenn Third-Party-Komponenten in das IDS integriert wurden, da für sie meist kein Erkennungs- und Reaktionsmechanismus für Überlast-situationen implementiert ist.

Eine weitere Möglichkeit besteht in der *Blendung* von Komponenten, d.h. deren Manipulation derart, dass die Komponente den übrigen Komponenten signalisiert, ordnungsgemäß zu funktionieren, während sie tatsächlich bereits ihrer vollen Handlungsfähigkeit beraubt worden ist. Man unterscheidet zwischen zwei Alternativen:

- *Blendung durch manipulierte Sensor-Daten*
  Ist die Art der Blendung eines IDS-Sensors ausschließlich von der Beschaffenheit der erfassten Sensordaten abhängig, so ist sie hochgradig sensorspezifisch und es kann offensichtlich keine generische Gegenmaßnahme ergriffen werden. Als Beispiel stelle man sich einen Netzwerksensor vor, der aufgrund von Spezifikationslücken in den Internet-Protokollen (z.B. TCP-Flags) Pakete verwirft, während das angegriffene Zielsystem das Paket im Datenstrom belässt (vg. [13]).

- *Nichtautorisierte Rekonfiguration*
  Werden bestimmte Parameter für die Ausführung einzelner IDS-Komponenten ohne Wissen des restlichen Systems verändert, so kann diese Situation einer Blendung entsprechen. Als Beispiel ist das Löschen einzelner Einträge einer lokal verwalteten Signaturdatenbank eines Logfile-Sensors zu nennen, der in diesem Falle möglicherweise hochkritische Systemmeldungen ignorieren würde.

Um dieser Problematik Herr zu werden, müssen existierende Möglichkeiten für Gegenmaßnahmen in ein IDS integriert bzw. neue Möglichkeiten entwickelt, implementiert und getestet werden. Dies muss auf allen Ebenen der Informationsabsicherung (*Prävention, Detektion, Reaktion*) erfolgen. Es muss allerdings von vorneherein klar sein, welche Form und welchen Grad an Sicherheit (d.h. sowohl „Security“ als auch „Safety“) man erzielen kann und will. Aufgrund des in der betrachteten Implementation gewählten User-Space-Ansatzes ist bereits vorweggenommen, dass kein Schutz gegen Angreifer bestehen kann, die auf Betriebssystem-Ebene operieren.

Ein weiter wichtiger Aspekt ist die *Fehlertoleranz*. Es kann u.U. sein, dass sich ein Angriff auf das IDS in der gleichen Weise äußert, wie eine Fehlfunktion (z.B. Speicherzuordnungs-fehler mit Programmabbruch). Hier ist es unbedingt notwendig, Mechanismen vorzusehen, die eine Entscheidung über den vorliegenden Fall akkurat treffen, damit sich das IDS nicht selbst auf diese Weise mit Denial-of-Service-Angriffen außer Kraft setzt.

4 Gegenmaßnahmen gegen DoS

Betrachtet man zunächst nur die zentralen Einheiten verteilter IDS-Architekturen (d.h. Management- oder Auswertungs-Konsolen), so ist Redundanz eine sinnvolle Präventivmaßnahme gegen DoS. Kommerzielle IDS verfügen oft bereits über Backup-Systeme (als Hard- oder Software), die im Falle einer erkannten Fehlfunktion die Funktion des Primärsystems übernehmen, um einen *Single-Point-of-Failure* bereits auf Architekturebene zu vermeiden. Daher wird sich im folgenden auf die verteilten IDS-Softwarekomponenten beschränkt.

Zu den reaktiven Maßnahmen bei allen Arten erkannter DoS-Angriffe gehört die Generierung von Ereignismeldungen mit entsprechend hoher Priorität, die ggf. die sofortige Benachrichtigung des Sicherheitsadministrators zur Folge haben muss. Die im letzten Abschnitt vorgestellten DoS-Angriffe erfordern jedoch weitere spezifische Maßnahmen, die in den folgenden Unterabschnitten erläutert werden. Mögliche Gründe für Falschmeldungen werden ebenfalls angeführt.

4.1 Stören der IDS-Kommunikationsverbindungen

Prävention

Präventive Maßnahmen gegen das Stören der Netzwerkverbindungen sind zum einen kryptografischer Natur (Integritätssicherung, Authentifizierung der Kommunikationspartner), zum anderen aber auch die Wahl wechselnder Verbindungsparameter (z.B. TCP-Ports). Letzteres hat die folgenden beiden Effekte: Einerseits ist für den Angreifer nicht auf den ersten Blick ersichtlich, welches IDS seinen Dienst im Zielnetzwerk verrichtet (was bei fest zugeordneten TCP-Ports der Fall wäre); andererseits ist für den Angreifer nach einer bereits durchgeführten Terminierung der IDS-Kommunikation eine Re-Terminierung nicht unmittelbar möglich, ohne die neuen Verbindungsparameter zu bestimmen.

Manipulation der Inter-Prozess-Kommunikation kann nur durch kernintegrierte Mechanismen verhindert werden, denn sobald ein Angreifer in der Lage ist, beispielsweise den Inhalt von Shared-Memory-Bereichen zu verändern, ist ein im User-Space operierendes IDS machtlos.

Ein weiterer präventiver Ansatz ist die im Bereich fehlertoleranter Systeme immer wieder zum Einsatz kommende Redundanz – in diesem Falle also alternative Kommunikationskanäle oder -routen.

Detektion

Die Detektion terminierter Netzwerkverbindungen äußert sich vielfältig und ist u.a. vom verwendeten Betriebssystem und Netzwerk abhängig. Häufig werden entsprechende Betriebssystem-Signale an die betroffenen Prozesse ausgeliefert, auch das Signalisieren eines auf der Verbindung anliegenden, zu lesenden Datums der Länge 0 (bzw. EOF) kann ein Anzeichen für eine (möglicherweise unfreiwillig) beendete Verbindung sein.

Explizit in Kommunikationsprotokollen vorgesehene sogenannte *Keep-Alive-Nachrichten* stellen eine weitere Möglichkeit dar, da sie dem Kommunikationspartner in regelmäßigen Abständen u.a. signalisieren, dass die Verbindung noch mit ausreichender Qualität vorhanden ist. Bei der Detektion von Manipulationen des Datenstromes ist die Verschlüsselung ein wesentlicher Mechanismus, denn die Veränderung der Daten führt unweigerlich zu einer fehlschlagenden Entschlüsselungsoperation beim Empfänger.

Die Entscheidung, ob es sich um einen Angriff oder um eine Fehlfunktion handelt, gestaltet sich schwierig. Eine problemlose unmittelbare Wiederaufnahme der Verbindung kann zwar zum Eindruck verleiten, dass ein Netzwerkproblem vorlag, jedoch könnten in der Zwischenzeit weitere Aktivitäten eines Angreifers auf dem Zielsystem erfolgt sein. Methoden zur Bestimmung von Paketverlustraten, wie man sie in der Echtzeit-Datenübertragung verwendet, geben dagegen Aufschluss über die Wahrscheinlichkeit für eine netzwerkbedingte Störung.

Reaktion

Um die bestimmungsgemäße Aufgabe auch nach einer gestörten Netzwerkverbindung erfüllen zu können, ist eine unmittelbare Wiederaufnahme der Verbindung notwendig. Dabei ist es nur dann erforderlich, eine gestörte Verbindung zuvor abzubrechen, wenn entweder keine Synchronisation der Kommunikationspartner mehr möglich oder ein Angriff auf das System wahrscheinlich ist. Wie bei den Präventivmaßnahmen erwähnt, ist dann die Verwendung geänderter Verbindungsparameter angeraten.

Ist es jedoch nicht möglich, die Verbindung wieder aufzunehmen, so müssen lokale Reaktionsmechanismen zur Verfügung stehen, die einerseits alternative Kommunikationsmöglichkeiten zur Benachrichtigung des Administrators bieten (z.B. SNMP-Traps, SMTP, Pager, Drucker), andererseits aber auch – wenn möglich – eigenständige Maßnahmen zur Trennung des Angreifers vom System ergreifen (z.B. Rekonfiguration eines Paketfilters, Prozessterminierung). Allerdings können Fehlkonfigurationen sehr problematisch werden, so z.B. wenn das System einen legitimen Fernzugriff des Administrators als Angriff klassifiziert und den Zugriff unterbricht.

4.2 Prozessterminierung

Prävention

Bei den Maßnahmen präventiver Art gegen die Terminierung von verteilten IDS-Prozessen auf Endsystemen ist man bei einem User-Space-orientierten Ansatz sehr beschränkt. Hat der Angreifer entweder die Benutzerkennung, unter dem die IDS-Komponenten auf einem Zielsystem betrieben werden, – oder gar die des Administrators – erlangt, so können Anwendungen nicht verhindern, dass Prozesse terminiert werden. Dazu ist nur ein entsprechender kernintegrierter Audit-Mechanismus zur Überwachung von Prozessmanipulationen in der Lage.

Redundante Prozesse auf einem System eignen sich nicht als Präventivmaßnahmen, weil sie vom Angreifer (aufgrund von ähnlichen oder gleichen Parametern) identifiziert und ebenso terminiert werden können. Eine denkbare Alternative dazu könnte die Einbringung von Code in Fremdprozesse – vergleichbar mit Software-Viren – sein, der die Prozessüberwachung und etwaige Reaktionsmechanismen anstößt. Dieser Ansatz ist innerhalb des User-Space möglich und würde – insbesondere, wenn die Prozessüberwachung in unverzichtbare Prozesse des Betriebssystems integriert werden könnte – eine unerkannte Terminierung mit großer Sicherheit verhindern.

Detektion

Die Detektion von Prozessterminierungen ist durch Überwachung von Betriebssystem-Signalen möglich; allerdings nur, wenn der terminierte Prozess ein Kindsprozess des überwachenden Prozesses (im weiteren als *Observer* bezeichnet) ist oder über eine Prozessbindung (z.B. mit `ptrace()`) mit ihm verbunden ist. Zudem gibt es in der so

gebildeten Überwachungskette immer mindestens einen Prozess, der keinen Observer besitzt (*Top-Level-Observer*); seine Terminierung kann auf diese Weise dann nicht erkannt werden.

Auch die Beobachtung der Prozesstabelle des Betriebssystems kann zur Erkennung von Prozessterminierungen herangezogen werden; dieses Verfahren kann auch bei nicht voneinander abgespaltenen Prozessen verwendet werden, ist allerdings nicht effektiv portabel und erfordert systemspezifische Anpassungen.

Weiterhin sind Lösungen denkbar, die entgegen den ebengenannten von regelmäßig verschickten, nicht fälschbaren Keep-Alive-Nachrichten (vgl. Unterabschnitt „Stören der Kommunikationsverbindungen“) ausgehen, die das Vorhandensein der entsprechenden Prozesse signalisieren. Die Nicht-Fälschbarkeit muss durch ein nur den beiden Prozessen bekanntes, zur Laufzeit generiertes Datum erreicht werden. Bei eingesetzten Third-Party-Komponenten ist allerdings eine nachträgliche Integration eines solchen Mechanismus nicht oder nur schwer möglich.

Fehler bei der Erkennung von Prozessterminierungen können bei ungewollten Abbrüchen der Programmausführung auftreten (z.B. durch Speicherzuordnungsfehler), wenn der Grund für die Terminierung nicht ermittelt werden kann. Es ist selbstverständlich, dass die eingesetzten Komponenten eines IDS hinreichend fehlerfrei und exakt getestet sein müssen, um keine diesbezüglichen Fehler hervorzurufen. Da aber u.U. auch Third-Party-Produkte als Komponenten des IDS integriert sind, hat man möglicherweise keinen Einfluss auf derartige Fehler, sondern muss andere Wege finden, um Programmabbrüche zu verhindern (oder aber zu tolerieren).

Reaktion

Als primäre reaktive Maßnahme ist sicherlich der Neustart des terminierten Prozesses zu sehen, um den Betrieb wieder aufnehmen zu können. Dies ist für Prozesse möglich, die als Kindsprozesse des Observer-Prozesses gestartet werden; wurde aber der Top-Level-Observer terminiert, so kann – mangels Möglichkeit zur Erkennung dieses Sachverhaltes – ein Neustart durch andere IDS-Prozesse durchgeführt werden. Es kann jedoch ein automatischer Neustart des Top-Level-Observers durch den Initialprozess des Betriebssystems (z.B. init-Prozess unter Unix System V) veranlasst werden, was wiederum dessen intakte Konfiguration voraussetzt.

Eine mögliche – wenn auch erneut mit Vorsicht zu genießende – autonome Sekundärmaßnahme ist das Ausfindigmachen desjenigen Prozesses, der die Terminierung eingeleitet hat. Falls sich dieser mit hoher Sicherheit feststellen lässt, ist auch eine Gegenterminierung denkbar.

4.3 Überlastsituationen

Prävention

Ist es möglich, obere Schranken für Ein- und Ausgabegrößen von verteilten IDS-Komponenten festzulegen, so kann eine dementsprechende Überlast vermieden werden. Betrachtet man beispielsweise die Ereignismeldungs-verarbeitenden Einheiten eines IDS (z.B. Analysierer, Netzwerk-Kommunikation, Datenbank), so kann man Maximalzahlen für die zu verarbeitenden Meldungen pro Zeiteinheit definieren und auch leicht durchsetzen. Es besteht jedoch die Gefahr, dass auch die Bearbeitung besonders wichtiger Meldungen durch diese Maßnahme verzögert wird, d.h. es muss eine Datenreduktion durch Zusammenfassung redundanter oder ggf. auch nur korrelierter Daten so früh wie möglich im Datenfluss erfolgen.

Werden jedoch Third-Party-Komponenten in das IDS integriert, so ist Prävention gegen deren Überlastung generisch nur sehr schwer möglich. Unter Verwendung von Echtzeit-Betriebssystemen oder –erweiterungen sind Maximalgrößen für die den Prozessen zugeteilten Rechenzeitscheiben definierbar; herkömmliche Betriebssysteme bieten eine solche Funktionalität nicht.

Detektion

Eine Überlastsituation kann beispielsweise durch Messen von Ein- und Ausgabegrößen (z.B. Anzahl der Ereignismeldungen) für die einzelnen IDS-Komponenten erkannt werden. Auch das Einbringen von *Watchpoints* in den Ausführungscode, die den Zeitabstand zweier aufeinanderfolgender Durchläufe messen und bei abnormal großen Abständen entsprechende Gegenmaßnahmen anstoßen, ist denkbar. Auch hier ist man beim Einsatz von Third-Party-Komponenten stark eingeschränkt; eine Überwachung von Parametern der Prozesstabelle ist möglich, aber – wie erwähnt – nicht portabel.

Falschmeldungen können ohne weiteres entstehen, wenn durch Überlastung anderer Prozesse den IDS-Prozessen nicht mehr genügend Rechenzeit zur Verfügung gestellt wird.

Reaktion

Überlastsituationen sollten durch manuelles Eingreifen behandelt werden. Wenn festgestellt wird, dass der Überlastungseffekt ungewollt durch andere Prozesse entstanden ist, ist der dafür zuständige Administrator zu benachrichtigen.

4.4 Blenden durch manipulierte Sensordaten

Prävention

Da die Art des Angriffes in diesem Falle nur von der Art der Datenerfassung abhängt, und diese im internen Datenmodell des IDS nicht generell abgebildet werden kann, können keine generischen Präventivmaßnahmen von der IDS-Infrastruktur ergriffen werden. Hier muss entweder eine inhärente Immunisierung des Sensors durchgeführt oder durch externe Maßnahmen die Anfälligkeit des Sensors kompensiert werden. Betrachtet man das im vorangehenden Abschnitt erwähnte Beispiel des umgangenen Netzwerksensors, so bieten Netzwerkverkehr-bereinigende Einheiten (sog. Protocol Scrubber oder Traffic Normalizer, vgl. [7]) einen entsprechenden Schutz.

Detektion

Erkennen lassen sich Datenmanipulations-Blendvorgänge nur durch den Einsatz redundanter Sensoren, die komplementär arbeiten, d.h. gegenseitig Unzulänglichkeiten ausgleichen. Allerdings ist die Anzahl der hier denkbaren Anwendungsfälle sehr gering, da Sensoren, die einen vergleichbaren Erkennungsraum eines existierenden Sensors mit komplementären Eigenschaften abdecken, sehr selten sind (schon rein aus wirtschaftlichen Erwägungen). Eines der wenigen Beispiele ist das bereits genannte Ausnutzen von Lücken in Protokoll-spezifikationen; ein Unix-basierender und ein Windows-basierender Sensor, die ihre Daten unmittelbar vom TCP/IP-Stack des Betriebssystems beziehen, erfassen u.U. alle Fälle der Protokollauslegung und können so einen Täuschungsversuch gegen den jeweils anderen Sensor erkennen. Die zuvor genannten Präventivmaßnahmen stellen aber die bessere Alternative gegenüber der Detektion dar, sofern sie angewandt werden können.

Reaktion

Reaktive Maßnahmen auf Blendung durch manipulierte Sensordaten erfordern ein hohes Maß an manuellen Eingriffen des Sicherheitsadministrators, weil der potentielle Schaden sehr unterschiedlicher Natur sein kann. Nur in unkritischen Fällen sollte ein autonomes Handeln der verteilten IDS-Komponenten in Betracht gezogen werden, d.h. beispielsweise ein sauberes Herunterfahren des Betriebssystems von Computern, die nicht als „mission-critical“ anzusehen sind.

4.5 Blenden durch nichtautorisierte Rekonfiguration

Prävention

Einer nichtautorisierten Rekonfiguration von IDS-Komponenten ist zunächst durch eine restriktive Handhabung von Dateizugriffsrechten auf dem verwendeten Dateisystem vorzubeugen, sodass außer dem Benutzer, der das IDS betreibt, und dem Administrator niemand in die Lage versetzt werden kann, die verteilten IDS-Komponenten zu manipulieren, d.h. Dateien in der Weise zu verändern, dass der gewünschte Blendungseffekt eintritt. Der schreibende Zugriff auf die sogenannten *Prozessumgebungen* (ausführbare Programme, Konfigurationsdateien, Shared Libraries, statische Daten) darf also nur dem entsprechenden Benutzer gestattet werden. Damit es einem Angreifer unmöglich wird die Identität des IDS-Benutzer anzunehmen (zumindest, ohne dabei entdeckt zu werden), sind wiederum entsprechende kernintegrierte Mechanismen notwendig, die zusätzliche Autorisierungen für privilegierte Aktionen erfordern (vgl. Abschnitt 6).

Detektion

Die Detektion von veränderten Prozessumgebungen auf dem Dateisystem gestaltet sich relativ einfach. Da eine Manipulation der Prozessumgebung immer einen Verlust der Integrität nach sich zieht, gilt es, eine Integritätsprüfung für die entsprechenden Dateien durchzuführen. Schlägt die Integritätsprüfung fehl, so liegt mit sehr hoher Wahrscheinlichkeit eine nichtautorisierte Manipulation vor. Bekannte Tools wie Tripwire [17] oder AIDE [11] bewerkstelligen genau diese Aufgabe. Sie tun dies zwar sehr effizient, ihre auf dem lokalen Dateisystem abgelegten Datenbanken können jedoch ebenfalls korrumpiert werden.

Ein möglicher Grund für eine diesbezügliche Falschmeldung kann ein inkonsistentes Dateisystem auf dem lokalen Datenträger sein, das zum Fehlschlagen der Integritätsprüfung führen kann, wobei die Wahrscheinlichkeit dafür bei der Verwendung entsprechend robuster Dateisysteme vernachlässigbar gering sein dürfte. Eine andere Möglichkeit besteht in falsch konfigurierten Backup/Restore-Vorgängen, die zum Überschreiben der Prozessumgebungen führen.

Ein durch Integritätsprüfung von Prozessumgebungen nicht abgedeckter Aspekt ist die Manipulation von IDS-Prozessen zur Laufzeit, die auf verschiedene Arten bewerkstelligt werden kann. Durch Veränderung des Prozessabbildes im Arbeitsspeicher oder durch sogenanntes Injizieren von dynamischen Bibliotheken kann die Arbeitsweise von einzelnen Prozessen gezielt beeinflusst werden (vgl. [2], [14]).

Reaktion

Auch bei der nichtautorisierten Rekonfiguration verteilter IDS-Komponenten ist nur ein manuelles Eingreifen des Sicherheitsadministrators angebracht, weil eine Falschmeldung relativ unwahrscheinlich und weil die Größe eines möglicherweise entstandenen Schadens über den einer kompromittierten IDS-Komponente weit hinausgeht.

5 Implementation

Basierend auf fundamentalen Anforderungen an verteilte IDS [3], [15], sowie auf weiteren, konkreten Anforderungen [9] wurde eine Menge universell einsetzbarer, generischer Software-Komponenten entwickelt, auf deren Basis man verschiedene verteilte IDS-Architekturen realisieren kann. Sie wurden als reine User-Space-Komponenten entworfen, d.h. ohne Zugriff auf Betriebssystem-interne Funktionalitäten und auf dedizierte Hardware. Der vorgestellte Ansatz zielt daher darauf ab, einfach zu installierende Zusatzdienste für heute bereits im Einsatz befindliche Plattformen zur Verfügung zu stellen („Add-On"-Ansatz).

Um in der Lage zu sein, nahezu jede beliebige Art existierender Datenquellen (Programme, Prozesse, Dateien) - auch Third-Party-Produkte - als IDS-Sensor einsetzen zu können, wurden generische *Sensor-Adapter* entwickelt, die einfach rekonfiguriert und angepasst werden können. Detailliertere Informationen zur Implementation finden sich in [9]. Die Realisierung einiger der im vorangegangenen Abschnitt geschilderten Gegenmaßnahmen gegen DoS erfolgte auf Basis dieses Prototypen und wird im folgenden umrissen.

5.1 Stören der IDS-Kommunikationsverbindungen

Prävention

Der Einsatz von Verschlüsselung sowie Integritätsprüfung und Authentifizierung der Kommunikationspartner ist im TLS-Protokoll (Transport Layer Security [5]) spezifiziert. TLS wird als Bestandteil des IDXP-Profils (Intrusion Detection message eXchange Protocol, [6]) für alle Netzwerk-Kommunikationsvorgänge unter den IDS-Komponenten eingesetzt. Somit werden Veränderungen am Datenstrom zuverlässig erkannt und das Mitlesen durch potentielle Angreifer verhindert. Zusätzlich werden wechselnde TCP-Ports verwendet, um einer sofortige Re-Terminierung einer unterbundenen Netzwerkverbindung vorzubeugen. Aus Aufwandsgründen, die dem gewählten „Add-On"-Ansatz (s.o.) entgegenstehen, wurde auf eine Implementierung redundanter Kommunikationskanäle verzichtet. Zu versendende Ereignismeldungen werden auf dem lokalen Datenträger zwischengespeichert, um sie ggf. neu versenden zu können.

Detektion

Zur Erkennung unterbrochener Netzwerkverbindungen wird zum einen nach fehlgeschlagenen Lesevorgängen (Datum der Länge 0 bzw. EOF), zum anderen nach auftretenden Betriebssystem-Signalen (z.B. SIGPIPE) Ausschau gehalten. Die im verwendeten IDMEF-Datenformat [4] spezifizierten „Heartbeat-Messages" werden von den Agenten als Keep-Alive-Nachrichten in festgelegten Zeiträumen an die Konsole verschickt und von dieser ein eventuelles Ausbleiben nach Zeitüberschreitung überprüft.

Reaktion

Wird von einem Agenten festgestellt, dass die Kommunikationsverbindung zur Konsole unterbrochen wurde, so wird zunächst versucht, eine neue Verbindung zu etablieren. Gelingt dies nach einer spezifizierten Anzahl von Versuchen nicht, so wird ein Response-Modul aktiviert, das eine SMTP-Nachricht an die Administratorkennung absetzt. Dies macht selbstverständlich nur Sinn, wenn nicht der gesamte Netzwerkverkehr, sondern nur beispielsweise die TCP-Verbindung des IDS unterbunden worden ist. Daher sind auch andere Response-Möglichkeiten konfigurierbar, so etwa ein sauberes Herunterfahren des Systems oder die Rekonfiguration eines Paktfilters.

5.2 Prozessterminierung

Detektion

Zur Erkennung terminierter IDS-Prozesse werden Betriebssystem-Signale beobachtet, d.h. Agenten erkennen, wenn Sie selbst (z.B. SIGTERM) oder Sensoren als ihre Kindsprozesse (SIGCHLD) terminiert werden sollen. Erhält ein Agent ein SIGKILL-Signal, so kann es dieses naturgemäß nicht abfangen; die Prozessterminierung stellt sich auf der Konsole wie eine unterbrochene Netzwerkverbindung dar.

Von einer Realisierung Prozess-überwachenden Codes, der in Fremdprozesse eingebracht wird, wurde zunächst wegen der fehlenden Portabilität Abstand genommen.

Reaktion

Erkennt ein Agenten-Prozess, dass einer seiner Sensoren als Kindsprozess terminiert werden soll, so setzt er eine entsprechende Ereignismeldung an die Konsole ab. Der Sensor wird – wenn möglich – sofort neu gestartet, um den Betrieb aufrecht zu erhalten. Soll der Agent dagegen selbst terminiert werden, so verhindert er dies, soweit möglich, und setzt ebenfalls eine entsprechende Meldung an die Konsole ab.

5.3 Blenden durch nichtautorisierte Rekonfiguration

Detektion

Die Detektion von nichtautorisierten Manipulationen an den Prozessumgebungen der verteilten Komponenten wird durch Integritätsprüfung bewerkstelligt. Zu diesem Zweck werden zunächst die Bestandteile der Prozessumgebungen von Agenten und Sensoren spezifiziert und dann über Dateiinhalte, Benutzer- und Gruppenzugehörigkeit, Zugriffsrechte und Zugriffszeiten der Dateien *Checksummen* (Digest-Funktionen, z.B. MD5) gebildet. Die Berechnung wird durch den Agenten durchgeführt. Die initialen Werte werden an einer nicht veränderbaren Stelle (z.B. auf einem nicht wiederbeschreibbaren Dateisystem auf dem Konsolen-Server) abgelegt. Zu bestimmten Zeitpunkten, d.h.

- nach einem Neustart des IDS oder einer seiner verteilten Komponenten,
- periodisch sowie
- auf Anforderung des Sicherheitsadministrators

werden die Checksummen neu berechnet und von der Konsole mit den dort gespeicherten Werten verglichen.

Eine Implementierung einer Integritätsprüfung für Prozesse zur Laufzeit (z.B. mit `ptrace()`) ist sehr betriebssystemspezifisch und wurde daher nicht durchgeführt.

6 Zusammenfassung und Schlussfolgerungen

In diesem Beitrag wurden zunächst Methoden und Auswirkungen von Denial-of-Service-Angriffen gegen Komponenten verteilter Intrusion-Detection-Systeme erläutert. Davon ausgehend wurden mögliche Mechanismen zur Prävention und Erkennung sowie zur Reaktion auf DoS-Angriffe aufgeführt. Zusätzlich wurden auch Gründe für mögliche Falschmeldungen in Betracht gezogen und die Grenzen aufgezeigt, die sich durch einen User-Space-orientierten Ansatz ergeben. Die Implementation einiger dieser Schutzmechanismen wurde umrissen.

Im Laufe der Entwicklungsarbeit haben sich folgende Erkenntnisse ergeben:

- Es ist möglich, auch mit einem User-Space-basierenden Ansatz ein gewisses (für manche Einsatzbereiche durchaus ausreichendes) Maß an Sicherheit vor schadhaften Vorfällen herzustellen.
- Privilegierte Operationen (Veränderung bestimmter Dateien, Prozessmanipulationen, Änderung der Benutzerkennung, Laufzeitveränderungen im Prozess- und Kernspeicher) müssen durch zusätzliche, kernbasierte Autorisierungsmechanismen vor unbefugter Ausführung geschützt werden.
- Ein Umgehen vieler der hier vorgestellten Ansätze ist nur durch Manipulation des Betriebssystem-Kerns möglich. Wenn ausgeschlossen werden kann, dass derartige Änderungen nicht ohne Neustart (der mit hoher Wahrscheinlichkeit von einem Überwachungssystem erkannt wird) des Systems möglich sind, so hat die Verteidigung einen – wenn auch geringen – zeitlichen Vorsprung erlangt, der mitunter ausreicht, um entsprechende Maßnahmen einzuleiten.
- Bei Netzwerk-Sensoren sind dedizierte Hardware-Komponenten, die im sogenannten „Stealth Mode“ alle eine Leitung passierenden Datenpakete inspizieren (ohne selbst durch eine IP-Adresse in das Netzwerk integriert zu sein), den Software-Installationen auf herkömmlich vernetzten Rechnern vorzuziehen, weil sie die geringstmögliche Angriffsfläche bieten.

Das Thema Überlebensfähigkeit und Selbstschutz von verteilten IDS ist bisher noch nicht erschöpfend behandelt worden; allerdings versprechen Ansätze zur Durchsetzung von Sicherheitspolicies durch entsprechende Auslegung von Betriebssystem-internen Funktionalitäten mittel- und langfristig wesentlich mehr Erfolg.

7 Literaturverzeichnis

[1] Bhattacharya S. und N. Ye: *Design of a Robust, Survivable Intrusion Detection Agent*. Proceedings of the 1st Asia-Pacific Intelligent Agent Technology Conference (IAT'99), 274-278, 1999.

[2] Cesare, S.: *Syscall Redirection without Modifying the Syscall Table*. http://www.big.net.au/~silvio/stealth-syscall.txt, 1998.

[3] Crosbie, M. und E. Spafford: *Active Defense of a computer system using autonomous agents*. Technischer Bericht, The COAST Group, Department of Computer Science, Purdue University, West Lafayette, IN, Februar 1995.

[4] Curry, D. und H. Debar: *Intrusion Detection Message Exchange Format - Data Model and Extensible Markup Language (XML) Document Type Definition*. IETF Internet Draft draft-ietf-idwg-idmef-xml-03.txt, 2001. IETF IDWG.

[5] Dierks, T. und T. Allen: RFC 2246: *The TLS Protocol Version 1.0*. http://www.ietf.org/rfc/rfc2246.txt, Januar 1999.

[6] Feinstein, B., G. Matthews and J. White: *The Intrusion Detection Exchange Protocol*. IETF Internet Draft draft-ietf-idwg-beep-idxp-03.txt, September 2001. IETF IDWG.

[7] Handley, M., V. Paxson und C. Kreibich: *Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics*. Proc. USENIX Security Symposium 2001.

[8] Holz, T., M. Meier und H. Koenig: *High-efficient Intrusion Detection Infrastructure*, May 2002. Presented at the NATO/RTO Symposium on Real-Time Intrusion Detection (RTID) 2002, Estoril, Portugal.

[9] Jahnke, M.*: An Open and Secure Infrastructure for Distributed Intrusion Detection Sensors*. Proceedings of the NATO Regional Conference on Communication and Information Systems (RCMCIS'02), Zegrze, Polen, Oktober 2002.

[10] Jansen, W., P. Mell, T. Karygiannis und D. Marks*: Mobile Agents in Intrusion Detection and Response*. Proceedings of the 12th Annual Canadian Information Technology Security Symposium. National Institute for Standards and Technology, Gaithersboutg, MD, 2000.

[11] Lehti, R.: AIDE Download Page. http://www.cs.tut.fi/~rammer/aide.html, 2001.

[12] Porras, P. und P. Neumann: EMERALD*: Event Monitoring Enabling Responses to Anomalous Live Disturbances*. Proceedings 20th NIST-NCSC National Information Systems Security Conference, 353-365, 1997.

[13] Ptacek, H. und H. Newsham: Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. Technischer Bericht, Secure Networks, Inc., 1998.

[14] Secure Reality Pty Ltd.: *InjectSO Version 0.2 download page*. http://www.secureреality.com.au/archives.html, 2002.

[15] Spafford, E.und D. Zamboni: *Intrusion detection using autonomous agents*. Computer Networks, 34:547-570, 2000.

[16] Takada, T. und H. Koike: NIGELOG*: Protecting Logging Information by Hiding Multiple Backups in Directories*. Proceedings of the International Workshop on Electronic Commerce and Security, 874-878, 1999.

[17] Tripwire Download Page. http://www.tripwire.org/downloads/index.php, 2001.

[18] Vigna, G., S. Eckmann und R. Kemmerer: *The STAT Tool Suite*. Proceedings of DISCEX 2000, Hilton Head, South Carolina, 2000. IEEE Computer Society Press

[19] Wolthusen, S.: *Embedding Policy-Controlled ID Sensors within Host Operating System Security Enforcement Components for Real Time Monitoring*. NATO/RTO Symposium on Real-Time Intrusion Detection (RTID), Estoril, Portugal, 2002.

[20] Wolthusen, S.*: Security Policy Enforcement at the File System Level in the Windows NT Operating System Family*. Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC'01), New Orleans, LA, 2001.