Ein flexibles komponentenstrukturiertes Intrusion Detection System

Abdelhak Berraqa, Peter Herrmann, Nasser Ilayyan, Heiko Krumm,
Niels Schmitt, Jens Schröder, Marc-Christian Schröer, Sebastian Stöcker,
Anton Stoll, Jens Thiemann, Lars Wiebusch

Universität Dortmund, FB Informatik
LS IV, Rechnernetze und verteilte Systeme
D-44221 Dortmund
E-Mail {herrmann | krumm }@ cs.uni-dortmund.de
Fax: +49-231-7554730

Intrusion Detection Systeme IDS haben eine stark wachsende Bedeutung für die Sicherheit in Rechnernetzen. Zukünftige Systeme sollen flexibel und verteilt sein, um bei geringer Systembelastung umfassende Erkennungsfunktionen zu unterstützen und dynamisch an veränderliche Anforderungen anpassbar zu sein. Wir stellen ein modulares IDS vor. Seine Architektur folgt dem von der DARPA geförderten Ansatz des Common Intrusion Detection Framework CIDF, das Komponententypen und Mittel zur dynamischen Systembildung sowie zur Kommunikation und Kooperation zwischen Komponenten definiert. CIDF-Komponenten bilden Module im Großen. Sie lassen sich sehr gut mit dem neueren Softwaretechnik-Ansatz der Komponentenstrukturierung verbinden, der darauf abzielt, Anwendungen bedarfsgesteuert durch Zusammensetzen aus Komponenten zu bilden. Intrusion Detection Systeme sind spezielle Managementsysteme. Deshalb orientieren wir uns insbesondere an Ansätzen zur Unterstützung komponentenstrukturierter Managementsysteme und verwenden das Java Dynamic Management Kit J-DMK. Die Management-Oberfläche unseres IDS ist Web-basiert und unterstützt Bedienung, Pflege und Anpassung über gängige Web-Browser. Es wird zur Zeit in Windows NT Netzen erprobt.
Wir stellen die Architektur des Systems vor und berichten über Betriebserfahrungen und Erprobungen mit typischen Angriffsszenarien und Angriffstools.

Intrusion Detection Systeme

Intrusion Detection Systeme IDS ergänzen das traditionell durch Zugangs- und Zugriffskontrolldienste geprägte Sicherheitsmanagement vernetzter Systeme. Sie führen automatisierte Monitoring- und Auditing-Funktionen aus, die darauf spezialisiert sind, frühzeitig zu erkennen, ob vernetzte Rechner unbefugt genutzt oder mißbraucht werden [vHK98]. Ein IDS besitzt dazu eine Schnittstelle, an der Ereignisse des Netzes (z.B. Paketaustausch), Ereignisse der Betriebssysteme (z.B. Sitzung öffnen) und / oder Ereignisse von Applikationen (z.B. spezielle Ressourcenzugriffe) vorzugsweise in Echtzeit entgegengenommen werden. Das IDS analysiert die Ereignishistorie, um Besonderheiten zu erkennen, die auf Angriffe oder Missbrauch hinweisen und erzeugt Meldungen an den Administrator. Es kann um Reaktionsfunktionen zu einem Intrusion Detection and Response System ergänzt werden. Bisherige Erfahrungen mit automatisierten Response Systemen weisen allerdings auf Tendenzen zu stark betriebsstörenden Überreaktionen hin, so dass sich die meisten aktuellen Ansätze auf die Erkennung und Administrator-Unterrichtung konzentrieren. Zentral für ein IDS sind die Analysefunktionen, bei denen grob zwischen musterbasierter Angriffserkennung und Anomalieerkennung unterschieden werden kann. Häufig werden unterschiedliche Analyseverfahren kombiniert, um einerseits auch unbekannte Angriffstypen zu erkennen und andererseits allzu häufige Fehlalarme zu vermeiden. Verschiedene Quellen gehen davon aus, dass die praktische Bedeutung von IDS stark zunimmt und sie schon in den nächsten Jahren ein mit Virenscannern vergleichbares Marktvolumen besitzen werden.

Verteilte Intrusion Detection Systeme

Flexibilität stellt sich als sehr wichtige Anforderung an zukünftige IDS, da sie stark-dynamischen Einflüssen begegnen müssen. Bei erweiterten Aufgabenfeldern und veränderlichen Nutzungsprofilen zukünftiger vernetzter IT-Anwendungen sind vielfältige Schwachstellen zu befürchten, die von wachsenden und sehr aktiven Gruppen generell Intrusion-Interessierter immer schneller entdeckt und verwertet werden. Die aus diesen Gruppen stammenden Hinweise und Tools werden dabei immer häufiger auch von weiteren Personen zur Verfolgung direkter Interessen benutzt (z.B. von Insidern), so dass ein hohes Schadenspotential entsteht und sogar gut nach außen abgeschottete Netze durch interne Angriffe gefährdet sind. Aus dieser Dynamik resultieren Anforderungen nach Flexibilität, Erweiterbarkeit und Leistungsfähigkeit. Sie sollen durch Verteilung, Komponentenstrukturierung und Kooperation zwischen unterschiedlichen IDS erreicht werden. Im Bereich der Komponentenstrukturierung ist dazu das von der DARPA geförderte Common Intrusion Detection Framework CIDF interessant [cidf99]. Es unterscheidet zwischen Ereigniserzeuger-, Ereignisspeicher-, Analyse- und Reaktionskomponenten und definiert die erweiterbare Sprache CISL (Common Intrusion Specification Language) zur Kommunikation zwischen Komponenten. Weiterhin unterstützt es dynamische IDS-Konfigurationen durch einen Matchmaker genannten Informations- und Bindedienst. Man kann ferner starke Zusammenhänge zur aktuellen Arbeit der Internet Engineering Task Force Arbeitsgruppe IDEF finden, die sich schwerpunktmäßig der Definition eines Austauschformats für die Kooperation zwischen unterschiedlichen IDS widmet [idef99].

Komponentenstrukturiertes Management

Die Anforderungen nach Flexibilität, Erweiterbarkeit und belastungsmindernder Verteilung bestehen nicht nur bei IDS sondern treffen allgemein auf moderne Managementsysteme zu. Auch dort besteht das Problem, ein flexibles System mit großem Funktionsumfang zu unterhalten und dynamisch an veränderte Randbedingungen anzupassen. Schon etwas länger bekannte Lösungsansätze beginnen mit der Funktionsaufteilung auf Manager und Agenten. Sie sehen die Delegation von Aufgaben an Agenten vor und führen Abstraktionsebenen mit entsprechenden Funktionshierarchien ein [KaB97]. Relativ neu ist eine Kombination dieser Ansätze mit expliziter Komponentenstrukturierung [Der97], wie sie insbesondere durch das Java Dynamic Management Kit J-DMK unterstützt wird [dmk98]. Es wendet das Java Bean Komponentenmodell an und stellt einen Rahmen für (u.U. Web-basiert zugegriffene) dynamische Agenten- und Masteragentenstrukturen zur Verfügung. Darüberhinaus kann die Funktionalität einzelner Agenten im Push- oder auch Pull-Verfahren durch Integration und Desintegration von Komponenten verändert werden. Dem Komponentenansatz aus der Softwaretechnik folgend [Szy97] können die Komponenten eines Systems durchaus von unterschiedlichen Herstellern und Lieferanten stammen, so dass die Vorteile eines hoffentlich bald entstehenden weltweiten Management-Bean-Marktes genutzt werden können.

Intrusion Detection System JIMbean

Das von uns entwickelte Intrusion Detection System JIMbean (**J**ava **I**ntrusion Detection and **M**anagement System based on **Beans**) kombiniert die Ansätze verteilter IDS und komponentenstrukturierter Management-Systeme. Es orientiert sich auf logischer Ebene am CIDF. So sind entsprechende Generator-, Datenbasis-, Analyse- und Reaktionskomponenten (im CIDF sogenannte E-, D-, A- und R-Boxes) definiert und die Kommunikation zwischen Komponenten besteht wie im CIDF vorgesehen aus dem Austausch CISL-basierter General Intrusion Detection Objects GIDOs.

Die Implementierung basiert im wesentlichen auf der allgemeinen Java-Plattform. Sie wurde durch das J-DMK zur Unterstützung der verteilten Komponentenstrukturierung ergänzt. Eine dem J-DMK entsprechende Struktur aus Agenten, Masteragenten und Web-Zugängen dient der Aufnahme und Verwaltung der logischen IDS-Komponenten, die eine Ausprägung als Java Beans haben und dynamisch im System konfiguriert werden können. Weiterhin ist zu erwähnen, dass die Java Cryptographic Extension JCE zur Verschlüsselung der Kommunikation zwischen den Agenten eingesetzt wird. Die Web-Browser-Schnittstelle zur Administration des Systems wird durch Servlets unterstützt, die unter Verwendung des Java Servlet Development Kit JSDK erstellt wurden. Eine weitere Besonderheit ist, dass ereigniserzeugende Komponenten das Java Native Interface JNI verwenden, um Betriebssystem- und Netzwerkereignisse zu erkennen.

Makroskopische Architektur

Ein vollständiges JIMbean-System besteht aus der Management-Applikation, die als Servlet realisiert auf einem Web-Server installiert wird, dem Lookup-Service, der als eine Art Directory-Service fungierend die einzelnen Agenten miteinander in Beziehung bringt, und einer Menge von Agenten, die auf den zu überwachenden Netzknoten arbeiten.

Agenten teilen dem Lookup-Service mit, welche Ereignisse von ihnen verschickt werden können. Außerdem fragen sie beim Lookup-Service an, welche Agenten die von ihnen gewünschten Ereignisse erzeugen. Diese Ereignisse werden dann direkt bei dem erzeugenden Agenten bestellt. Der Lookup-Service enthält zu jeder Zeit Informationen über die aktiven Agenten und die von ihnen erzeugten Ereignisse. Er informiert Agenten über neue Quellen für von ihnen abonnierte Ereignisse, über das Versiegen dieser Quellen und über neue und verlorene Agenten.

Mikroskopische Architektur

Bild 1 gibt eine Übersicht über die detailliertere Architektur des JIMbean-Systems. Sie ist durch Agenten im Sinne des J-DMK geprägt. Jeder Agent kann Komponenten aufnehmen, die als Managed Beans (M-Beans im Sinne des J-DMK) realisiert wurden. Dabei können zur Laufzeit neue Komponenten dem Agenten hinzugefügt oder aus ihm entfernt werden. Durch eine Konfigurationsdatei, die auf dem Web-Server hinterlegt ist, wird einem Agenten mitgeteilt, welche Komponenten bei Systemstart zu laden sind.

Eine der wichtigsten Komponenten ist der EventBroker, der die Kommunikation des Agenten mit dem Lookup-Service und den anderen Agenten übernimmt. Der EventBroker teilt dem Lookup-Service mit, welche Ereignisse von den Komponenten erzeugt werden. Darüber hinaus informiert er sich über Agenten, die von den Komponenten benötigte Ereignisse liefern und bestellt diese direkt bei diesen Agenten. Er wird bei Konfigurationsänderungen über neue Ereignissquellen informiert und nimmt dann Kontakt mit diesen auf, um diese Ereignisse zu abonnieren.

Der EventBroker liefert auf Anfrage außerdem Informationen über die installierten Komponenten. Dazu gehören textuelle Beschreibungen sowie nähere Informationen über die Parametrisierungsmöglichkeiten, die dem Administrator gegeben sind.

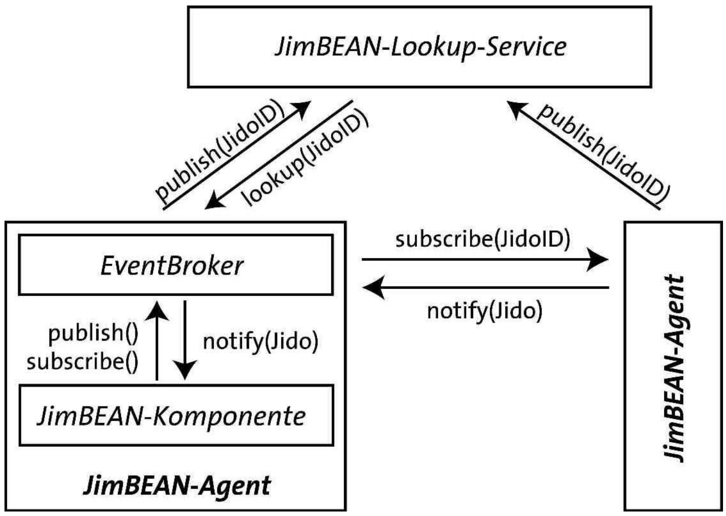

Bild 1: Architekturübersicht

Ereignisse

Aus Effizienzgründen werden innerhalb des IDS anstatt der vom CIDF vorgeschlagenen GIDO-Nachrichten Java-Objekte ausgetauscht, die von uns Java Intrusion Detection Objects JIDOs genannt werden und direkte Ereignisbeschreibungen repräsentieren. Die im CIDF vorgesehene explizite Darstellung von GIDOs in Nachrichten findet deshalb nur noch zur Kommunikation mit Fremdsystemen Anwendung. Von Ereignisgenerator-Komponenten erzeugte JIDOs bilden die Schnittstelle zur konkreten Systemumgebung. Sie abstrahieren vom jeweiligen Betriebssystem und kapseln Informationen über sicherheitsrelevante Aktionen im Rechnernetz, wie z.B. Zugriffe auf Dateien, Anfragen auf Ports und Anmeldeversuche in Netzknoten. Neben Ereignisgeneratoren können aber auch die anderen Komponenten JIDOs erzeugen. Insbesondere Ereignisspeicher- und Analyse-Komponenten können Informationen aus empfangenen JIDOs in gefilterter und aggregierter Form in selbst erzeugten JIDOs hinterlegen und somit weiteren Komponenten verfügbar machen.

Analyse-Komponenten

Die wichtigsten Funktionen eines IDS werden von den Ereigniserzeugungs- und Analyse-Komponenten übernommen, da sie die Folge auftretender Ereignisse überwachen sollen, um Angriffe und Anomalien zuverlässig zu erkennen und zu signalisieren. Dabei geht gute Erkennung in der Regel mit deutlicher Systembelastung einher, die durch Ereigniserzeugung, Speicherung, Kommunikation und insbesondere auch komplexe Analysealgorithmen bedingt ist. Es ist deshalb nicht nur interessant, Analysekomponenten zur Verbesserung der Erkennungsqualität zu kombinieren, sondern günstige Konfigurationen unterschiedlicher Komponenten können auch zur deutlichen Reduktion der Systembelastung führen. So kann lokale Vorverarbeitung und Filterung den Kommunikationsaufwand vermindern und aufwendige Analysekomponenten können in eigens dafür reservierten Netzknoten platziert werden. Das JIMbean-System ist deshalb in der Lage, unterschiedliche Analyse-Komponenten aufzunehmen und kann sie auch in relativ freizügig gewählter Konfiguration betreiben.

Basisklasse Component

Alle Komponenten in JIMBean werden von der abstrakten Basisklasse Component abgeleitet. Sie stellt die Grundfunktionen einer JIMBean-Komponente zur Verfügung: Eingehende JIDOs werden lokal gepuffert und jeweils dem speziellen Analyseprozess übergeben. Die aus dem Analyseprozess resultierenden JIDOs werden automatisch versandt. Ferner wird eine SQL-Schnittstelle zu einer Datenbank unterstützt, um persistente Informationsspeicherung und leistungsfähige Retrieval-Funktionen zur Verfügung zu stellen. Im folgenden sollen drei Komponenten beispielhaft umrissen werden.

NetworkMonitor

Der NetworkMonitor ist eine Ereigniserzeugungskomponente für Netzwerk-Ereignisse. Es werden die den lokalen Netzknoten betreffenden IP-Pakete erfasst und auf JIDOs abgebildet. Der Monitor erzeugt SendMessageJIDOs, welche die Rahmen-Daten eines Paketes enthalten, und StreamJIDOs, welche die zusammengesetzten Inhalte der IP-Pakete IP-basierter Datenströme beschreiben.

TCPSignatureAnalysisComponent

Diese Komponente analysiert IP-Streams anhand von StreamJIDOs. Das Analyseverfahren ist eine musterbasierte Angriffserkennung. Die Angriffssignaturen sind in Form von regulären Ausdrücken (gemäß Perl5 [WaCS96]) in der Datenbank persistent abgelegt. Ein IP-Stream wird jeweils mit diesen Signaturen verglichen. Beispielsweise kann ein Angreifer durch einen HTTP-Request der Form „`GET ../..`“ einen denial-of-service Angriff auf den Internet Information Server durchführen (da der Request zu einem Crash des Windows-NT IIS führen kann). Der folgende reguläre Ausdruck erfasst diesen Angriff: ‚`GET\s\.\.[/\]\.\.`‘. Neben der signaturbasierten Analyse unterstützt die Komponente weiterhin eine Administrationsschnittstelle, über die auch die Einträge in der Signaturdatenbank gepflegt werden können.

TCPPortScanDetector

Diese Komponente analysiert den TCP/IP Verbindungsaufbau anhand von SendMessageJI-DOs. Zum Aufbau einer TCP/IP Verbindung müssen beide Teilnehmer eine initiale Sequenznummer vereinbaren. Dies geschieht anhand eines sogenannten 3-Way-Handshakes. Dazu sendet der Initiator des Verbindungsaufbaus eine Protokolldateneinheit mit SYN-Bit, das, falls auf dem angesprochenen Port ein Dienst aktiv ist, mit einem SYN/ACK beantwortet wird. Um den Verbindungsaufbau abzuschließen, muß der Initiator ein ACK senden. Ein nicht belegter Port antwortet mit einem RST anstatt mit einem SYN/ACK [rfc793].

Das entsprechende Analyseverfahren basiert auf Anomalieerkennung. Ziel ist es, Half-Open-Scans, das heißt nicht korrekt abgeschlossene 3-Way-Handshakes, und Port-Scans zu erkennen, die sich durch eine ungewöhnlich hohe Anzahl von TCP/IP Verbindungsaufbauten definieren. Auf jedem zu überwachenden Rechner ist deshalb ein NetworkMonitor installiert, der SendMessageJIDOs generiert und diese an einen im überwachten Sub-Netz befindlichen TCPPortScanDetector versendet.

Ein Half-Open-Scan liegt vor, wenn das letzte ACK-Paket eines 3-Way-Handshakes nicht übermittelt wurde. Um dies zu erkennen, muss die Zeit gelernt werden, die normalerweise benötigt wird, um einen 3-Way-Handshake korrekt abzuschließen. Sie ist von der lokalen Umgebung abhängig.

Um einen Port-Scan zu erkennen, muss die Anzahl korrekter TCP/IP Verbindungsaufbauten, die in einem gewissen Zeitfenster im normalen Betrieb zustande kommt, gelernt werden. Die einzelnen Daten werden in einer Datenbank protokolliert. Zur Berechnung der Normalwerte wird ein statistisches Verfahren benutzt, das auf iterativ berechnenden Schätzern für den Erwartungswert und die Varianz der Daten beruht. Die Parameter dieses Lernprozesses können über die Administrationsschnittstelle der Komponente von außen konfiguriert werden.

Angriffe

Neben Verfügbarkeitsangriffen, deren Auswirkungen in unserem nicht-kommerziellen Umfeld nicht so kritisch sind, spielen insbesondere Angriffe eine Rolle, die das Ziel haben, sich Nutzerrechte zu verschaffen, um existierende Accounts mitzuverwenden oder sich sogar über Administratorrechte neue eigene Accounts einzurichten. Die geschaffenen Möglichkeiten werden dabei bisher erfahrungsgemäß weniger dazu verwendet, um durch Datenmanipulation direkten Schaden zu erzeugen. Beliebter ist es, Netzknoten unerkannt als Relais-Stationen zu verwenden, um z.B. MP3-Dateien zu hinterlegen oder von dort aus weitergehende Netzerkundung zu betreiben. Auch scheint es sehr beliebt zu sein, unberechtigte Nutzungsmöglichkeiten für eventuelle spätere Zwecke zu schaffen und zu akkumulieren. Es ist zu vermuten, dass viele der gefundenen Möglichkeiten in einschlägigen Kreisen ausgetauscht werden.

Bei zahlreichen und relativ freizügig zugänglichen Arbeitsplatzrechnern sind auch lokale Angriffe interessant, die oft spätere abgesetzte Zugriffe vorbereiten sollen. Lokale Angriffe auf Windows-NT Clients können dazu z.B. auf die Passwort-Hashes vorheriger Nutzer abzielen. Die Entschlüsselung kann mit existierenden Tools durchgeführt werden. Zum Lesen der entsprechenden SAM-Dateien können Bootdisketten, u.U. unterstützt durch sogenannte Bios-Cracker-Tools zum Ermitteln eines Boot-Passworts, verwendet werden.

Entfernte Angriffe – wenn sie nicht von längerer Hand mittels Trojanischer Pferde vorbereitet wurden – werden typischerweise durch Scans eingeleitet, um zunächst Informationen über Netzknoten und Nutzer zu sammeln (Betriebssystem, Services und Versionen, Verzeichnisse und deren Einträge). Bei schlecht gepflegten Systemen mit älteren Versionsständen der Systemsoftware erfolgen daraufhin gezielte Angriffe, welche die dort bekannten Schwachstellen ausnutzen. Möglich sind weiterhin Zugriffsversuche, für die Nutzernamen aus gefundenen Verzeichnissen und geratene Passwörter verwendet werden. Beim Passwort-Raten helfen Listen häufig benutzter Passwörter. Außerdem wird versucht, Sniffer-Tools zu installieren, die Datenpakete auf Account-Informationen hin untersuchen.

Erprobung und Betrieb

Das flexible, komponentenstrukturierte und Web-basiert administrierbare IDS wurde in Java unter Anwendung des J-DMK-Frameworks implementiert. Umgebungsabhängige Komponenten (d.h. Generator-Komponenten) existieren bisher für ein MS Windows NT Netz, sie detektieren sowohl Netz- als auch Systemereignisse. Datenbasis- und Analysekomponenten sind weitgehend systemunabhängig. Es stehen sowohl statistikbasierte Anomalieerkennungskomponenten als auch musterbasierte Angriffserkennungskomponenten mit geeigneten Musterdefinitionen zur Verfügung. Zur Erprobung des Systems wurden im wesentlichen die aktuell für Windows NT relevanten Schwachstellen- und Angriffsdokumentationen des Computer Emergency Response Teams CERT für den Entwurf von Tests herangezogen. Darüberhinaus wurde eine umfangreiche Recherche im WWW durchgeführt. Sie zeigte, dass bereits heute eine sehr aktive Szene Intrusion-Interessierter existiert, die sich in weiten Teilen sehr freizügig austauschen und neben Adressen, Passwort-Listen und Schwachstellen-Informationen insbesondere auch Tools zur Verfügung stellen, mit denen auch Intrusion-Laien ohne größeren Aufwand bedenkliche Angriffe ausführen können. Wir setzten mehrere dieser Tools zum Testen ein. Außerdem durften wir in der Testperiode auch Zeugen echter Angriffe werden. Aus diesen Gründen wachsender Gefährdung erscheinen sensibel eingestellte IDS in Zukunft unumgänglich zu sein, obwohl ihr Betrieb naturgemäß zu nennenswerten Systembelastungen führt.

Leistungsbetrachtungen

Die Verwendung der Java-Plattform – ergänzt um J-DMK – ließ vermuten, dass das kontinuierliche Überwachen der Netz- und Systemaktivitäten in Verbindung mit den teilweise doch recht aufwendigen Analysefunktionen zu deutlichen Systembelastungen führen würde. Dies kam auch bei ersten Tests in einem sich über 5 PCs erstreckenden 10 Mbit/sec Ethernet-Segment zum Ausdruck. Es umfasste einen Linux-Server mit einer mySQL-Datenbank, die über JDBC angebunden wurde und zur Speicherung von JIDOs diente, einen Windows NT 4.0 Server als primären Domänencontroller und drei Windows NT 4.0 Workstations (Pentium II 300 MHz, 128 MByte Hauptspeicher). Auf der ersten Workstation war ein JIMbean-Agent mit verschiedenen IDS-Komponenten installiert, auf der zweiten wurden mittels verschiedener Tools Angriffe simuliert und die dritte diente als Managementoberfläche für den Agenten. Die Messergebnisse wurden mit Hilfe des Windows NT Systemmonitors ermittelt, der während des Tests auf der ersten Workstation die CPU-, Speicher-, und Netzwerkaktivität protokollierte. Dabei wurden die Rechner bewusst im Universitätsnetzwerk belassen, um die Belastung durch den normalen Netzwerkverkehr nicht aus der Analyse auszuschließen. Wie Bild

2 zeigt, ergab sich bei aktivierten Analysekomponenten eine so hohe CPU-Belastung, dass die Arbeitsstation kaum noch als solche verwendet werden konnte.

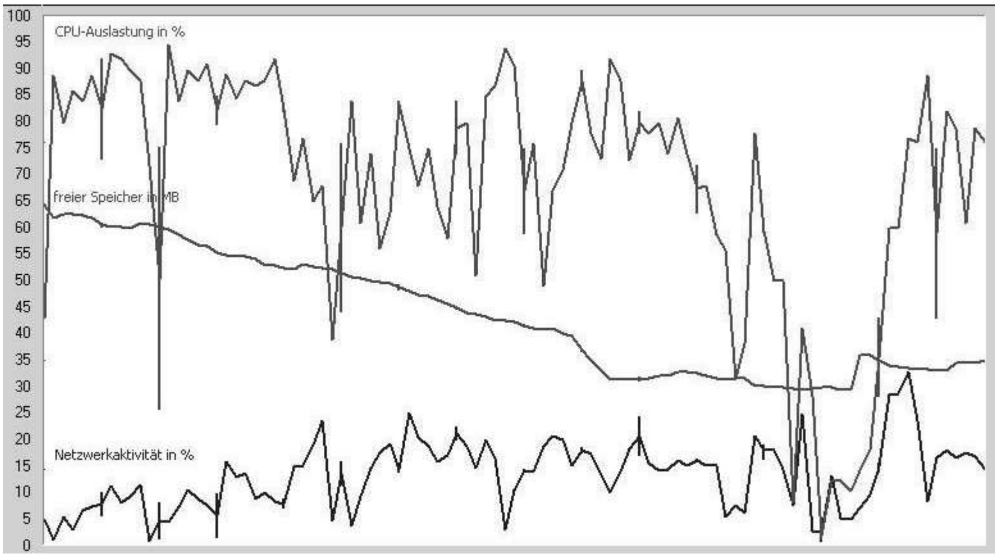

Bild 2: Belastung einer kombinierten Arbeits- und Analysestation

In einer anschließenden zweiten Testserie sollten nun die Vorteile der Verteilung untersucht werden. Wir reservierten eine Workstation ausschließlich für den Betrieb eines JIMbean-Agenten, der alle Analysekomponenten aufnahm. Im Ergebnis zeigte sich, dass diejenige Workstation, die nun nur noch durch Monitorkomponenten belastet war, tatsächlich auch nur noch in vertretbarem Umfang belastet wurde. Bild 3 zeigt dazu einen typischen Messwerteverlauf. Lediglich der Speicherbedarf führt hier zu nennenswerten Systembelastungen. Er ist annähernd konstant und im wesentlichen auf die in der virtuellen Java-Maschine vorzuhaltenden Java- und J-DMK-Klassen zurückzuführen.

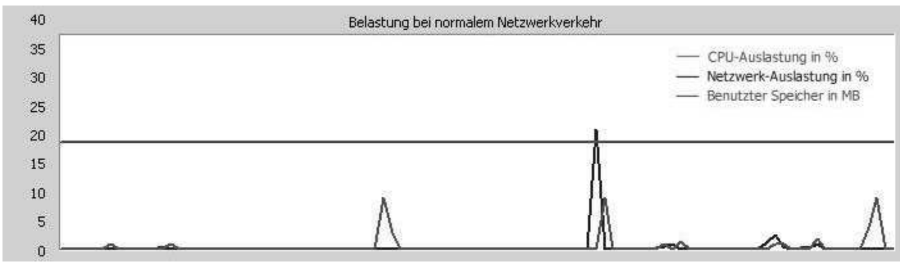

Bild 3: Belastung einer überwachten Arbeitsstation

Die Belastung der nun als „ID-Analyse-Server" betriebenen Workstation ist dagegen vergleichsweise hoch. Bild 4 veranschaulicht die entsprechenden Messwerte. Spitzen bei der CPU-Auslastung übersteigen hier die 90%-Marke, während die Speicherauslastung den typischen, durch den konstanten Java-Grundbedarf geprägten Verlauf besitzt. Die Spitzen bei der CPU-Auslastung werden von den rechenintensiven Analysekomponenten erzeugt. So verbraucht die signaturbasierte Analyse kurzfristig bis zu 70% der CPU-Zeit. Die Erkennung von Portscans als Anomalien kann in sehr kurzfristigen Spitzen mehr als 90% der CPU-Kapazität in Beschlag nehmen. Die Netzwerkauslastung bleibt moderat. Sie nahm auch im Vergleich zur ersten Testserie mit nicht-verteilter Agentenfunktion kaum zu, obwohl nun die von den Monitor-Komponenten erzeugten JIDOs über das Netz zur entfernten Analysestation transferiert wurden.

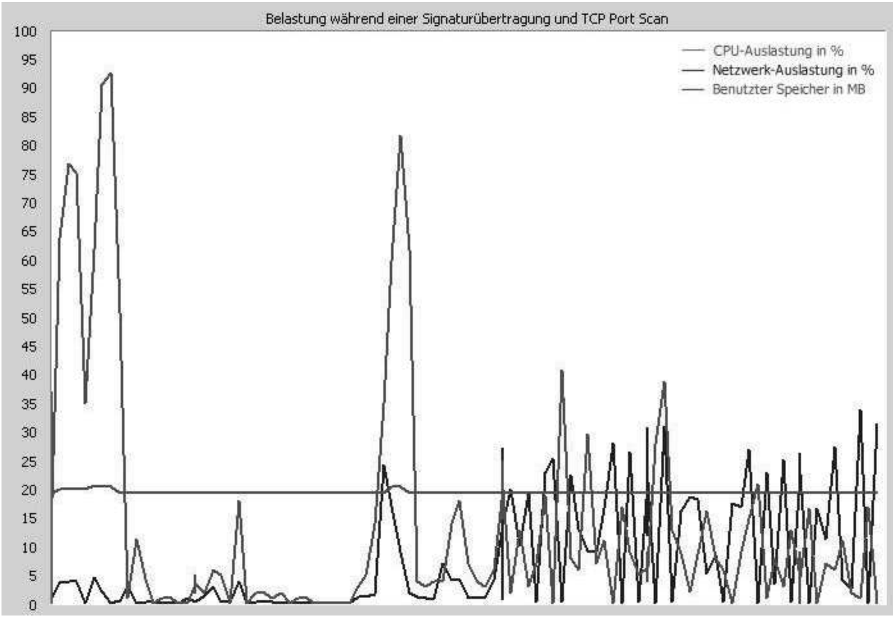

Bild 4: Belastung der Analyse-Station

Diese Experimentserie zur Leistungsbetrachtung wurde zwar einerseits unter vereinfachten Bedingungen durchgeführt, weil nur ein kleines 10Mbit/sec Ethernet-Segment verwendet wurde. Andererseits waren jedoch auch verschärfte Bedingungen gegeben, weil die Angriffe direkt aus diesem Segment heraus geführt wurden. Da sie auf einem ausschließlich dafür reservierten PC mit automatisierten Angriff-Tools erzeugt wurden, ergab sich eine im Vergleich zum Praxisbetrieb stark erhöhte Angriffsfrequenz.

Im Fazit wurde erkennbar, dass verteilte komponentenstrukturierte IDS gute Möglichkeiten eröffnen, um die Systembelastung durch geeignete Funktionsverteilung und durch Einsatz zusätzlicher Hardware auf ein erträgliches Maß zu reduzieren. Die von uns verwendete Plattform (Java und J-DMK) bringt zur Zeit allerdings noch einen deutlich spürbaren Overhead mit sich, so dass man damit rechnen muss, dass pro Netzsegment eine zusätzliche Analysestation und pro Station 20MB Arbeitsspeicher nur für die Unterstützung des Intrusion Detection Systems benötigt werden. Optimierungen des Systems werden interessant, die C++-basierte Monitorkomponenten für Workstations und auf Server-Seite effizient programmierte Analysekomponenten verwenden.

Ausblick

Das entwickelte IDS ist offen für den von der Internet Engineering Task Force Arbeitsgruppe IDEF verfolgten Ansatz der automatisierten Kooperation zwischen separaten und in unterschiedlichen Domänen arbeitenden IDS. Diese Kooperation soll auf längere Sicht den informellen Austausch von Erfahrungen, Warnungen, Erkennungsmustern und Abwehrhinweisen ergänzen, wie er momentan häufig interpersonell zwischen Administratoren stattfindet. Der Ansatz konnte allerdings bisher mangels Kooperationspartnern praktisch noch nicht verfolgt werden. Wir möchten im nächsten Schritt optimierte Komponenten einsetzen und weitere Erfahrungen mit der administratorgesteuerten flexiblen Anpassung des IDS gewinnen, bevor wir detailliertere Ansätze zur Integration von Selbstadaptionsfunktionen verfolgen.

Literatur

[cidf99] Common Intrusion Detection Framework CIDF, siehe im WWW unter http://seclab.cs.ucdavis.edu/cidf

[Der97] Luca Deri. A Component-based Architecture for Open, Independently Extensible Distributed Systems. Dissertation der philosophisch-naturwissenschaftlichen Fakultät Bern, Bern 1997

[dmk98] Java Dynamic Management Kit 3.0. siehe im WWW unter http://www.sun.com/software/java-dynamic/tech-overview.html

[idef99] Intrusion Detection Exchange Format, Internet Engineering Task Force Working Group IDEF, siehe im WWW unter http://www.ietf.org/ietf/lid-abstracts.txt

[KaB97] M. Kahani, H. Beadle. Decentralised Approaches for Network Management. ACM Computer Communications Review, July 1997, pg. 36-47

[rfc793] RFC 793, Transmission control protocol, Information Sciences Institute, University of Southern California, 1981

[Szy97] Clemens Szyperski. Component Software – Beyond Object Oriented Programming. Addison-Wesley, 1997

[vHK98] Josef von Helden, Stefan Karsch. Grundlagen, Forderungen und Marktübersicht für Intrusion Detection Systeme (IDS) und Intrusion Response Systeme (IRS). Debis IT Security Services, Bonn, 1998; erhältlich über Bundesamt für Sicherheit in der Informationstechnik BSI

[WaCS96] L. Wall, T. Christiansen, R. L. Schwartz, Programming perl, 2nd Edition, O'Reilly & Associates, 1996