Technische Universität Wien

DIPLOMARBEIT

Host- und Softwareinventarisierung in heterogenen Netzwerken am Fallbeispiel Securitymanagement mit *atc-inventory*

ausgeführt am Institut für Softwaretechnik und Interaktive Systeme
der Technischen Universität Wien

unter der Anleitung von Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Mag.rer.soc.oec. Stefan Biffl

durch

Norbert Andreatta
Schönbrunnerstr. 264/19
1120 Wien

Wien, 27. November 2007

_____

Unterschrift

Kurzzusammenfassung

In heterogenen Computernetzwerken mit verschiedenen Betriebssystemen, Serverdiensten, Applikationen und Netzwerkgeräten bekommt ein effizientes Sicherheitsmanagement eine immer größere Bedeutung. Sei es das Patchmanagement, das Konfigurationsmanagement, der Einsatz von verteilten Intrusion Detection Systemen oder die Risikobewertung realer Angriffsszenarien, als Grundlage wird immer das Wissen bzw. die Dokumentation der im Netzwerk aktiven Geräte und der installierten Software benötigt. Trotz der vielen wissenschaftlichen Publikationen über IT Asset Management gibt es keinen allgemein akzeptierten Standard zum Sammeln der für ein Sicherheitsmanagement relevanten hostbasierenden Daten, noch wie diese Daten weiterverarbeitet werden können. Meist bringen die verschiedenen Sicherheitsapplikationen eigene Hilfsmittel mit, die die bestehende Infrastruktur in einem der Applikation zugrundeliegenden Datenmodell abbildet. Dadurch werden nicht alle relevanten Informationen zur Netzwerk- und Hostinfrastruktur und zur installierten Software abgebildet, sondern nur die für diese spezielle Applikation benötigten Teilbereiche. Zudem ist es meist sehr aufwendig bzw. nicht möglich, dieses bestehende Datenmodell mit zusätzlichen Informationen zu erweitern.

Diese Diplomarbeit stellt einen alternativen und offenen Ansatz zum Inventarisieren der Netzwerkinfrastruktur, der aktiven Netzwerkkomponenten und der installierten Software vor. Die Idee hinter dem in dieser Diplomarbeit zu entwickelnden Inventarisierungsprogramm *atc-inventory* ist es, durch die Integration bestehender Sicherheits- und Netzwerkmanagementprogramme ein komplettes Bild der IT Infrastruktur zu erstellen und in einem einzigen Datenmodell abzubilden. Jede integrierte Applikation ist für einen speziellen Bereich der Inventarisierung zuständig, die Schnittstellen zu den eingebundenen Programmen ist genau spezifiziert. Durch den offenen Ansatz und die Integration verschiedener Sensoren ist es zudem möglich, alle relevanten Daten über die aktiven Hosts, die verwendeten Betriebssysteme und die installierte Software zu erfassen, und bei Bedarf mit zusätzlichen Informationen zu erweitern.

Da sich die Daten einzelner Sensoren überschneiden können, muß *atc-inventory* zudem die Daten der einzelnen Applikationen konsolidieren und zusammenführen. Die gesammelten Daten werden mittels XML in einer standardisierten Art und Weise abgespeichert. Dadurch wird sichergestellt, daß sie von verschiedenen Applikationen und in verschiedenen Kontexten weiterverarbeitet werden können. Die Applikation *atc-inventory* soll sowohl den kompletten Zyklus der Invenarisierung durchführen, als auch nur benötigte Teilbereiche inventarisieren.

Für Karin

Danksagung

Ich danke dem Kompetenzzentrum Secure Business Austria, insbesondere den Kollegen Stefan Fenz und Andreas Ekelhart, für die Unterstützung und für die interessanten Diskussionen zum Thema der Host- und Softwareinventarisierung.

Inhaltsverzeichnis

1 Einleitung

Sun Tzu - Art of War
Sun Tzu said:

Know the other and know yourself
Fight one hundred battles without danger. [TG88]

Durch die stetige Erweiterung des Internets und die Nutzung der sich dadurch ergebenden Möglichkeiten wird es immer wichtiger, die eigene Infrastruktur vor den ansteigenden Angriffsmöglichkeiten zu schützen. Die Kenntnis der eigenen Netzwerkinfrastruktur, der installierten Computer und der eingesetzten Software ist für diesen Vorgang essentiell. Laut den Top Ten Security Trends [13] des SANS (SysAdmin, Audit, Network, Security) Institut wird sich die Anzahl der Cyber Angriffe auf Firmen mit Kunden-, Entwicklungs- und Produktdaten immens vergrößern.
Zur Erkennung von Angriffen auf die Netzwerkinfrastruktur und auf die installierten Systeme werden meist verteilte Intrusion Detection Systeme eingesetzt. Diese untersuchen den Netzwerkverkehr und protokollieren jene Verbindungen, die Angriffsmuster und Schadkodesignaturen aufweisen, mit. Die wirklich relevanten und erfolgreichen Angriffe auf die zu schützende IT Infrastruktur zu finden, ist in der Fülle dieser Informationen jedoch schwierig. Es gibt verschiedene Lösungsansätze diese nicht relevanten Alarme zu unterdrücken. Der derzeit meist verwendete Ansatz ist die Korrelation der gemeldeten Alarme mit netzwerk- und systemspezifischen Daten. Es wird versucht, anhand der zusätzlichen Analyse von weiteren Informationen wie Log- und Asset Managementdaten zu bestimmen, ob ein Angriff erfolgreich war oder nicht. Treten keine weiteren Vorkommnisse im Netzwerk bzw. Anomalien von Diensten in Zusammenhang mit dem Angriff auf, war der Angriff nicht erfolgreich und der Alarm kann unterdrückt werden. Eine weitere sehr effiziente Möglichkeit zum Unterdrücken von nicht relevanten Alarmen, wird im Artikel *Using Alert Verification to Identify Successful Intrusion Attempts* [KRV04] beschrieben. Bei diesem Ansatz wird bei jedem vom Intrusion Detection System generierten Alarm anhand des Sicherheitsscanners Nessus [DRM+04] überprüft, ob das angegriffene System für diese Art von Angriff verwundbar ist. Ist das attackierte System für das im Netzwerkverkehr gefundene Angriffsmuster nicht anfällig, kann der Alarm unterdrückt

werden. Alle derzeit bestehenden Ansätze haben eine Gemeinsamkeit: Zur Überprüfung des Alarms werden genaue Informationen über die zu schützende Infrastruktur benötigt. Diese Daten erhält man durch die Inventarisierung der sich im Netzwerk befindlichen Computer- und Netzwerksysteme.
Ein weiterer Bereich, der eine Inventarisierung der Infrastruktur voraussetzt, ist das für ein gesamtheitliches Sicherheitsmanagement benötigte Patchmanagement. Durch ein effizientes Patchmanagement können sehr viele Angriffe durch Würmer, Viren und Trojaner abgewehrt werden. Dies kann sehr gut am Beispiel des Slammer Wurms (auch Saphire genannt) gezeigt werden. Dieser Wurm nützt eine Buffer Overflow Lücke im Microsoft SQL Server aus. Obwohl Microsoft für diese Lücke am 24. Juli 2002 den Sicherheitshinweis MS02-039 [5] publizierte und einen Patch bereitstellte, konnte der Wurm während des Ausbruchs am 25. Januar 2003 mindestens 75.000 Systeme mit einem installierten Microsoft SQL Server oder einer installierten Microsoft SQL Server Desktop Engine (MSDE) infizieren. Als Folge waren die infizierten Systeme und die betroffenen Netzwerke aufgrund der Überlastung nicht mehr verwendbar [MPS+03]. Dieser und weitere ähnliche Wurmausbrüche veranlaßten sehr viele Systemadministratoren, ihre Patchmanagementstrategien zu überdenken.
Die Wichtigkeit eines guten und verläßlichen Patchmanagements wird auch von Gartner, einer der führenden Information Technology Consulting Firmen, unterstrichen. In einer Studie [CWN02] kommt Gartner zum Schluß, daß das Patchmanagement ein kritischer Faktor für Verfügbarkeit und Integrität von geschäftsrelevanten Systemen ist. Obwohl das Patchmanagement durch den Einsatz von verschiedenen Systemen sehr komplex sein kann, muß bei Entdeckung von neuen Viren oder von Lücken in der eingesetzten Software eine schnell und automatisch durchführbare Installation von Patches möglich sein. Hierfür muß man zuerst wissen, welche Systeme im Netzwerk aktiv sind, und wo welche Software installiert ist.
Ein weiterer Bereich, der eine immer größere Relevanz bekommt und eine Inventarisierung des Netzwerks voraussetzt, ist das Risikomanagement. Eine Risikoeinschätzung diverser Angriffsszenarien, entdeckter Softwarelücken, Virenausbrüchen und auftretender Industriespionage kann nur durchgeführt werden, wenn der aktuelle Stand der installierten Systeme bekannt ist. Zudem müssen unternehmenskritische Systeme identifiziert werden. Für deren problemlosen Betrieb müssen sehr oft spezielle Voraussetzungen erfüllt sein. Durch die Dokumentation der Systeme wird die Durchführung von Risikoanalysen ermöglicht. Einzelne Szenarien können durchgespielt und das weitere Vorgehen geplant werden. Dies stellt für Klein- und Mittelbetriebe ein großes Problem dar, da sie großteils nicht die finanziellen Mittel für eine eigene EDV Abteilung mit Sicherheitsspezialisten haben. Deshalb müssen Möglichkeiten für ein kostensparendes Risikomanagement, das

eine Durchführung von Bedrohungsanalysen zuläßt, geschaffen werden [EFKW07].
Unternehmensnetzwerke sind einer ständigen Entwicklung unterworfen, es werden neue Systeme und Komponenten integriert, bestehende Softwarelösungen werden abgelöst und Abläufe neu definiert. In solchen Netzwerken gibt es im Regelfall verschiedene Insellösungen, die einzelne Teilbereiche abbilden. Ein solches Beispiel ist das Universitätsnetzwerk der Akademie der bildenden Künste Wien. Dort werden derzeit vier verschiedene Update- und Patchmanagementsysteme verwendet:

- Windows Server Update Services für Microsoft Windows Clients und Server
- Red Hat Network für Linux Server
- Sun Update Manager für Sun Solaris Server und
- Apple Remote Desktop für Mac OS X Clients

Als Intrustion Detection System kommt Snort [Bea04] zum Einsatz. Durch aufwendiges Konfigurieren wird ein Großeil der nicht relevanten Alarme unterdrückt. Risikoabschätzungen, ob und wo welche Patches und Updates eingespielt werden sollen, können aufgrund der heterogenen und mit der Zeit gewachsenen Netzwerkstruktur nur mit Hilfe verschiedener Managementsysteme und zusätzlicher Informationen über Softwareschwachstellen wie jenen von SecurityFocus, einer herstellerunabhängigen Webseite für sicherheitsrelevante IT bezogene Themen, durchgeführt werden. Und obwohl ein Großteil der für ein gesamtheitliches Sicherheitsmanagement benötigten Daten vorhanden sind, erhält man erst aufgrund von Erfahrung in der Betreuung ein Gesamtbild des Systems.
Das im Kompetenzzentrum Secure Business Austria durchgeführte Projekt ATCERT ermöglicht Risikoanalysen in solchen heterogenen Netzwerken und soll Klein- und Mittelbetrieben ermöglichen, Informationen über Softwareschwachstellen zeitgerecht zu verarbeiten, und dadurch weitergehende Aufgaben wie das Einspielen von Updates und Patches durchzuführen. Dafür wird ein leicht erweiterbares Inventarisierungsprogramm zur Dokumentation der aktiven Computersysteme und der dort installierten Software benötigt. Diese Anforderungen versucht das im Zuge dieser Diplomarbeit entwickelte Programm *atc-inventory* zu erfüllen, indem es über definierte Schnittstellen verschiedene Programme in die Funktionalität der Applikation integriert und anhand dieser alle benötigten Daten sammelt und vereinheitlicht.

2 Aufgabenstellung und Ziel der Diplomarbeit

IT Asset Management ist heute bereits Teil vieler Standards im Bereich der IT Sicherheit. Obwohl verschiedene Ansätze und Implementierung existieren, gibt es im Bereich der Klein- und Mittelbetriebe noch viele zu lösende Probleme. In diesem Kapitel werden derzeit vorhandene Standards und best practices wie ITIL und ISO17799 kurz umrissen und verschiedene Ansätze und Methoden für die Inventarisierung beschrieben. Zudem werden als Lösungsansatz ATCERT, ein am Kompetenzzentrum Secure Business Austria durchgeführtes Projekt, vorgestellt und die Anforderungen und Spezifikationenan an dem zu entwickelnden *atc-inventory* Prototypen dargestellt.

2.1 Ausgangslage

Im industriellen Einsatz werden sehr oft IT Prozesse anhand best practices wie ITIL (IT Infrastructure Library) organisiert. ITIL wurde durch die OGC (Office of Governance Commerce) im Auftrag der britischen Regierung entwickelt, und ist der derzeitige weltweite de-facto-Standard im Bereich Service- und Systemmanagement. ITIL stellt einen öffentlich zugänglichen Leitfaden für die Planung, Erbringung und Unterstützung von IT Serviceleistungen dar, und bietet eine Grundlage zur Verbesserung der operationellen EDV Infrastruktur [Köh06]. Eine Kernkomponente von ITIL ist die Configuration Management Database (CMDB). Diese Datenbank stellt alle Informationen für die in ITIL definierten Prozesse bereit, beinhaltet die Daten der kompletten IT Infrastruktur und der darin angebotenen Dienste und dokumentiert die Verbindungen zwischen ihnen [BGSS06]. Zu den Daten der IT Infrastruktur zählen neben der eingesetzten Hard- und Software auch die Dokumentation, die Konfiguration und die Namen der Betreuer der jeweiligen Anlagen. Das Konfigurationsmanagement beinhaltet 4 Punkte:

1. Identifikation: die Identifizierung und Spezifikation aller EDV Komponenten und die Eingabe in die CMDB

2. Kontrolle: die Betreuung der einzelnen Komponenten und das Benutzer- und Berechtigungsmanagement
3. Status: der Status der einzelnen Komponenten
4. Überprüfung: die Überprüfungen und Beurteilungen zur Sicherstellung der Aktualität der in der CMDB enthaltenen Daten

Mit der CMDB kann genau abgefragt werden, welche Geräte für welche Dienste im Einsatz sind. Ähnliche best practices wie ITIL sind das Grundschutzbuch [JB04] des deutschen Bundesamt für Sicherheit in der Informationstechnik und der ISO17799 Standard [TFHC03]. Bei ISO17799 ist ähnlich der ITIL die Inventarisierung der vorhandenen Komponenten und der Betriebssysteme ein benötigter Bestandteil. Die Dokumentation zu dieser best practice geht aber ebensowenig wie die Dokumentation zu ITIL auf die Art und Weise der Durchführung der Inventarisierung ein. Weiters gibt es bei ISO17799 derzeit noch Lücken im Bereich des Patchmanagements [BS03]; es wird nicht genügend auf Sicherheitsprobleme bei installierten Applikationen eingegangen, der Standard deckt nur periodische Änderungen des Betriebssystems ab.
Alle diese best practices haben gemein, daß sie zwar beschreiben, was für die Sicherheit in der IT Umgebung wichtig ist, jedoch nicht auf die Implementierung dieser eingehen. Für die Implementierung des Asset Managements gibt es verschiedene in der Industrie verbreitete Softwarelösungen. Diese Programme sind jedoch in der Regel kommerziell und kostenpflichtig, oder sie decken nur einen Teil der benötigten Anforderungen ab.
Netzwerke und aktive Komponenten im Netzwerk können heute über verschiedene Mechanismen inventarisiert werden. Alle diese Mechanismen implementieren eine bzw. eine Mischung der zwei Paradigmen *agentbased* und *agentless*. Beim agentenbasierenden Ansatz wird am zu inventarisierenden Host eine Agentensoftware installiert, die die Daten an einen zentralen Serverdienst schickt. Im Gegensatz dazu wird beim agentenlosen Ansatz der zu inventarisierende Host über standardisierte Schnittstellen wie z.B. SNMP [CFSD90], WBEM [2] oder Zugriffsdienste wie SSH (Secure Shell) direkt vom Server aus abgefragt. Beide Ansätze haben Vor- und Nachteile. Der erste Ansatz bedeutet einen großen Managementaufwand, bringt jedoch eine große Flexibilität durch die Möglichkeit, tief in das System einzugreifen. Der zweite Ansatz weist keinen so großen Managementaufwand auf, jedoch können nur die über eine Remote Schnittstelle verfügbaren Informationen abgefragt werden. Laut NIST, dem National Institute of Standards and Technology gibt es bei den beiden Ansätzen folgende Vor- und Nachteile [MBH05]:

Vorteile des agentenlosen Ansatzes

- keine Installation am zu inventarisierenden Computer

- große Flexibilität

Nachteile des agentenlosen Ansatzes

- für die Überprüfung wird eine große Bandbreite im Netzwerk benötigt
- benötigt möglicherweise Netzwerkports, die ansonsten aufgrund diverser Sicherheitsrichtlinien geschlossen sind (z.B. Remote Procedure Calls (RPC) unter Unix, administrative Freigaben und RemoteRegistry Dienst unter Windows)
- die Überprüfung kann bei großen und verteilten Netzwerken lange dauern
- bei Computern mit einer installierten und nicht korrekt konfigurierten Personal Firewall können die Ergebnisse fehlerhaft bzw. nicht aussagekräftig sein
- der für die Überprüfung verwendete zentrale Server muß Zugriff mit Administratorenrechten bzw. Root-Rechten auf den zu überprüfenden Computern haben

Vorteile des agentenbasierenden Ansatzes

- Möglichkeit der kompletten Administration des Computers
- Interoperabilität mit 3rd Party Produkten
- Möglichkeit, große Netze mit einem geringen Zeitaufwand zu inventarisieren
- die Netzwerkbandbreite wird nicht ausgelastet
- Möglichkeit zur Durchführung von low-level Funktionen

Nachteile des agentenbasierenden Ansatzes

- die Agentensoftware muß auf allen Geräten installiert sein. Ist der Agent auf dem zu inventarisierenden Computer nicht gestartet oder nicht richtig konfiguriert, wird dieser Host auch nicht aufgenommen
- die Agentensoftware muß mit Administrator- bzw. mit root-Rechten laufen. Dies ermöglicht neue Angriffsmöglichkeiten auf die einzelnen Computer

Um die Vorteile beider Ansätze in ein Produkt zu integrieren, werden in Softwarelösungen immer öfters beide Möglichkeiten implementiert. Benötigt man tiefergehende Informationen und die Möglichkeit, am Computersystem direkt einzugreifen, kommt eine Agentensoftware zum Einsatz. Ansonsten nutzt man den geringen Initialaufwand der agentenlosen

Abfrage des Clients.
Durch die steigenden Investitionen in den IT Security Bereich und Vorgaben vieler Konzerne zur Einhaltung diverser best practices wie ITIL, gibt es eine große Anzahl verschiedener Asset Management Softwarelösungen am Markt. Diese sind entweder Teile komplexer Lösungen wie IBM Tivoli, HP Openview, CA Unicenter, oder implementieren jeweils nur Teile der benötigten Funktionalität für ein gesamtheitliches Sicherheitsmanagement. Zudem gibt es kaum Asset Management Lösungen für heterogene Systemumgebungen im Open-Source Bereich. Es gibt jedoch sehr viele Applikationen, die einzelne Teilbereiche der benötigten Funktionalität abbilden. Meist sind mehrere solcher Applikationen parallel im Einsatz. Jede einzelne Software inventarisiert und kontrolliert einen Teil der sich im Netzwerk befindlichen Computersysteme.
Durch die Integration all dieser Lösungen in eine zentrale Applikation und die Entwicklung von Methoden zur Vereinheitlichung der von den einzelnen Programmen gesammelten Daten, könnte ohne die Installation komplexer oder teuerer Gesamtlösungen die benötigte Funktionalität für ein gesamtheitliches Sicherheitsmanagement implementiert werden.
Ein funktionierendes Asset Management ist auch die Basis für die Durchführung von Risikoanalysen. EDV Systeme sind heute für fast jedes Unternehmen businesskritisch und dementsprechend ist Risikoabschätzung inzwischen ein wichtiger Bestandteil bei der Implementierung von IT Sicherheit und Teil diverser best practices. Im Gegensatz zu Großunternehmen und Konzernen haben Klein- und Mittelbetriebe selten eine eigene auf IT Sicherheit spezialisierte EDV Abteilung. Meist sind in solchen Unternehmen IT Verantwortliche nicht in der Lage, ein effizientes IT Sicherheitsmanagement zu implementieren. Die Hauptgründe dafür liegen in [4]

- niedriger EDV Budgets, sowohl relativ zum kompletten Budget als auch absolut,
- dem Fehlen von Expertenwissen, da EDV Abteilungen entweder klein sind oder die Aufgaben von anderen Abteilungen durchgeführt werden,
- der Annahme, die EDV sei nicht unternehmenskritisch, obwohl immer mehr Unternehmensprozesse von der EDV abhängen, und
- heterogenen EDV Umgebungen.

2.2 Secure Business Austria (SBA)

Secure Business Austria beschäftigt sich unter anderem mit Lösungsmöglichkeiten für diese Probleme. Secure Business Austria wurde 2006 im Rahmen des "K-ind"Programms

des BMWA in Kooperation mit der Stadt Wien gegründet und sein Schwerpunkt liegt in den organisatorischen und technischen Aspekten der IT Sicherheit. Das Kompetenzzentrum hat als Hauptziel, zusammen mit verschiedenen akademischen und industriellen Partnern, Spitzenforschung in angewandten Bereichen der IT Security zu betreiben. Die Partner von SBA sind neben Großunternehmen österreichische auf IT Security spezialisierte Klein- und Mittelunternehmen. Daneben hat SBA Kooperationen mit diversen akademischen Institutionen wie dem Institut für Softwaretechnik und interaktive Systeme der Technischen Universität Wien, dem Institut für angewandte Informationsverarbeitung und Kommunikation der Technischen Universität Graz und der Abteilung für Knowledge Engineering der Universität Wien. Als Forschungsmitarbeiter kann SBA zudem auf diverse Wissenschaftler anderer Institutionen zurückgreifen.
Zur Forschung von Secure Business Austria zählen neben sicheren Geschäftsprozessen, Kosten-Nutzenanalysen von IT-Sicherheitsmaßnahmenbündel und Compliancefragen auch Penetrationstechnologien und deren Vorbeugung, Digitale Forensik sowie Erkennung von schädlichem Programmcode.
Der wichtigste strategische Bereich von SBA ist ATCERT, eine Sammlung von neuen Technologien, anhand derer IT Sicherheitswarnungen schnell und effizient für die Zielpersonen aufbearbeitet werden können. Aufbauend auf semantische Technologien und Security Ontologien versucht SBA innovative und praxistaugliche Prototypen zu diesem Thema zu entwickeln und die Probleme bei Klein- und Mittelbetrieben zu lösen. Eine Security Ontologie kann in vielen Fällen bei der Implementierung eines EDV Sicherheitskonzepts helfen. Die Security Ontolgie dient als Basis für das Sicherheitskonzept und hilft die sicherheitsrelevanten Begriffe und deren Abhängigkeiten zu klären und zu definieren [Don03]. Sehr oft sind die Begriffe im Bereich IT Sicherheitsmanagement sehr vage definiert. Ohne entsprechende Terminologie ist die Kommunikation schwierig und fehleranfällig. Eine Ontologie kann hier hilfreich sein. Weiters werden Entscheidungen meist von Managern oder leitenden Mitarbeitern intuitiv ohne tiefgehendes Wissen über die IT Infrastruktur und ohne vorherige Kosten-Nutzen-Analyse getroffen. Um eine vollständige Sicherheitsrichtlinie und ein effizientes Sicherheitskonzept zu entwickeln, werden jedoch genau diese Informationen benötigt. Zur Unterstützung wird dementsprechend oft auf Normen und best practices zurückgegriffen. Diese best practices bringen jedoch keine Hilfsmittel zur Visualisierung und Simulation der kompletten IT Infrastruktur mit und benötigen zudem den Einsatz erheblicher Ressourcen. Dementsprechend schwierig und fehlerbehaftet kann die Entscheidungsfindung von Managern [EFKW07] sein. Durch die Implementierung einer Security Ontologie können diese Probleme gelöst werden. Folgende Definition trifft den Begriff Security Ontologie für das bei Secure Business Austria durchgeführte Projekt am besten [EFKW07]:

"An ontology defines the basic terms and relations compromising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary." [GPFLC04]

Secure Business Austria integriert in ihre Security Ontologie zusätzlich Abhängigkeiten zu Bedrohungen, zu Gegenmaßnahmen und zu Ressourcen wie die Betriebsmittel. Dies ist nötig, damit die Klein- und Mittelbetriebe Risikoanalysen kostengünstig ohne teuere Sicherheitsaudits wie ISO17799 durchführen können.
Die Security Ontologie besteht aus drei Teilen [EFK+06]. Der erste Teil beinhaltet die Sicherheits- und Abhängigkeitsklassifizierung nach Landwehr [ALRL04], der zweite Teil beschreibt das Konzept der IT Infrastruktur und der dritte Teil gibt den Unternehmen die Möglichkeit, ihre Infrastruktur, ihre Ressourcen und ihre Mitarbeiter abzubilden. Die Ontologie ist in der OWL (Web Ontology Language) entwickelt.

2.3 Aufgabenstellung und Ziel der Diplomarbeit

Das Projekt ATCERT von Secure Business Austria basiert auf der oben beschriebenen Security Ontologie. Secure Business Austria versucht mit ATCERT die Schwachstellen klassischer CSIRT Ansätze zu beheben, indem es Warnungen um semantische Informationen erweitert und dadurch Automatisierungen und benutzerspezifische Filterungen ermöglicht. Computer Security Incident Response Teams (CSIRT) sind Organisationen, die Computerschwachstellen erkennen, Informationen dazu sammeln, diese analysieren und entsprechende Warnungen und Empfehlungen bereitstellen. Durch die Implementierung eines semantischen CSIRT wird Klein- und Mittelbetrieben die Möglichkeit gegeben, auf die ständig steigende Anzahl von Warnungen rechtzeitig zu reagieren, indem diese vorgefiltert werden. Daher müssen Unternehmen nur mehr die für sie relevanten Sicherheitsmeldungen bearbeiten. Die Architektur von ATCERT ist in Abbildung 2.1, entnommen der ATCERT Projektseite, dargestellt. Die Eingabe der Unternehmensinfrastruktur in die Ontologie kann auf zwei Arten erfolgen. Die Daten können händisch eingegeben werden. Oder Sensoren und Detektoren sammeln die Daten und speichern sie in der Security Ontologie ab.
Da der Einsatz von ATCERT auf Klein- und Mittelbetriebe fokussiert ist, müssen effiziente und für Betriebe dieser Größe optimierte Sensoren und Detektoren zum Einsatz kommen. Hier stellt sich die Frage, wie diese Infrastrukturdaten effizient für Klein- und Mittelbetriebe gesammelt werden können. Zudem muß beantwortet werden, welche In-

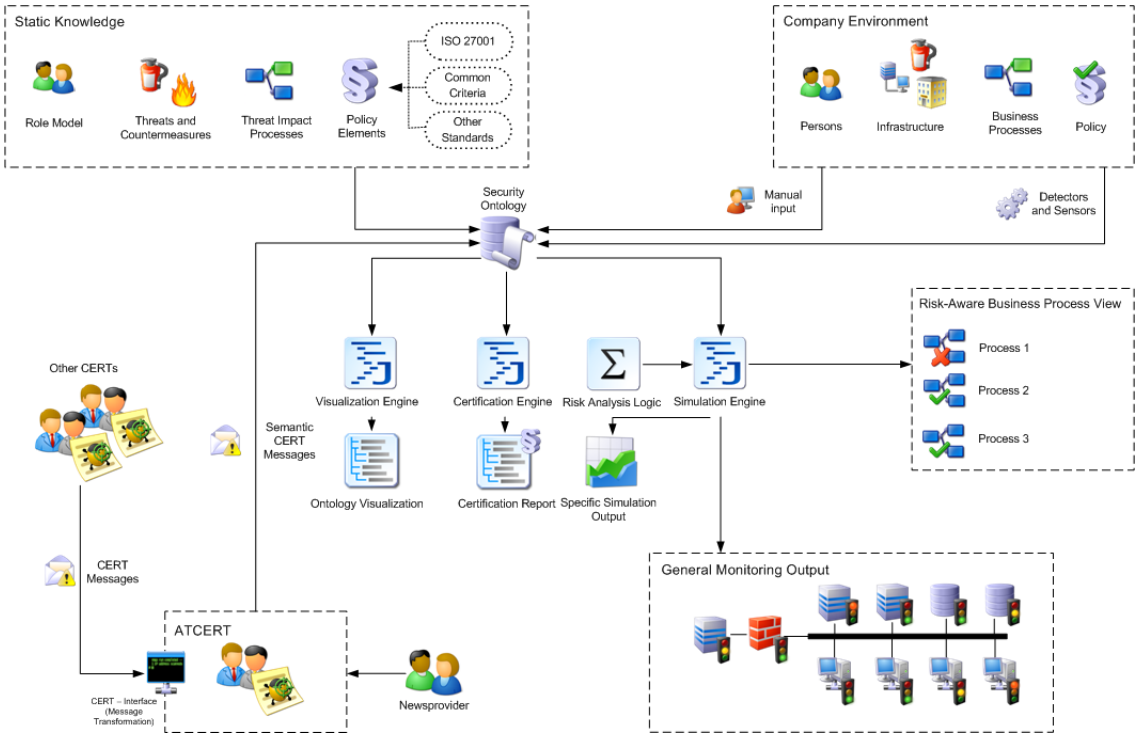

Abbildung 2.1: Architektur von ATCERT

frastrukturdaten benötigt werden und ob es bereits optimierte Lösungen für Klein- und Mittelbetriebe gibt, die in das Projekt ATCERT integrierbar sind.

Ziel der Diplomarbeit ist die Analyse verschiedener sich am Markt befindlicher Applikationen und die Spezifikation von Sensoren, die den Anforderungen von Klein- und Mittelbetrieben genügen und die von ATCERT benötigten Daten liefern. Da der Einsatz auf Klein- und Mittelbetriebe fokussiert ist, müssen die Sensoren und Detektoren mit einem für Klein- und Mittelbetriebe vertretbaren Aufwand implementierbar sein und es dürfen keine zusätzlichen Lizenzkosten auftreten. Zudem muß sowohl das Datenmodell als auch die Funktionalität der Sensoren erweiterbar sein, da die Sensoren die Infrastrukturen unterschiedlicher Betriebe inventarisieren müssen. Um die Anforderungen von Klein- und Mittelbetrieben abzudecken, soll in erster Linie versucht werden, bereits bestehende Software- und Managementlösungen als Sensoren zu verwenden. Daher sollen die Stärken des entwickelten Konzepts in der Erweiterbarkeit und in der Einbindung von verschiedenen Sensoren für die Datenerfassung liegen.

Aufgrund dieser Spezifikation soll ein Prototyp entwickelt und implementiert werden. Als zu überprüfende Plattform wird aufgrund der Verbreitung Microsoft Windows gewählt. Anhand des Prototypen soll die Effizienz des gewählten Ansatzes getestet und bewertet werden. Für den Test des Prototypen müssen aussagekräftige Messwerte definiert werden,

die die Effizienz für den Einsatz bei Klein- und Mittelbetrieben bewerten. Für die Bewertung wird der Prototyp im Netzwerk der Universität Akademie der bildenden Künste Wien getestet.

3 Bestehende Ansätze

Die Überprüfung des Netzwerks und der aktiven Netzwerkkomponenten wie Computer, Switches und Router ist essentiell für verschiedene Aspekte des Sicherheitsmanagements. Daher gibt es bereits eine große Anzahl von Ansätzen und Produkten, mit denen dieser Vorgang organisiert und durchgeführt werden kann. Dieses Kapitel beschreibt mehrere Applikationen und Lösungen zu diesem Thema. Es werden Lösungen aus dem akademischen Umfeld und kommerzielle Produkte vorgestellt.

3.1 NetInventory System

Im wissenschaftlichen Bereich sind hauptsächlich Algorithmen und Applikationen entwickelt worden, welche Netzwerktopologien dokumentieren und in diesem Bereich Optimierungen ermöglichen. Durch die Größe heutiger IP Netzwerke ist es sehr zeitaufwendig, wenn nicht unmöglich, Änderungen in Netzwerken ohne Automatismen zu dokumentieren. Daher sind effektive Algorithmen nötig. Bisher haben die verschiedenen Arbeiten sich entweder auf

1. die logische Topologie (Layer 3) konzentriert. Das bedeutet aber, dass Geräte wie Switches und Bridges teilweise nicht erkannt werden. Oder

2. die Lösung war auf eine gewisse Produktfamilie spezialisiert und damit herstellerabhängig.

Im Artikel *Topology Discovery in Heterogeneous IP Networks: The NetInventory System* [BGJ+04] wird ein Algorithmus vorgestellt, der die physikalische Netzwerktopologie in heterogenen Netzwerken erkennen und inventarisieren kann. Der Algorithmus basiert auf den von einem Großteil der Netzwerkhersteller unterstützten SNMP-MIB Informationen und benötigt keine weiteren Konfigurationen auf den Geräten. Da dieser Algorithmus zur Erkennung der aktiven Netzwerkkomponenten zwingend SNMP voraussetzt, funktioniert es bei Netzwerkkomponenten wie Switches, Bridges, Wireless LAN Access Points, Routern usw. problemlos. Bei Servern und Computern ist jedoch standardmäßig

kein SNMP installiert. Daher eignet sich dieses System nur, die aktiven SNMP-fähigen Netzwerkkomponenten zu erkennen, nicht aber alle aktiven Netzwerkknoten. Zudem kann der Algorithmus hinsichtlich einer Softwareinventarisierung nicht erweitert werden.

3.2 NetMap

Eine wissenschaftliche Arbeit, die sich genau mit dieser Problematik beschäftigt, wird im Artikel *Composable Tools For Network Discovery and Security Analysis* [VVZK02] beschrieben, und stellt das Programm NetMap vor. Diese Applikation ermöglicht die automatische Erkennung der aktiven Netzwerkknoten und eine Sicherheitsanalyse der erkannten Geräte. NetMap basiert auf einem Modell, das neben dem zu analysierenden Netzwerk auch die Programme für die Analyse beinhaltet. Beim Design des Modells wurden bestehende Programme für das Netzwerkmanagement und die Schwachstellenanalyse miteinbezogen. So verwendet NetMap unter anderem Nmap, das neben ping und port scans auch ein OS fingerprinting und die Möglichkeit von RPC scans unterstützt. Die Autoren von NetMap haben sich für dieses modulare Design aufgrund der Probleme und Limitierungen der meisten sich am Markt befindlichen Produkte für die Netzwerk- und Computeranalyse entschieden. Programme wie Nessus ermöglichen Überprüfungen anhand von Netzwerkscans, andere Applikationen können Netzwerkknoten anhand von SNMP oder WMI erfassen, aber die meisten Applikationen können nicht beides. Zudem sind die Speichermethoden der Applikationen großteils auf die Applikation zugeschnitten und nicht immer strukturiert. Manche Programme verwenden Datenbanken, manche XML und manche unstrukturierte Textdateien. Damit sind die wenigsten dieser Applikationen flexibel und können keine Daten mit anderen Applikationen austauschen.
Das Modell von NetMap beinhaltet Informationen über die Netzwerktopologie, die Infrastruktur, die jeweiligen Betriebssysteme, die eingesetzte Hardware, den physischen Standort, die angebotenen Netzwerkdienste und das Konzept ist für Erweiterungen offengehalten. Neben den genannten Informationen bildet es die Beziehungen zwischen den einzelnen Geräten und Diensten ab. Das Modell von NetMap unterstützt eine abstrakte Beschreibung einer Reihe von Netzwerkerkennungs- und Analyseprogrammen. Jedes dieser Programme ist für eine genau definierte Aufgabe zuständig. Alle Informationen für die erfolgreiche Durchführung der Aufgabe sind spezifiziert. Dazu gehören die Eingabe, die Ausgabe, die Kosten in Form einer Priorität und die Art der Durchführung. Durch die genaue Spezifikation der einzelnen integrierten Programme und deren Kosten kann bei der Inventarisierung eines Computersystems der *Query Processor* von NetMap selbständig entscheiden,

- welche Informationen für die Inventarisierung benötigt werden,
- welche Programme für diese Aufgabe verwendet werden können,
- anhand welcher Ein- und Ausgaben einzelner Programme die Aufgabe gelöst werden kann.

Dieses Verhalten ermöglicht ein extrem flexibles und einfach erweiterbares Programm, das problemlos eigene oder 3rd-party Applikationen integrieren und komplexe Aufgaben, die die Funktionalität mehrerer Applikationen benötigen, ausführen kann. Die Architektur von NetMap ist in Abbildung 3.1, entnommen dem Artikel *Composable Tools For Network Discovery and Security Analysis* [VVZK02], Seite 4, abgebildet.

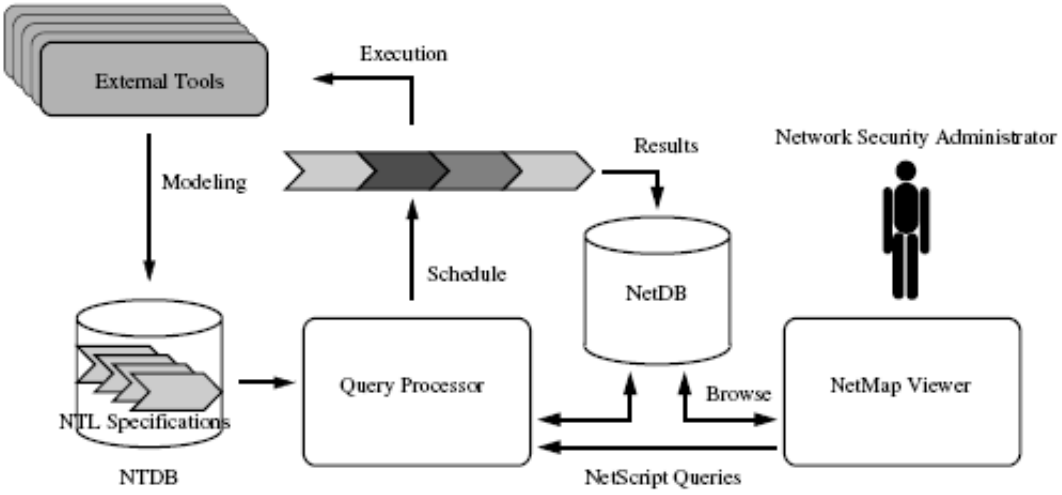

Abbildung 3.1: Architektur von NetMap

Die integrierten Programme sind anhand der *NTL (Network Tool Language)* spezifiziert und in der *NTDB (Network Tool Database)* abgespeichert. Der Systemadministrator kann die Daten über die *NetDB* abfragen, und neue Abfragen anhand der *NetScript* Sprache durchführen. Diese werden der *Query Processor* Komponente übergeben, die die dafür idealen Programme auswählt, ausführt und die Ergebnisse wiederum in der *NetDB* speichert. NetMap enthält folgende Kompontenten:

- die *NetDB*: in der Datenbank werden die gesammelten Informationen des überprüften Netzwerks und der überprüften Geräte gespeichert.
- die *NTDB*: in der *Tool Repository* Datenbank befinden sich die Definitionen der in NetMap integrierten Applikationen.
- die *Network Tool Language*: mit der *NTL* werden die integrierten Programme, deren Effizienz und die dazu definierten Parameter beschrieben.

- die *NetScript Query Language*: anhand dieser Sprache werden alle Erkennungs- und Analyseaufgaben durchgeführt. Dies beinhaltet simple Topologiedokumentationsaufgaben, als auch komplexe Sicherheitsanalysen der inventarisierten Geräte. Durch die Aufgabenbeschreibung erkennt der *Query Processor*, welche Programme er für die Ausführung verwenden muß, in welcher Reihenfolge die verschiedenen Hilfprogramme aufgerufen werden müssen, und welche Informationen in der Datenbank gespeichert werden sollen.

Das Netzwerkmodell ist in einer relationellen Datenbank abgebildet. Diese beschreibt die Topologie, die Struktur der angebotenen Dienste des Netzwerks und die Beziehung dazwischen. Interessant ist, daß neben der Topologie auch alle angebotenen Dienste (z.B. SMB Shares, HTTP Dienste, usw.) und die Beziehungen zwischen den Komponenten und Diensten mitaufgenommen werden. Das ermöglicht komplexe Angriffszenarien zu analysieren. Ein Beispiel dafür sind die eben genannten SMB Freigaben. Kann ein Angreifer über diese SMB Shares mit Trojanern oder Viren infizierte Dateien auf den zentralen Dateispeicherplatz legen, sind damit alle Computer, deren Benutzer auf den zentralen Datenspeicher zugreifen, betroffen. Damit ermöglicht die Applikation, die Ausmaße einfacher Attacken abzuschätzen, und weiterführende Aktionen auszuführen (z.B. die Wiederherstellung der Orginaldateien anhand einer Datensicherung).
Die Analyse und die Inventarisierung des Netzwerks wird anhand bestehender, spezialisierter Programme durchgeführt. Die Charakteristiken der integrierten Applikationen sind anhand der *Network Tool Language* definiert. Für jede Applikation sind Kosten definiert, diese Kosten setzen sich aus der Effizienz des Programms und Parametern wie z.B. der benötigten Bandbreite zusammen. Wird eine neue Anfrage mit der *NetScript* Sprache definiert, wählt der *Query Processor* anhand der Kosten eine Applikation und führt sie aus. Damit dies automatisiert ablaufen kann, muß zusätzlich zu den Kosten für jede Applikation eine Ein- und Ausgabe definiert sein. Ein Beispiel einer solchen Programmdefinition ist folgendes Programmlisting:

```
1  tool ping {
2          ipsetup s;
3
4          // Definition Ein- und Ausgabe
5          input s.ipaddress;
6          output s.status;
7
8          // Kostendefinition
9          efficiency = 2;
10         confidence = 9;
11
12         // Definition der Funktionalität
13         code {
```

```
14                  // Verwende die Pingfunktion von Nmap und überprüfe ob der Host aktiv ist
15                  nargs -n 1 -i "nmap -SP {} |
16                          if grep 'appears to be up.' > /dev/null ; then
17                                  echo 1;
18                          else
19                                  echo 0;
20                          fi"
21          }
22  }
```

Listing 3.1: Beispiel NetMap Definition Ping anhand der Software Nmap

Nach der erfolgreichen Durchführung der für die Anfrage benötigten Programme normalisiert der *Query Processor* die Daten und speichert sie in der *NetDB*. Trotzdem können Inkonsistenzen wie sogenannte *ghost entries* in der Datenbank auftreten. Bei *ghost entries* repräsentieren mehrere Einträge in der Datenbank das selbe Netzwerkobjekt. Zum Auflösen dieser Inkonsistenzen stellt NetMap eigene Prozeduren bereit.
Derzeit stehen zwei Frontend-Prototypen für System- und Netzwerkadministratoren zur Verfügung: eine tkinet-basierende Version und eine Webapplikation. Beide ermöglichen die *NetDB* zu durchsuchen, neue Systeme zu erfassen, bestehende Systeme zu aktualisieren und *NetQuery* Skripte zu definieren.
Das interessante an dem Ansatz von NetMap ist seine Flexibilität. Die Applikation kann durch die Integration weiterer spezialisierter Programme sehr einfach erweitert werden. Solche Hilfsprogramme können selbst entwickelte Softwarelösungen oder auch third-party Produkte sein. Das Schema der zugrundeliegenden Datenbank *NetDB* ist herstellerunabhängig definiert und kann relativ einfach erweitert werden.

3.3 Integrated Vulnerability Management System (IVM)

In *Integrated Vulnerability Management System for Enterprise Networks* [WYYR05] wird ein Schwachstellenmanagementsystem beschrieben, das als Basis verschiedene Datenquellen verwendet. Ausgehend von den verschiedenen Standards und Technologien wie das Instrusion Detection Message Exchange Format (IDMEF), das Common Information Model (CIM), das Common Vulnerabilites and Exposures dictionary (CVE), die Open Vulnerability Assessment Language (OVAL), schlagen die Autoren das *Security Vulnerability Information Model* (SVIM) vor. Dieses Modell beinhaltet jedoch nicht wie bei NetMap die Verwendung verschiedener Programme für die Datensammlung, sondern implementiert verschiedene Technologien für die Datenauswertung. Damit eignet sich das Modell aufgrund der Integration von Standards, der Flexibilität, der Erweiterbarkeit und

der Portabilität für verschiedene industrielle Bereiche. Das Modell besteht aus drei Modulen:

1. das *Vulnerability Information Collection (VIC)* Modul sammelt sicherheitsrelevante Informationen der einzelnen zu inventarisierenden Geräte anhand dem vorgeschlagenen standardisierten *Security Vulnerability Information Model*.

2. das *Vulnerability Assessment (VA)* Modul wertet die gesammelten Daten aus. Die Auswertung basiert auf dem *Vulnerability Index Assessment (VIA)* Modell.

3. das *Vulnerability Resolution (VR)* Modul analysiert die Daten des Assessment Moduls und erstellt die Dokumentation.

Die Modell beinhaltet eine Verbindung zu verschiedenen externen Datenquellen wie z.B. dem CVE Repository, den OVAL Definitionen und verschiedenen Antiviren Datenbanken. Diese Informationen werden für die Auswertung der gesammelten Daten verwendet. In Abbildung 3.2 sind die einzelnen Module abgebildet (die Abbildung ist dem Artikel *Integrated Vulnerability Management System for Enterprise Networks* [WYYR05], Seite 3, entnommen).

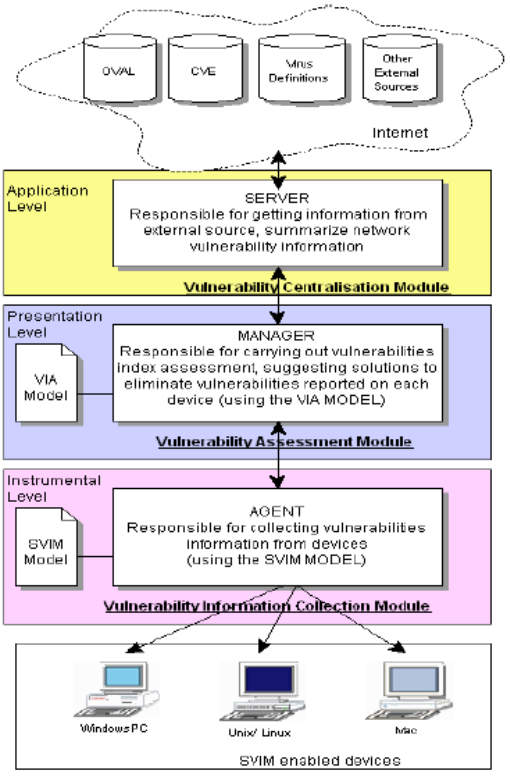

Abbildung 3.2: Integrated Vulnerability Management System (IVM) Module

Da das vorgeschlagene Modell SVIM auf den Definitionen von OVAL basiert, verwenden die Autoren als Basis für die Agentensoftware (SVIM Agent) den von Mitre freigegebenen OVAL Interpreter. Der OVAL Interpreter sammelt die gewünschten sicherheitsrelevanten Daten auf den zu untersuchenden Computersystemen. Neben den OVAL Definitionen können zusätzlich eigene Sicherheitsdefinitionen dem Agenten übergeben werden, diese Daten fließen auch in das SVIM Schema mit ein (siehe Abbildung 3.3, entnommen dem Artikel *Integrated Vulnerability Management System for Enterprise Networks* [WYYR05], Seite 4).

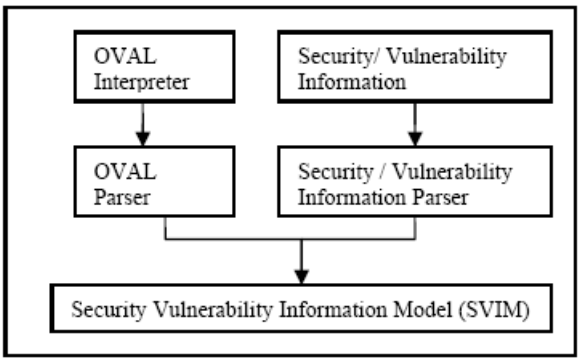

Abbildung 3.3: Datenquellen des SVIM Modells

Der im Artikel vorgeschlagene SVIM Agent implementiert einen Dienst am Computersystem, mit dem die sicherheitsrelevanten Informationen in die standardisierte Struktur überführt werden. Die Konfiguration des Agenten erfolgt mittels einer XML Datei. Ab dem Zeitpunkt der Installation des Agenten können über eine verschlüsselte Verbindung alle sicherheitsrelvanten Informationen wie die Betriebssystemversion, installierte Antivirenprogramme und -definitionen, eine konfigurierte Firewall, installierte Softwareupdates und -patches, Systemlogdateien, laufende Prozesse und offene Netzwerkports vom Host abgefragt werden. Die Agentensoftware sammelt die gewünschten Daten und sendet diese dann über die verschlüsselte Verbindung an den Server, der die Daten weiterverarbeitet. Die SVIM Agentensoftware muß auf allen zu inventarisierenden Computersystemen installiert werden, erst dann besteht die Möglichkeit die sicherheitsrelevanten Daten abzurufen. Die Daten werden in dem SVIM XML Schema gespeichert.

```
1 <SVIM version="0.1">
2         This is a simple SVIM Model
3         <OperatingSystem>Microsoft Windows 2000
4                 <Build>2021</Build>
5                 <ServicePack>SP4</ServicePack>
6         </OperatingSystem>
7         <VirusScanner>Mcafee Virus Enterprise
```

```
8                   <Vendor>Mcafee</Vendor>
9                   <Version>4.51</Version>
10                  <VirusDefLastUpdated>18-09-2004</VirusDefLastUpdated>
11                  <VirusDef>4351
12                  <Date>20-09-2004</Date>
13                  </VirusDef>
14          </VirusScanner>
15          <Ports>
16                  <OpenPort>21</OpenPort>
17                  <OpenPort>80</OpenPort>
18          </Ports>
19  </SVIM>
```

Listing 3.2: Beispiel SVIM XML Schema

Zu den großen Vorteilen dieses Ansatzes zählen seine Flexibilität und das breite Spektrum der unterstützten Daten und externen Datenquellen. Jedoch gibt es auch einige nicht zu vernachlässigende Nachteile. So muß die Agentensoftware für die jeweiligen Betriebssysteme adaptiert werden. Ein noch größerer Nachteil besteht jedoch in der Notwendigkeit, daß bei jeder neu gefundenen Schwachstelle die Agentensoftware angepasst werden muß.

3.4 Open Vulnerability and Assessment Language

Die *Open Vulnerability ans Assessment Language* (OVAL) ist ein internationaler Standard für Computer- und IT Sicherheit. Der Standard versucht mit offenen und öffentlich zugänglichen Inhalten, den Austausch von sicherheitsrelevanten und -kritischen Informationen zu vereinheitlichen. OVAL umfasst das komplette Spektrum sicherheitsrelevanter Themen im Bereich Informationstechnologie. Die großen Vorteile von OVAL sind [10]:

- ein einfacher und standardisierter Ansatz zum Überprüfen von Schwachstellen, Konfigurationen und Patchständen auf Computersystemen
- Standardschemas, die alle notwendigen sicherheitsrelevanten Konfigurationsinformationen beinhalten
- ein einzelnes XML Dokument, das die Details der zu überprüfenden Konfigurationen oder Schwachstellen beinhaltet
- eine offene und weiterverwendbare Alternative zu proprietären Systemen
- wird durch eine große Gruppe von Sicherheitsexperten, Systemadministratoren und Softwareentwicklern unterstützt

- industrieunterstützt und -fokussiert durch das OVAL Board und das OVAL Developers Forum

Das Ziel von OVAL ist es, die Interoperabilität und den Datenaustausch zwischen verschiedenen Sicherheitsprodukten mit einer standardisierten XML Schnittstelle zu gewährleisten. Das ermöglicht neben einzelnen Applikationen zusätzlich andere Programme für spezielle Aufgaben einzusetzen und die Daten miteinander weiterzuverarbeiten.
OVAL ist in drei verschiedenen Schemen aufgeteilt; jedes Schema repräsentiert eine der drei Phasen zur Überprüfung von Computersystemen. Hersteller können damit nur jenen Teil implementieren, der für ihr Produkt relevant ist. Durch diese Aufteilung können z.B. Sicherheitsbulletins erstellt, Schwachstellenanalysen und Systeminventarisierungen durchgeführt oder Patch- und Konfigurationsmanagementlösungen entwickelt werden. Es ist auch möglich, ein umfassendes Security Information Management System (SIMS) zu implementieren.
Wie bereits erwähnt umfasst die OVAL Spezifikation drei Phasen:

OVAL Schema Characteristics Schema zum Sammeln der Konfigurationsdaten der zu überprüfenden Systeme

OVAL Definition Schema zum Testen ob die zu überprüfenden Systeme spezielle Konfigurationen, Schwachstellen usw. aufweisen

OVAL Results Schema zur Auswertung der Ergebnisse der untersuchten Systeme

Das *OVAL Schema Characteristics* Schema definiert ein standardisiertes XML Format zum Abbilden von Systemkonfigurationsinformationen. Dazu gehören Informationen zum Betriebssystem, zur installierten Software und zu anderen sicherheitsrelevanten Konfigurationsparametern. Diese Informationen können für die Analyse mit den OVAL Definitionen verglichen werden. Das XML Schema kann jedoch auch verwendet werden, um die gesammelten Informationen mit anderen Applikationen auszutauschen. Damit müssen diese Programme nicht mehr die einzelnen Systeme inventarisieren, sondern können die weitergegebenen Daten verarbeiten. Ein Beispiel für eine solche Applikation ist der OVAL Interpreter, der z.B. im oben beschriebenen *Integrated Vulnerability Management System (IVM)* zum Einsatz kommt.
Das *OVAL Definition* Schema ist ein Framework zum Schreiben von OVAL Definitionen im XML Format. In den OVAL Definitionen sind spezielle Systemzustände definiert, anhand derer man Systeme automatisiert untersuchen kann. Dazu gehören z.B. Konfigurationen von Applikationen und des Betriebssystems oder das Vorhandensein von Softwareschwachstellen. Weiters können anhand der OVAL Definitionen Sicherheitsexperten technische Diskussionen z.B. über das Vorhandensein einzelner Sicherheitslücken führen.

Das Definitionsschema ist in zwei Teile aufgeteilt: dem allgemeinen Teil, der die grundlegenden Informationen definiert, und dem individuellen Teil, der auf die verschiedenen Betriebssysteme bzw. Applikationen eingeht.
Im *OVAL Results* Schema ist ein XML Schema definiert, mit dem die Ergebnisse von untersuchten Computersystemen ausgewertet werden können. Das Ergebnis beinhaltet den momentanen mit den OVAL Definitionen verglichenen Status der Systemkonfiguration des analysierten Systems. Das *OVAL Results* Schema ermöglicht Applikationen das Ergebnis zu interpretieren, und zusätzliche Aufgaben zu erledigen. Dazu gehören z.B. das Installieren von Patches oder die Konfiguration einer Firewall.
Das interessante an OVAL ist der offene Ansatz. OVAL versucht, die Datensammlung und den Datenaustausch zu standardisieren. Das geht so weit, daß, falls applikationsspezifische Informationen in dem bestehenden XML Format nicht abgebildetet werden können, die Applikationshersteller das Schema diesbezüglich erweitern können. Ein weiterer interessanter Ansatz besteht meines Erachtens darin, daß von Anfang an sowohl Hersteller als auch Sicherheitsexperten und Softwareentwickler miteinbezogen werden. Das führt zu einer breiten Akzeptanz und zur Berücksichtigung der definierten Schemas bei der Implementierung von neuer Computersicherheitssoftware. Einen Nachteil sehe ich in der Einschränkung, daß nur jene Teile untersucht werden können, für die entsprechende OVAL Definitionen verfügbar sind. Das setzt voraus, daß die großen Softwarehersteller den Standard unterstützen und bei der Bekanntmachung von Softwareschwachstellen ebenfalls die zugehörigen OVAL Definitionen veröffentlichen.

3.5 Novell ZENworks Asset Management

Da die Hard- und Softwareinventarisierung viele Bereiche des IT Management beeinflußt, gibt es neben wissenschaftlichen und offenen Ansätzen auch etliche kommerzielle Anwendungen. Diese lizenzpflichtigen Anwendungen gehen von spezialisierten Systemen wie z.B. dem Patchmanagement bis hin zu Komplettlösungen, die alle Aspekte in diesem Bereich abdecken. Eine solche Komplettlösung ist Novells ZENworks Asset Management. Novell hat eine lange Tradition im Management von heterogenen Clientsystemen. So unterstützt die aktuelle ZENworks Asset Management Suite auch alle gängigen Betriebssysteme. Serverseitig können alle aktuellen Windows Server Versionen, die Linux Enterprise Versionen von SUSE und Red Hat, AIX, HP UX und Solaris inventarisiert werden. Im Clientbereich werden alle Windowsversionen seit Windows 95, Mac OS X seit 10.2.4 und die Linux Desktop Versionen von SUSE und Red Hat unterstützt. Weiters können über SNMP Switches, Router und Drucker mitaufgenommen werden.

Clientsysteme können über mehrere Methoden inventarisiert werden. Entweder wird ein Agent im zu überprüfenden Computersystem installiert (dies kann zentral vom Server aus gesteuert werden), oder die Daten werden mittels SNMP und WMI gesammelt. Dies setzt voraus, daß am Client WMI und SNMP konfiguriert und zugelassen ist. Bei SNMP ist dies jedoch weder bei Windows PCs, noch bei Apple und Linux Clients standardmäßig der Fall. Der Abruf der gewünschten Daten funktioniert zudem nur, wenn keine lokale Firewall installiert ist. Dementsprechend wird die agentenbasierende Lösung in der Regel bevorzugt. Der Benutzer kann über eine Weboberfläche alle Aufgaben ausführen. Inventarisierungen können automatisiert und zeitgesteuert oder bei Bedarf durchgeführt werden. Die Weboberfläche ermöglicht laut dem White Paper *Get Control of Your Software Assets* [11], neben den hunderten vordefinierten Auswertungen eigene Abfragen zu definieren. Weiters gibt es die Möglichkeit der Trendanalyse, da alle nicht mehr aktuellen Daten weiterhin in der Datenbank als Historie gespeichert bleiben.
Die zentrale Komponente von ZENworks Asset Management ist seine Datenbank mit den monatlichen Knowledgebase Updates. Diese Knowledgebase Datenbank ist die Basis für die Inventarisierung, und enthält eine Liste aller bekannten Applikationen. Der Inventarisierungsalgorithmus von Novell überprüft die Laufwerke und die Registry, analysiert mittels der Knowledgebase alle gefundenen Applikationen und binären Dateien und klassifiziert sie. Dadurch kann ZENworks Asset Management unterscheiden, ob und welche Applikationen installiert sind.
Novell sieht sein ZENworks Asset Management jedoch nicht primär als Sicherheits- oder Updatemanagementprogramm, sondern als Gesamtwerkzeug für alle unternehmensrelevanten Informationen bezüglich eingesetzter Software. Das Hauptziel ist es, Unternehmen die Möglichkeit zu bieten, die Ausgaben durch ein effizientes Lizenz-, Vertrags- und Softwaremanagement zu senken. Dementsprechend kann es meiner Meinung nach auch nur bedingt für die Implementierung eines gesamtheitliches Sicherheitsmanagements eingesetzt werden. Die gesammelten Daten würden sehr wohl alle Informationen bereitstellen, das Datenbankschema ist jedoch proprietär, und kann nicht für die Verarbeitung durch weitere Applikationen verwendet werden.

3.6 CA Unicenter Asset Management

Ähnlich dem ZENworks Asset Management plaziert CA sein Unicenter Asset Management auch als Gesamtwerkzeug für das IT Management. Laut dem Datenblatt von Unicenter Asset Management [1] unterstützt CA zusätzlich zu den gängigen Betriebssystemen auf Server- und Clientebene auch mobile Geräte wie PDA's. Neben der Inventarisierung

der Hard- und Software (agentenbasiert oder agentenlos über WMI) können detaillierte Auswertungen zu diesen Themen durchgeführt werden. Dies beinhaltet auch die Historie der Daten für Trendanalysen und für das Change Management, und die Möglichkeit von Risikoanalysen.
Da das Programm sich nahtlos in das CA Unicenter Framework integrieren lässt, kann anhand der gesammelten Daten ein effizientes Patchmanagement implementiert werden. Die Softwareinventarisierung analysiert über mehrere Verfahren (Datei- und Registryüberprüfung, Auslesen der Windows Installer Datenbank) die installierte Software und vergleicht sie mit der *Recognition Database*, die wie von CA hervorgehoben über 13.000 verschiedene Applikationen kennt. Zudem erkennt der Inventarisierungsalgorithmus das Betriebssystem inkl. Service Packs und installierten Updates. Neben der Hard- und Softwareinventarisierung implementiert CA ein Lizenz- und Vertragsmanagement und eine ITIL konforme CMDB. Hier sieht CA auch den Hauptfokus: ein Managementwerkzeug für die IT Organisation, das einen Überblick über die eingesetzten Mittel und Posten ermöglicht und strategische Entscheidungen unterstützt.
In der von CA bei CMP Research in Auftrag gegebenen Studie *The State of IT Asset Management (ITAM) in North America* [12] wird dies als zentraler Punkt hervorgehoben. Laut der Studie ist die Implementierung und Einführung eines IT Asset Management Systems im industriellen Bereich unumgänglich. Für diese Studie interviewte CMP Research 130 Leser von *Network Computing*, *InformationWeek*, *Optimize* und *Intelligent Enterprise*. Die befragten Personen waren hauptsächlich in den Bereichen Industrie, Gesundheitswesen und regierungsnahen Organisationen tätig, und bekleideten verantwortliche Positionen in den Bereichen Finanz und EDV. Fünfundsiebzig Prozent waren zudem in Unternehmen mit über 10.000 Mitarbeitern beschäftigt. Ein Großteil dieser Personen sieht ein Asset Management als zentralen Bestandteil des IT Managements, ohne dem Entscheidungen wesentlich erschwert werden. Damit das IT Asset Management effizient ist, muß es unternehmensweit und prozessorientiert implementiert sein. Zwei Drittel der befragten Personen sagten, der größte Nutzen von einem Asset Management sei zudem die Möglichkeit, die eingesetzten Mittel und deren Kosten zu berechnen und strategische Entscheidungen zu treffen. So können z.B. durch den Überblick der Softwarenutzung Entscheidungen im Lizenz- und Vertragsbereich getroffen werden. Zusammenfassend zeigt die Studie die Wichtigkeit von Asset Management, da heute wesentlich mehr Firmen solche best practices einsetzen, als es noch vor einigen Jahren der Fall war. Neu ist, daß dieser Bereich hauptsächlich von der Geschäftsführung bzw. von der IT Leitung organisiert und vorgegeben wird.
Die Lösung von CA wirkt wie ein komplettes System, das alle erdenklichen Aufgaben im Bereich Softwareinventarisierung und Weiterverarbeitung der Daten zulässt. Hier ist

anzumerken, wie bereits bei Novell, daß das System proprietär ist, und keine Weiterverarbeitung der Daten durch Drittapplikationen ermöglicht. Dazu kommen ebenfalls Lizenzkosten, die es für kleinere Unternehmen erschweren, die Gesamtlösung von CA zu implementieren.

4 Anforderungen von Klein- und Mittelbetrieben

Es gibt verschiedene Anforderungen an Managementlösungen im Bereich der IT Sicherheit für Klein- und Mittelbetriebe. Wichtig für die Implementierung solcher Lösungen ist ein effektiver Nutzen für die Unternehmen. Dementsprechend müssen Kosten-Nutzen-Analysen durchgeführt werden, die den Nutzen für Klein- und Mittelbetriebe zeigen und die anfallenden Kosten berechnen. Dieses Kapitel beschreibt die verschiedenen Anforderungen an IT Sicherheitslösungen für Klein- und Mittelbetriebe und berechnet anhand von Fallbeispielen die Kosten und den Nutzen solcher Lösungen.

4.1 Anforderungen an IT Sicherheitslösungen

Für einen effizienten Einsatz von Inventarisierungslösungen in Klein- und Mittelbetrieben muß die ausgewählte Software verschiedenen Anforderungen genügen. Klein- und Mittelbetrieben fehlt es unter Umständen an Expertenwissen im Bereich der IT Security, und es stehen auch nicht die dafür erforderlichen Mittel zur Verfügung [4]. Ohne dieses Expertenwissen ist es für solche Unternehmen schwierig, einen Gesamtüberblick über die zur Verfügung stehenden Möglichkeiten im EDV Bereich zu haben, und dementsprechend gibt es selten eine definierte Strategie für den Ausbau des Unternehmensnetzwerks. Obwohl die EDV für immer mehr Klein- und Mittelbetriebe unternehmenskritisch ist, wird in die Sicherheit und in die Verwaltung derselben nur bei dringendem Bedarf investiert. Dies führt zu kurzfristigen Entscheidungen und endet meist in sehr heterogenen Netzwerken ohne langfristige Strategie.
Softwarelösungen für den Mittelstand orientieren sich in der Regel an technischen, betriebswirtschaftlichen und personellen Bedingungen der einzelnen Unternehmen. Und diese sind gleich unterschiedlich wie die einzelnen Unternehmen selbst. Daher ist es schwierig, einen allgemeinen Ansatz für die Bewertung des Aufwands und der Kosten einer Softwareinventarisierung zu definieren und eine Standardlösung für Klein- und Mittelbetriebe zu entwickeln. Obwohl Managementlösungen wie eine Softwareinventarisierung im Regelfall den TCO (Total Costs of Ownership) der gesamten EDV optimieren [16],

kann die Implementierung einer solchen Lösung für Unternehmen dieser Größe einen höheren Aufwand als Nutzen bedeuten.
Anhand der Besonderheiten des Mittelstandes lassen sich folgende Anforderungen an IT Sicherheitslösungen feststellen:

- die Lösungen müssen auf das Unternehmen abgestimmt sein
- es gibt meist ein begrenztes Investitionsbudget, daher muß die Lösung mit einem Fixpreis implementierbar sein
- die Lösung muß einfach und schnell implementierbar sein
- die Lösung muß einfach und problemlos wartbar sein
- der Funktionsumfang muß an das Unternehmen angepasst sein.

Klein- und Mittelunternehmen führen im Gegensatz zu Großunternehmen meist keine ROI (Return on Investment) und TCO (Total Costs of Ownership) Analysen durch. Daher muß eine Inventarisierungslösung einen praktischen und offensichtlichen Nutzen mitbringen, der eine Zeit- und Kostenersparnis bei Unternehmensprozessen bedeutet. Eine Softwareinventarisierung kann in verschiedenen Unternehmensprozessen die Kosten verringern, da sie die Basis diverser Managementlösungen ist. Zum Beispiel kann durch ein Patchmanagement mit automatisierbaren Installationen von Updates der Aufwand für das EDV Personals minimiert werden. Ein weiteres Beispiel ist das Lizenzmanagement mit einer Optimierung der Lizenzkosten. Es ist jedoch schwierig, eine allgemeine Nutzen-Kostenrechnung zu erstellen, da der Nutzen in höchstem Maße von den Unternehmensprozessen der einzelnen Betriebe abhängt.
Sehr oft treffen Entscheidungsträger und Prozessmanager Entscheidungen im Bereich der IT Sicherheit in einer kosten- und gewinnneutralen Art, ohne diese Prozesse mit den Kernunternehmensprozessen in Verbindung zu setzen. Obwohl die IT Sicherheit an sich keinen Unternehmenswert mit sich bringt, ist diese Verknüpfung jedoch sehr wichtig, da Sicherheitsprobleme den Gewinn der Unternehmen beeinträchtigen können und schlechte Werbung für das Unternehmen bedeuten. Was wiederum zu indirekten Kosten wie dem Verlust von Kunden führen kann [NKB05]. Dieses Zusammenspiel der Prozesse zeigt die dem Artikel *Business Process-based Valuation of IT-Security* [NKB05] entnommene Abbildung 4.1.
Es gibt drei Arten von Sicherheitskosten:

1. Investitionskosten für die Implementierung eines definierten Sicherheitslevels
2. laufende Kosten zur Erhaltung des definierten Sicherheitslevels

3. Wiederherstellungskosten, die den Zeitaufwand und den finanziellen Aufwand für das Wiederherstellen des Systems nach einem Sicherheitseinbruch beinhalten

Die Kosten eines Systemausfalls bedeuten indirekte Opportunitätskosten, die anhand

- des Gewinnverlustes
- der Personalkosten
- und indirekter Kosten wie z.B. den Verlust von Kunden oder den Verlust der Reputation

berechnet werden können.

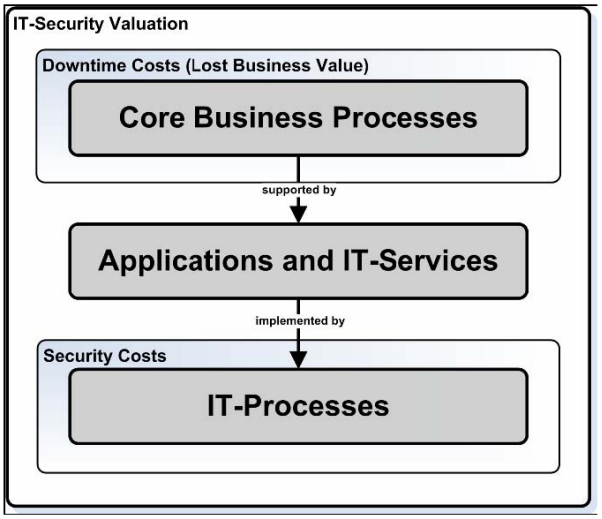

Abbildung 4.1: Zusammenhang zwischen den Kernunternehmensprozessen und den IT Prozessen

Obwohl es schwierig ist, den Nutzen der IT Sicherheit vom finanziellen Gesichtspunkt aus zu beziffern [And01], gibt es verschiedene Ansätze zum Berechnen der Kosten und des Nutzens. Dazu zählt zum Beispiel ALE (*Annual Loss Expectancy*) [oST79], eine quantitative Methode für die Risikoanalyse. Hierfür wird für jede Applikation die Anzahl des Auftretens eines Sicherheitsproblems berechnet, und die daraus entstehenden Konsequenzen monetär bewertet. Kritikpunkte dieses Ansatzes sind jedoch die Nichtberücksichtigung statistischer Daten über aufgetretene Fälle und der Konsequenzen [Mer03], und daß alle Sicherheitsprobleme die selben Kosten verursachen [NKB05].
Eine weitere Berechnungsmöglichkeit bietet CBA (*Cost-benefit analysis*) [Tho80]. Anhand der in der Mikroökonomie angesiedelten und auf die Theorie der Kostenrechnung basierenden *Cost-benefit analysis* Technologie können die Ausgaben und Aufwendungen

aufgrund des Wertes der zu schützenden Systeme berechnet werden. Dementsprechend ist dieser Berechnungsansatz applikationsunabhängig und ermöglicht die Identifizierung und das Messen des Nutzens und aller zugehörigen Kosten.

4.2 Kostenberechnung am Beispiel eines Patchmanagements

Am Beispiel eines Patchmanagementsystems, entnommen den von dem National Institute of Standards and Technology veröffentlichten Empfehlungen zum Erstellen eines Patch- und Schwachstellenanalysemanagements [MBH05], kann gezeigt werden, daß durchaus ein realer Nutzen durch die Implementierung von Sicherheitsmanagementlösungen entsteht.
Wird ein neuer Patch seitens eines Unternehmens nicht eingespielt, und bricht ein Wurm aus, der anhand dieser Sicherheitslücke die Computersysteme des Unternehmens zerstört, können die Kosten für die Wiederherstellung für das Unternehmen mit folgender Formel berechnet werden:

$$Kosten = W * R * T$$

W .. Anzahl der Computersysteme

T .. Zeit der Wiederherstellung des Computersystems und Verlust der Produktivität des Computerbenutzers

R .. Kosten pro Stunde

Bei einem Unternehmen mit 50 Computern, 8 Stunden Nichtverfügbarkeit des Rechners (4 Stunden für die Systemwiederherstellung, 4 Stunden für den Ausfall des Benutzers des nichtverwendbaren Systems) und angenommenen 70,– € pro Stunde, bedeutet dies für das Unternehmen einen Verlust von 28.000,– €.
Gegenübergestellt liegen die Kosten für das manuelle Überwachen und Aktualisieren der Computersysteme des Unternehmens. Das manuelle Überprüfen des Patchstandes eines Computers benötigt in etwa 10 Minuten des EDV Technikers, bei wiederum angenommenen Kosten von 70,– € pro Stunde bedeutet das pro Überprüfung eines PCs Kosten von 11,70 €. Bei 50 Computersystemen bedeutet das Überprüfen der installierten Patches bei Bekanntwerden einer neuen Sicherheitslücke jeweils ca. 583,– €. Die händische Installation eines Patches dauert in etwa 10 Minuten, das sind bei den angesprochenen 50 Computern nocheinmal 583,– €. Dementsprechend kann pro veröffentlichter Schwachstelle für das Überprüfen des Patchstandes und das Installieren eines fehlenden Patches

ein Gesamtaufwand von 1.166,–€ angenommen werden. Bei dem stetigen Wachstum der veröffentlichten Schwachstellen (Abbildung 4.2) bedeutet dies einen nicht zu vernachlässigenden Aufwand für Klein- und Mittelbetriebe.

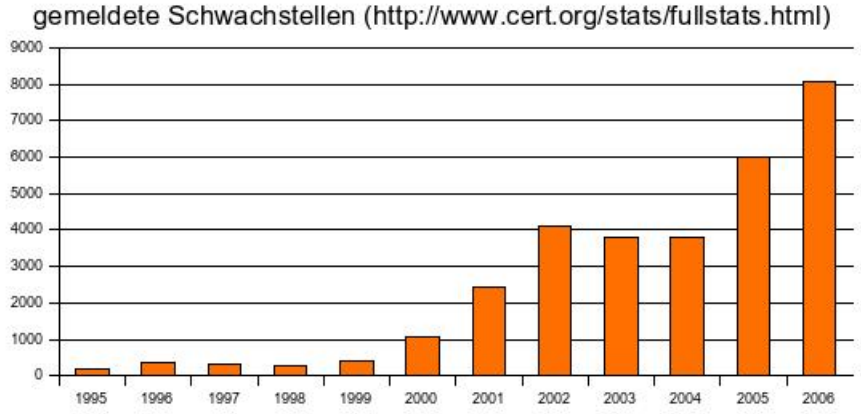

Abbildung 4.2: gemeldete Schwachstellen (CERT)

Stellt man dem die Einführung und Implementierung einer kommerziellen Patchmanagementlösung gegenüber, kann mit einem überschaubaren Aufwand das Risiko für die beispielhaft genannte Wurmattacke minimiert werden. Die Implementierung einer Patchmanagementlösung setzt sich aus den Initialkosten für die Software, die Implementierung und den jährlichen Wartungsvertrag zusammen. Für ein für Klein- und Mittelunternehmen optimiertes Patchmanagementsystem muß initital mit ca. 3.000,–€ gerechnet werden. Der Installationsaufwand bedeutet nocheinmal Kosten für einen externen Dienstleister von zumindest 2.000,–€, und jährlich werden ca. 20% des Initialkaufpreises der Software für Updatelizenzen anfallen. Dies bedeutet für das erste Jahr einen Gesamtaufwand von ca. 5.000,–€.

4.3 Kostenberechnung Softwareinventarisierung

Die Kosten von Softwareinventarisierungslösungen können abgeleitet von der Berechnung der Wiederherstellungskosten mit folgender Formel berechnet werden:

$$Kosten = IK + (T * R) + \sum_{i=1}^{n} (t * R)$$

IK .. initiale Kosten für die Software

n .. Anzahl der Computersysteme

T .. benötigte Zeit für die Installation der zentralen Serverkomponente und Integration in das bestehende Netzwerk

t .. benötigte Zeit für die Integration eines Computers in die zentrale Inventarisierungslösung

R .. Kosten pro Stunde

Im Folgenden sollen die Kosten einer kommerziellen, einer frei verfügbaren und einer auf einer wissenschaftlichen Arbeit basierender Computerinventarisierungslösung der im Zuge dieser Diplomarbeit entwickelten und hier vorgestellten Lösung *atc-inventory* gegenübergestellt werden. Die Kostenberechnung wird anhand der Lösungen Novell ZENworks Asset Management, OVAL und Netmap durchgeführt und ist in Tabelle 4.1 zusammengefasst. Für die Berechnung wird ein Stundensatz für einen EDV Techniker von 70,– € und eine Anzahl von 50 Computersystemen angenommen.

Produkt	Lizenzkosten	Installation Server-komponente	Integration eines Clients	Gesamtkosten
Novell ZENworks Asset Management	3.000,– €	16 Stunden	10 Minuten	4.703,33 €
OVAL	0 €	40 Stunden	15 Minuten	3.675,– €
NetMap	0 €	40 Stunden	0 Minuten	2.800,– €
atc-inventory in einem Netzwerk mit bestehenden Managementlösungen	0 €	4 Stunden	0 Minuten	280,– €
atc-inventory in einem Netzwerk ohne bestehende Managementlösungen	0 €	36 Stunden	0 Minuten	2.520,– €

Tabelle 4.1: Vergleich der Installationskosten für Softwareinventarisierungsapplikationen

Die Implementierung einer Enterpriselösung für die Inventarisierung von Computern mit dem Betriebssystem Microsoft Windows wird am Beispiel von Novell ZENworks Asset Management gezeigt. Für die Verwendung von Novell ZENworks Asset Management wird ein Server mit einem Windows Betriebssystem und einer MS SQL oder Oracle Datenbank vorausgesetzt. Da dies Standardvorgaben sind, und dementsprechend solche Systeme teilweise bereits bei Klein- und Mittelbetrieben im Einsatz sind, werden die Kosten für diese Hard- und Software nicht in die Berechnung miteinbezogen. Die Kosten für die Software, entnommen der Novell Webseite, inkl. 1 Jahr Wartung betragen jeweils

pro System 39,–€ für die Software ZENworks 7.5 Asset Management und 21,–€ für die Software Asset Inventory for ZENworks 7.5 Asset Management. Somit fallen für 50 Computer das erste Jahr Softwarekosten von 3.000,–€ an. Die Installation der Serversoftware und die Integration der Lösung in das bestehende Netzwerk beansprucht inklusive Tests in etwa 16 Stunden eines EDV Technikers, die Integration pro Client hängt von der ausgewählten Inventarisierungstechnologie und dem Vorhandensein eines zentralen Verzeichnisdienstes ab, es wird für die Intergration eines Clients ein Aufwand von 10 Minuten angenommen.

$$\text{Kosten Novell ZENworks} = 3000 + (16 * 70) + \sum_{i=1}^{50} (\frac{1}{6} * 70) = 4703,33$$

Dies bedeutet Initialkosten von 4.703,33 € für die Implementierung einer kommerziellen Softwareinventarisierung.
Implementiert man stattdessen OVAL als Inventarisierungssoftware, berechnen sich die Kosten folgendermaßen. Mitre stellt mit OVAL kein fertiges Produkt zur Verfügung, sondern vielmehr XML Schemas zum Repräsentieren von System- und Computerinformationen. Als Beispielimplementierung gibt es von Mitre den OVAL Interpreter, der alle Anforderungen für den Einsatz in einem produktiven Netzwerk mitbringt. Der OVAL Interpreter muß auf jedem Client installiert werden, dafür sind pro Client 15 Minuten einzuplanen. Der OVAL Interpreter sammelt zwar die Daten des lokalen Computers, es gibt seitens Mitre jedoch keinen Dienst, der die gesammelten Daten zentral zusammenfasst. Daher ist hier ein relativ großer Integrationsaufwand zu berechnen. Dieser ist mit einem Minimum von einer Mannwoche anzugeben, da abhängig vom zu inventarisierenden Netzwerk und der vorhandenen EDV Infrastruktur ein Dienst implementiert werden muß, der die Daten von den einzelnen Clients holt, und zentral zusammenfasst. Softwarekosten fallen keine an, da die OVAL Software und die OVAL XML Schemas frei sind. Daher muß man mit Kosten von 3.675,–€ für eine OVAL Installation rechnen.

$$\text{Kosten OVAL} = 0 + (40 * 70) + \sum_{i=1}^{50} (\frac{1}{4} * 70) = 3675$$

Bei der Implementierung von Netmap fallen ähnliche Kosten wie bei einer OVAL Installation an. Netmap integriert für jede Aufgabe eine eigene bereits bestehende Software. Bei einer Auswahl von agentenlosen Komponenten kann der Zeitaufwand der Integration der einzelnen Clients in die zentrale Lösung zwar auf ein Minimum verringert werden, die Installation und der Test der von Netmap benötigten Hilfsprogramme würde schätzungs-

weise ebenfalls einen Zeitaufwand von mindestens einer Mannwoche bedeuten.

$$\text{Kosten NetMap} = 0 + (40 * 70) + \sum_{i=1}^{50} (0 * 70) = 2800$$

Im Gegensatz zu den bisherigen Kalkulationsbeispielen bedeutet die Implementierung des in dieser Diplomarbeit vorgestellten Konzepts der Softwareinventarisierung für die meisten Unternehmen wesentlich weniger Intitialkosten. *atc-inventory* verwendet ähnlich wie Netmap bereits bestehende Applikationen für die Sammlung der relevanten Computerdaten. Im Gegensatz zu Netmap versucht *atc-inventory* aber bereits im Netzwerk bestehende Managementlösungen miteinzubeziehen, was zu einem wesentlich geringeren Integrationsaufwand führt. Daher gibt es zwei Berechnungsszenarien für die Verwendung von *atc-inventory*. Der erste Fall ist die Implementierung von *atc-inventory* in einem Netzwerk, welches bereits Managementlösungen einsetzt und dafür bereits *atc-inventory* Plugins bestehen. Das zweite Szenario ist die Integration in ein Netzwerk ohne Managementlösungen, was zu einem ähnlich hohen Integrationsaufwand wie NetMap oder OVAL führt.
Für Szenario 1 (ein Netzwerk mit bestehenden Managementlösungen und vorhandenen *atc-inventory* Plugins) können die Kosten für die Integration von *atc-inventory* und dem Test der Installation auf in etwa 280,–€ reduziert werden.

$$\text{Kosten atc-inventory Szenario 1} = 0 + (4 * 70) + \sum_{i=1}^{50} (0 * 70) = 280$$

Diese Kosten berechnen sich aus einem Aufwand von 4 Stunden für die Anpassung der Konfigurationsdateien und dem Test der Konfiguration. Gibt es keine bestehenden Managementlösungen und kann auf die Clients nicht zentral per standardisierter Schnittstellen wie z.B. WMI, RemoteRegistry oder SNMP zugegriffen werden, müssen von *atc-inventory* benötigte Applikationen zuerst installiert werden. Hier kann mit einem Aufwand von 4 bis 5 Manntage gerechnet werden, was sich bei den angenommenen Kosten von 70,–€ pro Stunde in Kosten von ca. 2.520,–€ niederschlägt.

$$\text{Kosten atc-inventory Szenario 2} = 0 + (36 * 70) + \sum_{i=1}^{50} (0 * 70) = 2520$$

Aufgrund der angeführten Kostenbeispiele für kommerzielle und frei verfügbare Softwareinventarisierungslösungen wird ersichtlich, daß *atc-inventory* für Klein- und Mittelbetriebe eine interessante Alternative darstellt. Neben dem Kostenfaktor ist *atc-inventory*

zudem auf Klein- und Mittelbetriebe zugeschnitten und erfüllt die oben genannten Erfordernisse an Softwarelösungen für den KMU Bereich.

4.4 Aufwand für laufende Überprüfungen mit einer Softwareinventarisierungslösung am Beispiel von *atc-inventory*

Aus den dargestellten Kostenberechnungen für die einzelnen Lösungen ist jedoch nicht ersichtlich, welchen Aufwand der Einsatz einer Inventarisierungslösung in der laufenden Betreuung bedeutet. Dieser wird am Beispiel von *atc-inventory* gezeigt.
Eine automatisierbare Softwareinventarisierung wird sehr oft mittels zentral ausgeführte Computerüberprüfungen durchgeführt. Für solche Überprüfungen werden in der Regel administrative Benutzerrechte auf den Clientsystemen benötigt. Die Überprüfung wird zentral gestartet und überprüft jedes Zielsystem auf die gewünschten Informationen. Dazu zählen bei *atc-inventory* zum Beispiel Netzwerkinformationen wie der Hostname, die IP Adresse, die MAC Adresse und die angebotenen Netzwerkdienste, und systembasierende Informationen wie das installierte Betriebssystem und die installierte Software. Die manuelle Durchführung eines Scans hängt von mehreren Faktoren ab, wie der Größe des zu überprüfenden Netzwerks, der Möglichkeit des administrativen Zugangs auf alle zu überprüfenden Computersysteme und dem Vorhandensein von Firewalls. Für die Überprüfung wird pro Computersystem bei einer Standardkonfiguration eine Zeitspanne von bis zu 10 Minuten angenommen.
Einen größeren Aufwand bedeutet jedoch die Auswertung und die Interpretation der Ergebnisse. Für die Berechnung dieses Aufwands dient wieder das fiktive Beispiel des Internetwurms. Ob die Systeme des Unternehmens durch den Wurm kompromitiert werden können, hängt von der Installation eines bestimmten Updates für das Betriebssystem ab. Der Aufwand für dieses fiktive Überprüfen eines installierten Updates ist in der Tabelle 4.2 zusammengefasst. Wiederum wird eine Anzahl von 50 zu überprüfende Computersysteme angenommen. Der oben beschriebene Zeitaufwand pro Überprüfung und pro Auswertung ist geschätzt und basiert auf langjährigen Erfahrungswerten.
Wird händisch ohne Hilfsprogramme für die Inventarisierung überprüft, ob ein spezieller Patch installiert ist, bedeutet dies neben dem Aufwand von ca. 10 Minuten pro Client für das Überprüfen der installierten Software zusätzlich den Aufwand für das händische Mitführen einer Liste der installierten Software pro Computer. Obwohl in der Praxis diese Listen computerunterstützt mit Applikationen wie zum Beispiel Microsoft Excel erstellt und aktualisiert werden, muß pro Client zusätzlich ein Aufwand von 10 Minuten berech-

Szenario	Inventarisierung von 50 Systemen	Auswertung Stand einer installierten Software	Gesamtaufwand
manuell ohne Inventarisierungs-programm	1000 Minuten	50 Minuten	17,5 Stunden
manuell mit *atc-inventory*	10 Minuten	50 Minuten	1 Stunde
Managementlösung mit *atc-inventory*	0 Minuten	5 Minuten	5 Minuten
ATCERT mit *atc-inventory*	0 Minuten	5 Minuten	5 Minuten

Tabelle 4.2: Vergleich des Aufwands für die Inventarisierung und Abfrage einer installierten Software

net werden. Das bedeutet pro Client einen Aufwand von 20 Minuten. Für die Auswertung muß nocheinmal 1 Minute pro Client berechnet werden.
Bei der manuellen Überprüfung anhand einer Inventarisierungslösung, wie der im Zuge dieser Diplomarbeit entwickelten *atc-inventory* Applikation, fällt kein Überprüfungsaufwand pro Computersystem an, da dies das Programm macht. Daher kann man für die Intialisierung der Überprüfung ca. 10 Minuten rechnen. Die händische Auswertung des Ergebnisses bzgl. des installierten Patches bedeutet jedoch pro Client wiederum einen Aufwand von 1 Minute.
In der Akademie der bildenden Künste soll das System automatisiert eingesetzt werden. *atc-inventory* soll automatisch einmal pro Stunde gestartet werden und die Ergebnisse der überprüften Netzbereiche in eine zentrale XML Datei schreiben. Diese XML Datei kann dann einmal pro Tag in die bestehende Oracle Datenbank eingelesen werden. Dementsprechend ist das Überprüfen eines Patches auf allen inventarisierten Hosts eine Abfrage in der Datenbank. Das bedeutet einen Zeitaufwand von maximal 5 Minuten.
Beim Projekt ATCERT wird die Inventarisierung ebenfalls automatisiert durchgeführt. Das System stellt die Möglichkeit bereit, automatisierbare Inventarisierungen mit diversen Sensoren durchzuführen. Zu diesen Sensoren zählt auch der Prototyp von *atc-inventory*. Nach der Implementierung der automatischen Inventarisierung bedeutet das Überprüfen eines vorhandenen Patches für die Risikoanalyse eine Abfrage im ATCERT System. Dieser Aufwand ist ähnlich hoch wie der Aufwand bei der Datenbanklösung der Akademie der bildenden Künste Wien.

5 Konzeption und Entwurf

Dieses Kapitel beschreibt neben der Nachteile der sich am Markt befindlichen Produkte die verschiedenen Netzwerk- und Sicherheitslösungen an der Akademie der bildenden Künste Wien. Auf dieser Arbeitssituation basiert der Arbeitsansatz, die von verschiedenen Programmen gesammelten Daten zu bündeln und in einer Lösung zusammenzufassen. Es wird der Lösungsansatz und das Design des zu entwickelnden Prototypen *atc-inventory* beschrieben, ein Datenmodell entworfen und die Schnittstellen für die integrierten third-party Applikationen definiert.

5.1 Ausgangssituation

5.1.1 Nachteile der sich am Markt befindlichen Produkte

Durch die Relevanz von IT Asset Management im IT Sicherheitsmanagement und in best practices wie ITIL gibt es eine große Anzahl von bereits implementierten Lösungen. Diese Lösungen implementieren nützliche Funktionen, decken aber meistens nicht die komplett benötigte Funktionalität ab. Viele der sich am Markt befindlichen Softwarelösungen weisen mindestens einen der folgenden gravierenden Nachteile auf [VVZK02]:

1. *begrenzter Anwendungsbereich*: die meisten Applikationen implementieren eine Lösung für ein genau definiertes Gebiet. Nmap z.B. implementiert einen Scanner für die Analyse offener Netzwerkports eines Systems und die Möglichkeit, anhand der TCP Antwortpakete das Betriebssystem zu bestimmen [3]. Die Möglichkeit von Bannergrabbing für die Bestimmung der aktiven Netzwerkdienste und der installierten Versionen bringt es aber nur eingeschränkt mit. Sehr viele Applikationen analysieren nur ein spezielles Gebiet, erfassen aber das untersuchte System nicht gesamtheitlich.

2. *applikationsabhängiges Datenmodell*: die verschiedenen Applikationen speichern ihre gesammelten und ausgewerteten Daten in unterschiedlichen Modellen. Einige schreiben sie in eine Datenbank, andere geben unstrukturierte Textdateien aus, und wiederum andere verwenden applikationsspezifische XML Schemas. Die den Applikationen zugrundeliegenden Datenmodelle decken jedoch in der Regel nur die

Informationen der Applikation ab und sind meist nicht erweiterbar. Dies erschwert die gemeinsame Verwendung der Ergebnisse verschiedener Programme.

3. *nicht erweiterbar*: in den meisten Fällen ist es sehr schwer bzw. sogar unmöglich, die Funktionalität der einzelnen Applikationen zu erweitern. Ein Beispiel hierfür sind SNMP-basierte Applikationen. SNMP ist immer noch der de-facto Standard für Netzwerkgeräte wie Switches und Router. Bei Serversystemen ist SNMP aus Sicherheitsgründen sehr oft deaktiviert. Auch wenn die gewünschten Informationen über andere Mechanismen wie z.B. durch die Ausführung eines Skripts zugänglich wären, können die existierenden Programme nicht auf diese Möglichkeiten zurückgreifen.

4. *keine automatisierbare Auswahl geeigneter Werkzeuge*: bei Systemanalyse- oder Systemüberwachungsaufgaben gibt es keine automatisierbare Unterstützung, welche der verfügbaren Applikationen für die Lösung einer Aufgabe in welcher Weise eingesetzt werden soll.

Diese Liste beinhaltet die Anforderungen an eine effiziente IT Asset Management Applikation. Da ein Großteil der Programme im Bereich IT Sicherheitsmanagement und IT Asset Management diesen Anforderungen nicht entspricht, setzen Unternehmen in der Regel verschiedene Produkte für die verschiedenen Aufgaben im Bereich der Netzwerk- und Computersicherheit ein. Damit werden Insellösungen implementiert, für jeden Bereich wird eine dafür spezialisierte Applikation verwendet, die Daten der parallel implementierten Lösungen werden nicht berücksichtigt.
Zum Beispiel setzt die A. B. Freeman School of Business der Tulane University als Patchmanagementsystem HFNetChkPro der Shavlik Corporation ein. Die Verwendung eines Microsoft Windows Update Server und die in das Microsoft Windows Betriebssystem integrierten Updatemechanismen entsprachen nicht den Anforderungen der Schule, daher wählte die EDV Abteilung die Applikation von Shavlik als Patchmanagementsystem aus [GM04]. Obwohl diese Applikation eine Hard- und Softwareinventarisierung durchführt und Informationen über die Konfiguration der in das Managementsystem integrierten Computer sammelt, ist die Funktionalität auf das Patchmanagement reduziert.

5.1.2 Ausgangssituation an der Akademie der bildenden Künste Wien

Ein ähnlicher Fall ist das Netzwerk der Akademie der bildenden Künste Wien. In der Tabelle 5.1 sind die derzeit installierten Computer- und Netzwerkmanagementlösungen aufgelistet.
Jede einzelne sich im Einsatz befindliche Lösung deckt einen speziellen Bereich der IT

Produkt	**Beschreibung**	**Betriebssystem**	**Speichermedium**
Windows Server Update Services	Patchmanagement	Microsoft Windows	Datenbank
Red Hat Network	Patchmanagement, Lizenzmanagement	Red Hat Linux	Datenbank
Sun Update Manager	Patchmanagement	Solaris	Datenbank
Apple Remote Management	Patchmanagement, Remoteadministration, Hard- und Software-inventarisierung Softwareverteilung	Mac OS X	Datenbank
OCS Inventory NG	Hard- und Software-inventarisierung, Softwareverteilung	Microsoft Windows, Mac OS X, Linux, Solaris	Datenbank
Nmap	Netzwerkport-analyse	Microsoft Windows, Mac OS X, Linux, Solaris	Textdatei, XML
Nessus	Schwachstellen-analyse	Microsoft Windows, Mac OS X, Linux, Solaris	Textdatei, XML
Nagios	Hostüberwachung	Microsoft Windows, Mac OS X, Linux Solaris, SNMP	Textdatei, Datenbank
Cacti	Hostüberwachung, Perfomancegraphen	Microsoft Windows, Mac OS X, Linux, Solaris, SNMP	Datenbank
Snort	Intrusion Detection	gesamter IP Netzwerkverkehr	Textdatei, Datenbank
NTOP	Netzwerkverkehr-überwachung, Performancegraphen Netzwerkflüsse	gesamter IP Netzwerkverkehr	Textdatei
Ironview	Netzwerk-management	SNMP	Datenbank

Tabelle 5.1: eingesetzte Computer- und Netzwerkmanagementlösungen in der Akademie der bildenden Künste Wien

Infrastruktur der Akademie der bildenden Künste Wien ab. Obwohl die Anforderungen z.B. im Bereich Patchmanagement bei den vier sich im Einsatz befindlichen Betriebssystemen Microsoft Windows, Mac OS X, Red Hat Enterprise Linux und Solaris ähnlich sind, werden vier auf die einzelnen Betriebssysteme abgestimmte Lösungen verwendet.
Für Microsoft Windows Clients und Server kommt die Software Microsoft Windows Server Update Services 2.0 (WSUS) zum Einsatz. Diese Updatemanagementlösung ermöglicht Adminstratoren die Distribution und Installation der aktuellen Microsoft Updates auf allen im WSUS registrierten Computern und Servern. Unterstützt werden die Clientbetriebssysteme Microsoft Windows 2000, Microsoft Windows XP, Microsoft Windows Vista und die Serverbetriebssysteme Microsoft Windows Server 2000 und Microsoft Windows Server 2003. Updates für die Applikationen Microsoft Exchange Server, Microsoft SQL Server und Microsoft Office können ebenfalls automatisiert verteilt werden. Der Administrator hat die Möglichkeit, alle von Microsoft über das Microsoft Update veröffentlichten Updates und Service Packs zu installieren. Installiert werden nur jene Updates und Service Packs, die vom Administrator für die jeweiligen Computer freigegeben sind. Das gibt dem Administrator die Möglicherkeit, Server bzw. Computer mit Spezialsoftware nicht sofort in einen Updatezyklus mitaufzunehmen, sondern diese Geräte zeitversetzt und z.B. nach Durchführung zusätzlicher Tests zu aktualisieren. Alle Update- und Managementaufgaben können über eine Webapplikation durchgeführt werden (Abbildung 5.1). Ein Nachteil dieses webbasierenden Programms besteht darin, daß zwingend ein Microsoft Internet Explorer vorausgesetzt wird, mit alternativen Browsern funktioniert die Applikation nicht.
Das Produkt stellt rudimentäre Reportingfunktionen für die Administratoren bereit. Dazu zählen z.B. eine Liste aller auf einem Rechner installierten Patches oder eine Liste von Clients, die freigegebene Updates noch nicht installiert haben. Diese Auswertungen sind jedoch nicht ausreichend und benutzerdefinierte Abfragen wie z.B. den Status aller PCs einer Abteilung können in der Weboberfläche nicht durchgeführt werden. Hierfür stellt Microsoft eine eigene API bereit, über die selbst definierte Abfragen programmiert werden können. Direkt auf die Microsoft SQL Datenbank zuzugreifen ist nicht vorgesehen. Der Zugriff auf die Datenbank ist auch nicht empfehlenswert, da weder das Datenbankschema noch die Zugriffsrechte auf die Datenbank von Microsoft öffentlich dokumentiert sind.
Das Updatemanagement für die Red Hat Linux Server kann auf zwei Arten durchgeführt werden. Der Einsatz von Red Hat Enterprise Linux ist lizenzpflichtig, daher müssen alle Red Hat Enterprise Linux Server bei Red Hat registriert werden. Für diese lizenzierten Server bietet Red Hat einen Updatedienst, der über eine Weboberfläche zugänglich ist. Über das Webfrontend können alle registrierten Systeme überwacht werden, die

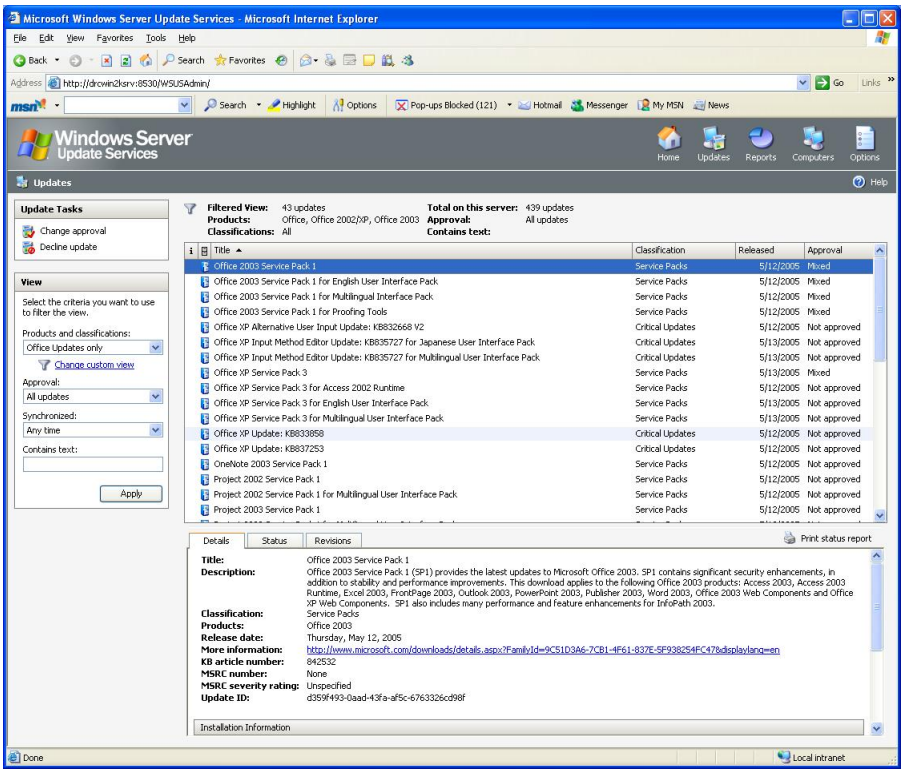

Abbildung 5.1: WSUS Webansicht

Webapplikation inventarisiert die Hard- und Software, sichert betriebssystemnahe Konfigurationsdaten wie die Konfiguration der Netzwerkkarten und die Routingtabellen, und stellt einen Updatemechanismus zur Verfügung (Abbildung 5.2). Dieses System hat aber einige gravierende Mängel im Bereich Interoperabilität. Das größte Problem liegt darin, daß der Dienst bei Red Hat gehostet wird, und man nur über die Weboberfläche auf die gesammelten Informationen zugreifen kann. Nebenbei stellt Red Hat den Dienst nur für registrierte Red Hat Enterprise Linux Systeme zur Verfügung. Möchte man bei den Red Hat Systemen nicht auf die Weboberfläche zurückgreifen, können über die Kommandozeile auch verfügbare Updates überprüft und installiert werden.

Der Grund für den Einsatz einer Enterprise Linux Distribution liegt darin, daß getestete Softwareapplikationen und eine Garantie für die längerfristige Verfügbarkeit von Updates sehr wichtig für den problemlosen Betrieb des Universitätsnetzwerks sind [15] [DOFC00]. Wo kein kommerzieller Support für die installierten Applikationen und Serverdienste benötigt wird, kommt statt Red Hat Enterprise Linux die Linuxdistribution von CentOS zum Einsatz. CentOS kompiliert die Quellpakete der Red Hat Enterprise Distributionen neu, und stellt diese lizenzkostenfrei zur Verfügung. Das bedeutet, der Benutzer hat ein auf Unternehmen optimiertes Linuxsystem wie Red Hat Enterprise Linux, muß jedoch keine Lizenzkosten tragen. Diese CentOS Systeme können nicht über den Update-

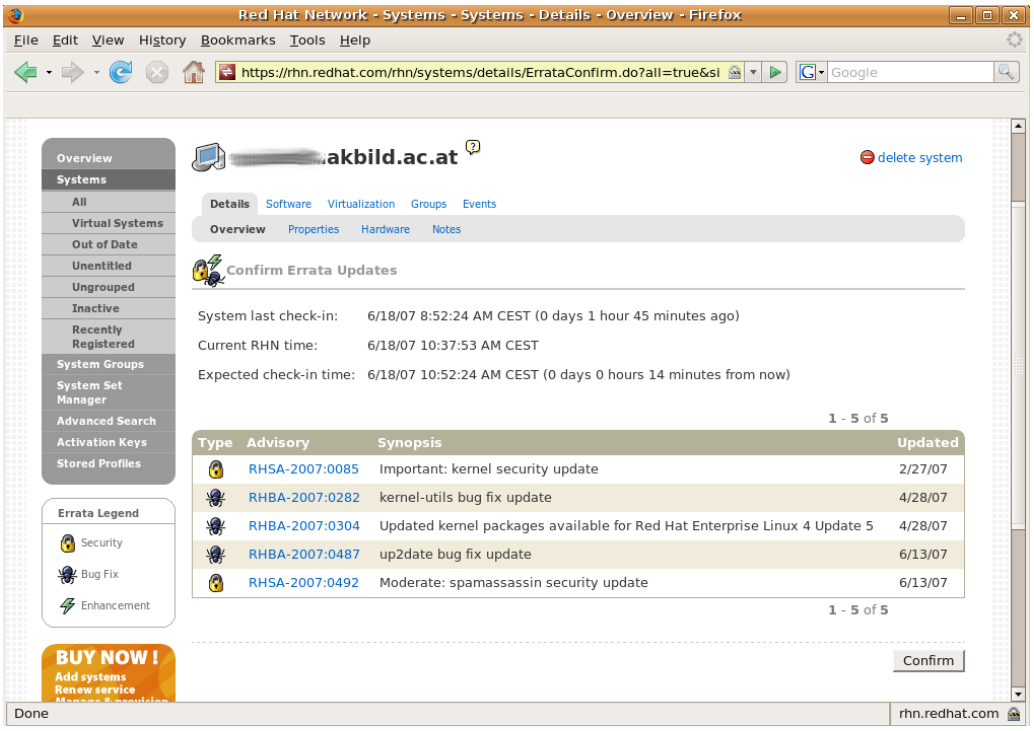

Abbildung 5.2: Red Hat Network Updatemanagement

dienst von Red Hat verwaltet werden, die Administration und Aktualisierung erfolgt über das lokale System.

In der Akademie der bildenden Künste werden derzeit Automatismen in Form von selbst erstellten Programmen und Skripts für die Updateverwaltung der Linuxsysteme eingesetzt. Damit diese Automatismen alle Linuxsysteme abdecken, wird im täglichen Betrieb nicht auf die Weboberfläche von Red Hat Network zurückgegriffen. Die entwickelten Skripts fragen über die Kommandozeile sowohl bei Red Hat als auch bei CentOS die aktuell installierte Software inkl. Patchlevel ab und installieren bei Bedarf zusätzliche Software und Updates. Die Kommandozeilenabfragen überprüfen den Status der installierten Software über die RPM Datenbank, daher werden nur Applikationen, die über den Paketmanager installiert wurden, berücksichtigt.

Ähnlich ist es bei den Sun Solaris Servern. Sun bietet wie Red Hat einen webbasierenden Update- und Inventarisierungsdienst an. Dieser Dienst wird ähnlich wie der Red Hat Network Service nicht regelmäßig verwendet. Für die Administration der Solaris Server wird auf die lokalen Kommandozeilenprogramme zurückgegriffen. Wie bei den Linuxsystemen kann damit die installierte Software überprüft und aktualisiert bzw. neue Software installiert werden (Abbildung 5.3). Auch hier wird nur die vom Paketmanager installierte Software berücksichtigt

```
-bash-3.00# smpatch analyze
119252-17 SunOS 5.10: System Administration Applications Patch
124630-06
121430-14 SunOS 5.8 5.9 5.10: Live Upgrade Patch
121308-09 SunOS 5.10: Solaris Management Console Patch
119254-38 SunOS 5.10: Install and Patch Utilities Patch
```

Abbildung 5.3: Überprüfen von verfügbaren Updates unter Solaris 10

Als vierte Update- und Patchmanagementsoftware kommt für Apple Mac OS X Clients die Apple Software Remote Desktop zum Einsatz. Diese Applikation bringt neben dem Patchmanagement sowohl eine Remotemanagementfunktionalität als auch die Möglichkeit von Softwareverteilung und Inventarisierung der eingesetzten Hard- und Software mit. Die gesammelten Daten werden in eine Postgresql Datenbank geschrieben, eine weitere Verwendung der Daten ist aufgrund des Datenbankschemas nicht vorgesehen.
Als Ticket Tracking System kommt eine in PHP selbst entwickelte Datenbankapplikation zum Einsatz. Als Datenbank wird Mysql verwendet. Diese Datenbank enthält unter anderem alle vom Zentralen Informatikdienst der Universität betreuten Computer. Die Hard- und Softwareinventarisierung wird nicht automatisiert durchgeführt, bei Änderungen der Hardware oder der Softwareinstallationen werden diese Daten vom durchführenden Techniker in die Datenbank eingepflegt. Dies führt zu einer nicht aktuellen Datenbank, da dieses händische Pflegen der Daten fehleranfällig ist.
Wie man an der Anzahl der eingesetzten Computer- und Netzwerkmanagementlösungen sieht, gibt es zwar alle für ein gesamtheitliches Sicherheitsmanagment benötigten Informationen, nur sind diese statt an einer zentralen Stelle in verschiedenen Formaten und Datenbanken verfügbar. Bei der Analyse der einzelnen verwendeten Managementsoftwarelösungen fällt auf, daß alle Lösungen ähnliche Informationen sammeln und daß damit teilweise dieselben Daten von mehreren Applikationen gespeichert werden. Die einzelnen Systeme bei Bedarf getrennt zu verwenden, ist zwar zeitaufwendig und nicht effizient, aber die gemeinsamen Daten werden für die einzelnen Insellösungen nicht unbedingt benötigt.
Mit Wintersemester 2006/2007 wurde an der Akademie der bildenden Künste Wien ein neues Informationsmanagementsystem eingeführt. Dieses auf einer Oracle Datenbank basierende System beinhaltet alle relevanten Daten von Forschung und Lehre. Zudem ist es auch das datenführende System für die Benutzerdaten der Universitätsangehörigen (Bedienstete und Studierende). Mit der Implementierung dieses Informationssystems liegen ein Teil der vom Ticket Tracking System benötigten Daten wie z.B. die Räume der Universität und die Benutzer jetzt in der Oracle Datenbank. Daher wurde begonnen, das Troubleticket System von der PHP/Mysql Lösung auf die Oracle Datenbank zu migrieren. In

diesem Zuge wurde auch eingeplant, die Hard- und Softwareinventarisierung zu automatisieren und diese Daten ebenfalls in der Oracle Datenbank abzulegen. Da die eingesetzten Managementlösungen proprietär sind und eigene Datenbanken mitbringen, gibt es keine Möglichkeit, diese Daten direkt in der Oracle Datenbank zu speichern.
Ein weiteres System, das für einen effizienten Einsatz in einem Universitätsnetzwerk einen zentralen Datenstand voraussetzt, ist das sich im Einsatz befindliche Intrusion Detection System Snort. Snort generiert wie ein Großteil der sich am Markt befindlichen Intrusion Detection Lösungen eine große Anzahl von *false positives* [KV02]. Das bedeutet, das Intrusion Detection System meldet für jeden entdeckten Angriff bzw. für jede verdächtige Netzwerkkommunikation einen Alarm. Ob der Angriff erfolgreich ist, oder ob das angegriffene Ziel überhaupt die dem Angriff zugrundeliegende Sicherheitslücke aufweist, kann das Intrusion Detection System nicht bestimmen. Die Folge ist, daß reelle und gezielt ausgeführte Angriffe in der großen Zahl der gemeldeten Angriffe untergehen. Mit aufwendiger Konfiguration von Snort und regelmäßigen Anpassungen der Konfiguration können die *false positives* zwar signifikant reduziert werden, die Anzahl der Alarme übersteigt jedoch immer noch jenen Punkt, der als sinnvoll administrierbar gilt, und anhand dem man für gezielte Angriffe Gegenmaßnahmen ergreifen kann.
Für dieses Problem gibt es verschiedene Lösungsansätze. Die meisten Lösungen versuchen irrelevante Alarme zu unterdrücken, indem die generierten Alarme mit anderen Systeminformationen wie Logdateien, Firewallalarme oder Netzwerkflüsse korrelliert werden. Eine weitere Möglichkeit, nicht relevante Alarme zu unterdrücken, ist die Installation des snort patch für *alert verification* [KRV04]. Sourcefire, der Hersteller der Open Source Software Snort, implementiert für ihre lizenzpflichtige Produktlinie eine weitere Lösung. Die RNA (Real Network Awareness) Lösung von Sourcefire liest den Verkehr nicht nur für die Analyse durch die Intrusion Detection Engine mit ihrer Angriffsmusterdatenbank mit [14]. Anhand passiver Tests werden vorhandene aktive Komponenten im Netzwerkverkehr gesucht. Mittels Banner Grabbing und der Analyse verschiedener Parameter in den IP, TCP und UDP Headern wie z.B. das *Time To Live (TTL) field*, das *Don't Fragment (DF) Bit* oder das *Type of Service (TOS) field*, werden das Betriebssystem, die installierten Netzwerkdienste und deren Versionsnummern erkannt. Unterstützt werden die passiven Tests zusätzlich durch aktive Tests der sich im Netzwerk befindlichen Netzwerkknoten. Damit inventarisiert Sourcefires RNA Lösung die gesamte IT Infrastruktur. Mit dieser Methode können die gemeldeten Alarme reduziert werden, da Alarme nur generiert werden, falls das Angriffsziel durch der dem Angriff zugrundeliegenden Schwachstelle kompromitierbar ist. Das bedeutet, mit einem zentralen Datenbestand der eingesetzten Hardware und der darauf installierten Software kann Snort effizient konfiguriert werden.

5.1.3 Beschreibung des Arbeitsansatzes

Die Idee von *atc-inventory* liegt darin, die Daten der einzelnen bereits verfügbaren Managementlösungen zusammenzuführen und zu konsolidieren, und bei Bedarf durch weitere Applikationen zu erweitern. Hierfür müssen Schnittstellen für die Integration bestehender und neu entwickelter 3rd-party Produkte spezifiziert werden. Nach der Analyse der sich in der Akademie der bildenden Künste Wien im Einsatz befindlichen Produkte, der Ontologie von ATCERT und diversen wissenschaftlichen Arbeiten über Asset- und Patchmanagement werden folgende Schlüsselpunkte in das Design von *atc-inventory* einfließen:

- *atc-inventory* soll anhand genau definierter Schnittstellen mit spezifizierten Ein- und Ausgabeparametern third-party Produkte in die Funktionalität miteinbinden. Damit soll die Applikation sehr flexibel bleiben und die Bereiche aller integrierten Applikationen abdecken.

- *atc-inventory* soll auf ein erweiterbares XML Schema basieren. Für die Definition des XML Schemas werden die Datenmodelle verschiedener Applikationen und die dem Projekt ATCERT zugrundeliegende Datenontologie analysiert. Das XML Schema soll problemlos um weitere Merkmale erweitert werden können, da integrierte Applikation jeweils nur den für sie relevanten Teil des XML Schemas aktualisieren.

- Durch die Integration von third-party Produkten soll die Funktionalität von *atc-inventory* problemlos erweitertbar sein

- Die Schnittstellendefinition für den Einsatz von third-party Produkten in *atc-inventory* soll spezifizieren, welche Systeme und welche Informationen von den einzelnen Applikationen abgedeckt werden. Damit kann *atc-inventory* entscheiden, welche Applikation für welchen Zweck verwendet wird. Zudem soll in der Schnittstellendefinition eine Priorisierung spezifiziert werden. Anhand dieser Priorisierung kann *atc-inventory* entscheiden, welche Applikation bei Überschneidungen die größte Gewichtung besitzt.

- *atc-inventory* soll auf einer freien Programmiersprache basieren und auf allen von der Sprache unterstützten Betriebssystemen laufen. Die integrierten Applikationen können jedoch betriebssystem- oder technologieabhängig sein, das wird in der Schnittstellendefinition berücksichtigt.

5.2 Entwurf des Prototyps

5.2.1 Definition des Datenmodells

Für die Definition des Datenmodells von *atc-inventory* wurden die Ontologie von AT-CERT, verschiedene Applikationen wie Windows Server Update Service, Microsoft Baseline Security Analyzer, Nmap, OCS Inventory NG, Nessus, Apple Remote Desktop und NetMap sowie das XML Schema von OVAL und SVIM analysiert. *atc-inventory* speichert die Daten in einem XML Schema, und nicht in einer relationalen Datenbank. Dafür gibt es mehrere Gründe [Bed99, Lea99]

1. XML steht für strukturierte Daten
2. XML ist selbstbeschreibend
3. XML ist ein verbindlich dokumentierter Industriestandard und plattformunabhängig
4. XML kann automatisiert und maschinell weiterverarbeitet und auf Fehler überprüft werden
5. mit XML können Daten automatisiert ausgetauscht werden
6. in XML können jederzeit Attribute und Elemente hinzugefügt werden
7. für den Zugriff auf XML stehen standardisierte XML API's zur Verfügung
8. es gibt viele freie XML Parser am Markt
9. XML unterstützt Internationalisierung und Lokalisierung

XML hat aber auch einige Nachteile. So ist XML langsam und speicherhungrig. Im Gegensatz zu relationalen Datenbanken werden zudem keine Lockingmechanismen und Transaktionen unterstützt. XML sollte jedoch nicht mit relationalen Datenbanken verglichen werden, da XML und Datenbanken keine Konkurrenztechnologien darstellen, sondern sich ergänzen [SR01]. Die Datenbanksysteme der großen Hersteller wie Oracle, Microsoft, IBM und Mysql können XML in- und exportieren. Damit können die von *atc-inventory* generierten Daten problemlos von anderen Applikationen weiterverarbeitet werden. Das XML Schema beinhaltet verschiedene system- und konfigurationsspezifische Attribute, die den zu inventarisierenden Host, die Netzwerkkonfiguration, das laufende Betriebssystem und die installierte Software abbilden. Derzeit sind keine hardwareabhängigen Elemente und Attribute im Schema vorgesehen, bei Bedarf können zusätzliche Elemente für die Abbildung der Hardware eingeführt werden. Folgende Elemente sind im

XML Schema für jedes inventarisierte System definiert:
Im Element `<lastscan>` steht das Datum und die Uhrzeit der letzten Überprüfung. Das Element `<status>` gibt an, ob der Host zu diesem Zeitpunkt aktiv war. Falls der Host nicht aktiv war, werden für die Neuaufnahme oder Aktualisierung der Daten nur hostbasierende Applikationen aufgerufen.

```
<lastscan>Datum Zeit</lastscan>
<status>Computerstatus beim letzten Durchlauf</status>
```

Im Element `<ipaddress>` wird die IP Adresse des Computers gespeichert, im Element `<macaddress>` steht die Hardwareadresse der an die IP Adresse gebundenen Netzwerkkarte.

```
<addresses>
  <ipaddress>a.b.c.d</ipaddress>
  <macaddress>00:FF:FF:FF:FF:FF</macaddress>
</addresses>
```

Im Element `<hostname>` steht der *full-qualified-domain-name*, der komplette Computername. Dieser setzt sich aus dem Computernamen und der dazugehörigen DNS Domäne zusammen. Da der Computername nicht immer eindeutig bestimmt werden kann, können mehrere Sonderfälle auftreten. Falls kein funktionierender DNS Server zur Verfügung steht oder der überprüfte Computer keinen DNS Eintrag besitzt, wird nur der Computername ohne DNS Suffix in das XML Element geschrieben. Ein weiterer Sonderfall besteht darin, daß manche Applikationen den Computernamen nicht auslesen können. Kann der Computername nicht bestimmt werden, wird ein generischer Name, der sich aus dem Wort `HOST-` und der Hardwareadresse der Netzwerkkarte zusammensetzt, verwendet. Kann die Hardwareadresse ebenfalls nicht bestimmt werden, dann setzt sich der Hostname aus `HOST-` und der IP Adresse zusammen.

```
<hostnames>
  <hostname>Computername</hostname>
  <dnstype>Art des DNS Eintrags des Computername</dnstype>
</hostnames>
```

Bei den Netzwerkports werden nur die offenen und die gefilterten Ports angegeben. Falls die Netzwerkports mit einem Programm wie Nmap überprüft werden, können die Informationen über die geschlossenen Ports möglicherweise durch den Einsatz einer Firewall nicht den Status des überprüften Computers widerspiegeln, sondern nur eine Betrachtung aus einem anderen Netz darstellen. Daher werden die geschlossenen Ports nicht in das XML Schema geschrieben. Das Element `<status>` definiert, ob das Port offen oder gefiltert ist, das Element `<protocol>` beinhaltet das Protokoll des Ports (TCP, UDP) und die `<id>` gibt die Portnummer aus.

```
<ports>
  <port>
    <status>Status des Netzwerkports</status>
    <protocol>Protokoll des Ports</protocol>
    <id>Portnummer</id>
  </port>
</ports>
```

Beim installierten Betriebssystem wird der Hersteller, die Betriebssystemgeneration (z.B. Microsoft Windows XP Professional) und das installierte Service Pack gespeichert.

```
<os>
  <osvendor>Hersteller</osvendor>
  <osgen>Betriebssystemversion</osgen>
  <servicepack>installiertes Service Pack</servicepack>
</os>
```

Die Softwareliste listet alle auf dem Computersystem installierten Applikationen, Patches und Service Packs auf. Ob Patches und Service Packs aufgelistet werden, hängt von der Installation dieser Updates ab. Bei Microsoft sind Patches und Service Packs eigene Installationen, daher scheinen sie auf. Wird hingegen ein Patch in die Applikation integriert, und die komplette Applikation wird neu installiert, scheint der Patch nicht als eigene Software auf. Für jede installierte Software wird der Hersteller, der Produktname, die Version und die Patchnummer in die XML Datei geschrieben.

```
<softwarelist>
  <software>
    <vendor>Hersteller</vendor>
    <product>Softwareprodukt</product>
    <version>Softwareversion</version>
    <patchlevel>Patchnummer</patchlevel>
  </software>
</softwarelist>
```

Als installierte Software gelten auch Updates für das Betriebssystem, die als eigene Softwarepakete installiert werden. Das bedeutet, daß im Element `<product>` z.B. Sicherheitsupdate für Windows XP steht. Dies wird insbesondere für Risikoanalysen bei Microsoft Windows Umgebungen benötigt. Microsoft veröffentlicht Updates für die verschiedenen Betriebssystemgenerationen als eigene Installationen. So kann z.B. der in der Microsoft Sicherheitswarnung MS03-027 [6] beschriebene Fehler in der Windows Shell durch die Installation des Patches KB821557 behoben werden. Für eine Risikoabschätzung der Gefährdung eines Unternehmens aufgrund dieses Fehlers muß analysiert werden, ob das Update auf allen Microsoft Windows XP Computern des Unternehmens installiert ist. Daher

werden diese Informationen in der *atc-inventory* XML Datei benötigt.
Im folgendem Listing ist ein Beispiel vom *atc-inventory* XML Schema dargestellt. Untersucht wurde ein PC mit Windows XP. Die offenen Netzwerkports und die installierten Applikationen sind gekürzt.

```
<?xml version ="1.0" ?>
<host>
  <lastscan>Mon May  7 15:33:54 2007</lastscan>
  <state>up</state>
  <addresses>
    <ipaddress>10.3.3.33</ipaddress>
    <macaddress>00:13:21:04:21:93</macaddress>
  </addresses>
  <hostnames>
    <hostname>pc-352.akademie.bildendekunst.ac.at</hostname>
    <dnstype>PTR</dnstype>
  </hostnames>
  <os>
    <osvendor>Microsoft</osvendor>
    <osgen>Microsoft Windows XP Professional</osgen>
    <servicepack>Service Pack 2</servicepack>
  </os>
  <ports>
    <port>
      <status>open</status>
      <protocol>tcp</protocol>
      <id>80</id>
    </port>
    <port>
      <status>open|filtered</status>
      <protocol>udp</protocol>
      <id>123</id>
    </port>
  </ports>
  <softwarelist>
    <software>
      <vendor>Microsoft Corporation</vendor>
      <product>Windows Installer 3.1 (KB893803)</product>
      <version>3.1</version>
      <patchlevel>KB893803</patchlevel>
    </software>
    <software>
      <vendor>Microsoft Corporation</vendor>
      <product>Sicherheitsupdate für Windows XP (KB896358)</product>
      <version>1</version>
      <patchlevel>KB896358</patchlevel>
    </software>
    <software>
      <vendor>Adobe Systems</vendor>
      <product>
        Adobe Acrobat 7.0 Professional - English, Français, Deutsch
      </product>
      <version>7.0.8</version>
```

```
49        <patchlevel/>
50      </software>
51      <software>
52        <vendor>Microsoft Corporation</vendor>
53        <product>Microsoft Office Professional Edition 2003</product>
54        <version>11.0.7969.0</version>
55        <patchlevel/>
56      </software>
57    </softwarelist>
58  </host>
```

Listing 5.1: Beispiel *atc-inventory* XML Schema

5.2.2 Definition des Programmablaufs

Das Program *atc-inventory* ist modular aufgebaut und setzt sich aus zwei Teilen zusammen, dem *atc-net-inventory* und dem *atc-host-inventory* (Abbildung 5.4). Der *atc-net-*

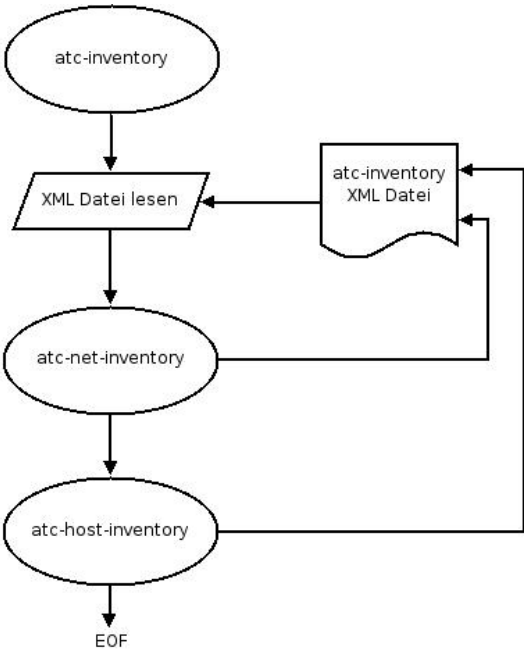

Abbildung 5.4: Design *atc-inventory*

inventory-Teil ist für die Inventarisierung der aktiven Netzwerkkomponenten zuständig, der *atc-host-inventory*-Teil inventarisiert die Software auf den untersuchten Computersystemen.
Die Ermittlung der aktiven Netzwerkknoten wird anhand von verfügbaren third-party Lösungen, neu entwickelten Applikationen oder einer Kombination aus mehreren Programmen durchgeführt. Durch die Einbindung von externen Applikationen als Plugins bleibt

die Applikation für zukünftige Anforderungen offen. Ein Vorteil der Verwendung bereits bestehender Produkte ist der geringe Entwicklungsaufwand. Zudem kann durch dieses Design jeder Teilbereich von einer dafür spezialisierten Software abgedeckt werden. Die *atc-net-inventory* Plugins sind standardmäßig agentenlose Programme, deren Aufgabe das Finden der aktiven Netzwerkknoten und -komponenten ist. Inventarisiert wird hier nur die IP Adresse, die Hardwareadresse der Netzwerkkarte, der Computername und wenn die Applikation ein Überprüfen der Netzwerkports unterstützt, die offenen und gefilterten Ports. Als Plugins werden hier typischerweise Programme wie Nmap, ping, arpping, nslookup und snmpwalk verwendet. Wichtig ist, daß bei integrierten Applikationen, die einen größeren Funktionalitätsumfang haben, über Parameter die benötigte Funktionalität genau definiert werden kann. Dadurch kann eine Applikation möglicherweise für verschiedene Operationen eingesetzt werden. Ein solches Programm ist z.B. Nmap. Damit kann überprüft werden, ob ein Host aktiv ist, ob er offene TCP- und UDP-Ports hat und welches Betriebssystem auf dem überprüften Host läuft. Zudem kann bei offenen Netzwerkports anhand eines RPC Scans überprüft werden, ob RPC fähige Programme dieses Port verwenden.
Nach der Ermittlung der aktiven über das Netzwerk erreichbaren Geräte analysieren die *atc-host-inventory* Plugins das laufende Betriebssystem, die installierte Software und die jeweils aktuellen Patchstände. Auch hier werden bevorzugt bereits implementierte Softwareprodukte integriert. Die *atc-host-inventory* Plugins können sowohl agentenlose wie auch agentenbasierte Applikationen sein. Das hängt davon ab, welche Programme in den überprüften Netzwerken bereits verfügbar sind. Ist die Implementierung von noch nicht vorhandenen Programmen als *atc-net-inventory* Plugins mit einem relativ geringen Aufwand durchzuführen, sollte bei *atc-host-inventory* Plugins versucht werden, bereits bestehende Lösungen zu verwenden. Dies liegt daran, daß diese Plugins wesentlich tiefer in die zu überprüfenden Systeme eingreifen, und dementsprechend der Installations- und Integrationsaufwand höher ist. Kommen bei den netzwerkbasierenden Werkzeugen meist nur agentenlose Applikationen, die keine Serversoftware im Hintergrund benötigen, zum Einsatz, ist dies bei der Hostinventarisierung nicht der Fall. Applikationen, die ein Asset Management ermöglichen, benötigen im Regelfall ein Backend, das die zeitpunktunabhängige Verwaltung der gesammelten Daten ermöglicht.
Beide Plugintypen schreiben ihre Ergebnisse in die zentrale *atc-inventory* XML Datei. Im ersten Schritt werden die aktiven Geräte ermittelt. Die gesammelten Informationen über die Adressen, Computernamen und Netzwerkports werden zusammen mit einem Zeitstempel in die XML Datei geschrieben. Daraufhin wird die installierte Software und das installierte Betriebssystem der in der XML Datei gespeicherten Computersysteme überprüft. Für diese hostbasierende Überprüfung werden die zur Verfügung stehenden Ap-

plikationen aufgerufen, und die konsolidierten Daten werden wiederum in die zentrale XML Datei geschrieben. In Abbildung 5.5 ist ein detailliertes Ablaufdiagramm von *atc-inventory* dargestellt.

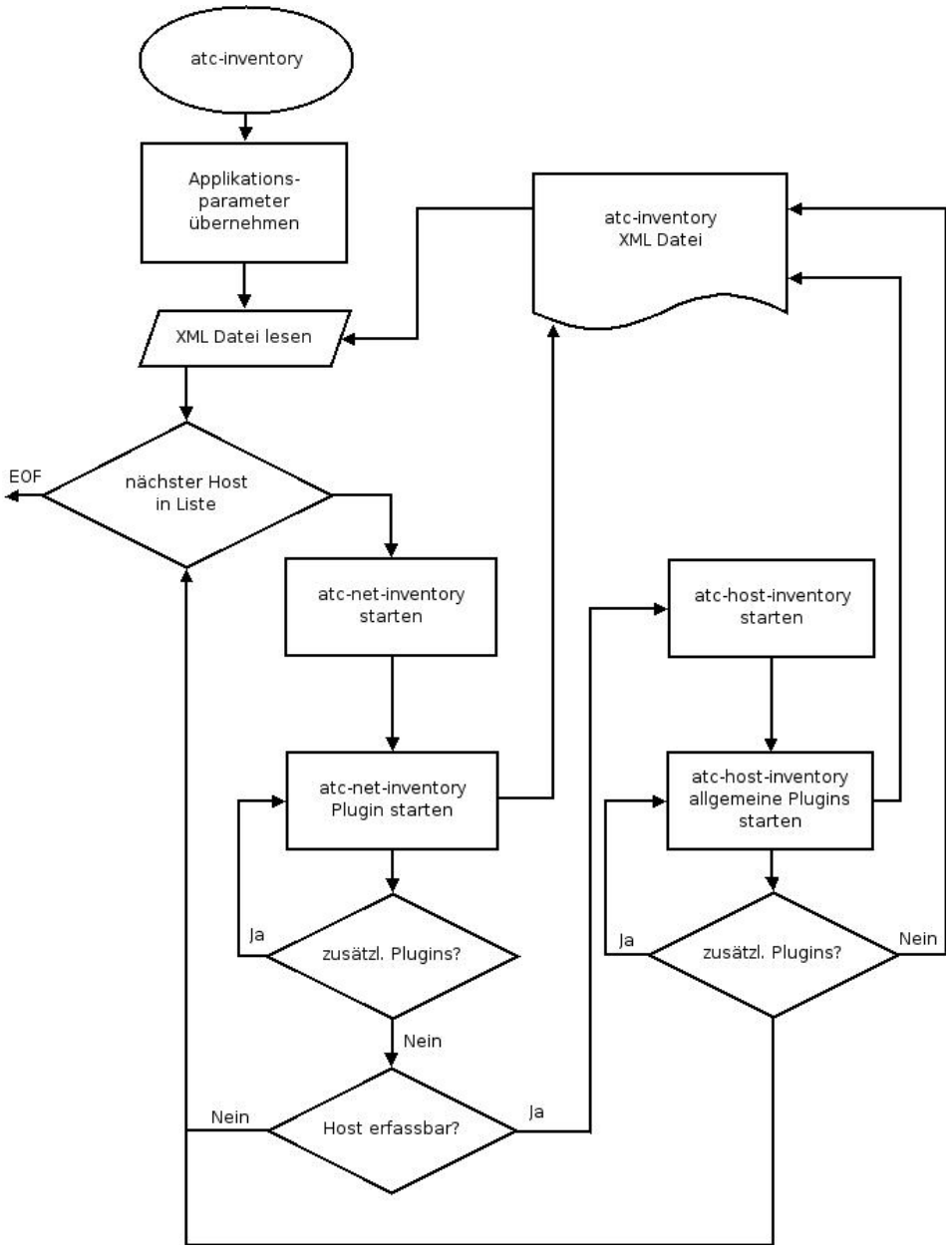

Abbildung 5.5: detailliertes Design *atc-inventory*

Zuerst werden die Parameter eingelesen. Mit diesen Vorgaben werden die zur Verfügung stehenden Sensoren definiert. Nach der Übernahme der Anwendungsparameter wird, falls verfügbar, eine bestehende *atc-inventory* XML Datei eingelesen oder eine neue XML Ausgabedatei initialisiert. Nach Initialisierung aller benötigten Komponenten werden die in der Konfigurationsdatei definierten Plugins und Sensoren für die Netzwerkinventarisie-

rung gestartet. Stehen mehrere Sensoren zur Verfügung, werden zuerst die Sensoren mit einer niedrigen Priorität aufgerufen und zum Schluß jene mit einer hohen Priorität. Nach Abarbeitung der zur Verfügung stehenden netzwerkbasierenden Plugins, werden die hostbasierenden Plugins aufgerufen. Auch hier werden die Plugins mit der höchsten Priorität zum Schluß gestartet. Damit ist die Überprüfung des Systems abgeschlossen und alle benötigten Informationen sind in der zentralen XML Datei gespeichert. Falls beim Start von *atc-inventory* eine Liste von Computern zum Überprüfen angegeben wurde, prüft *atc-inventory* den nächsten Host, ansonsten wird das Programm beendet.

5.2.3 Konzeption der Schnittstellen für die Plugins und Sensoren

Die Konzeption der Schnittstellen für die in die Funktionalität von *atc-inventory* integrierten Applikationen ist ein zentraler Punkt im Design. Da die Applikation leicht erweiterbar sein soll, müssen die Schnittstellen genau definiert sein, jedoch trotzdem die Flexibilität mitbringen, neue Applikationen bzw. Sensoren für erweiterte Funktionalitäten einzubinden. Jedes Plugin muß in der zentralen *atc-inventory* Plugin Konfigurationsdatei mit den für die Applikation relevanten Parametern definiert sein. Dazu zählen:

- der Pluginname
- eine Beschreibung der Applikation
- der *atc-inventory* Typ. Derzeit kann das `host` oder `net` sein
- ob das Plugin verwendet werden soll
- wie die Software aufgerufen wird
- eine Priorität, welche die Datenqualität definiert
- für welche Betriebssysteme die Software verwendet werden kann
- falls benötigt, der Benutzer und das Passwort, unter dem die Applikation laufen soll

Zu diesen Informationen können pro Sensor zusätzliche applikationsspezifische Parameter definiert werden. Die zentrale Plugindefinitionsdatei ist in XML definiert. Dies ermöglicht die Plattformunabhängigkeit und die Erweiterbarkeit beim Einsatz von neuen Applikationen. Das derzeitige XML Schema der Konfigurationsdatei definiert folgende Elemente:

1. `<plugin name=pluginname>` - Name des Plugins
2. `<description>` - Beschreibung des Plugins

3. <type> - host, net
4. <system> - windows, linux, macosx, solaris
5. <priority> - 1, 2, 3
6. <status> - active, disabled
7. <command> - aufzurufender Befehl in der Notation des Systems, auf dem *atc-inventory* läuft
8. <parameter> - Parameter, die zusätzlich dem aufzurufenden Befehl übergeben werden
9. <user> - Benutzer, mit dem der Befehl ausgeführt werden soll
10. <password> - Passwort des Benutzers
11. <reportfile> - Ausgabedatei einer Applikation, die von *atc-inventory* ausgewertet werden soll
12. <timeout> - Laufzeit, nachdem die Durchführung eines Programms unterbrochen werden soll
13. <dbhost> - Name oder IP Adresse des Datenbankservers
14. <dbport> - Netzwerkport für die Verbindung zum Datenbankserver
15. <dbuser> - Benutzer für die Datenbankverbindung
16. <dbpassword> - Passwort des Datenbankbenutzers
17. <dbname> - Name der Datenbank

Das Element <plugin> definiert anhand des Attributs name den Namen des Plugins. Der Name muß eindeutig sein und wird beim Aufruf des Plugins verwendet. Eine Beschreibung der verwendeten Applikation ist im Element <description> vorgesehen. Diese Beschreibung sollte auch wichtige Informationen zum Plugin beinhalten. Dazu zählen z.B. Abhängigkeiten zu anderen Applikationen, Installationshinweise und spezielle Einschränkungen beim Einsatz auf unterschiedlichen Betriebssystemen. Das Element <type> definiert, ob die integrierte Applikation ein *atc-net-inventory* oder ein *atc-host-inventory* Plugin ist. Dies ist für den Ablauf und den Aufruf des Plugins wichtig. Als Wert enthält dieses Element host oder net. Unter dem Element <system> werden die vom Plugin unterstützten Betriebssysteme gespeichert. Derzeit ist die Unterstützung von Microsoft

Windows, Mac OS X, Linux und Solaris vorgesehen. Die Priorität des Sensors wird unter `<priority>` definiert, die Priorität gilt für den im Element `<type>` beschriebenen *atc-inventory* Typ. Die Priorität wird bei der Vereinheitlichung der gesammelten Daten verwendet. Manche Daten werden von mehreren Sensoren ermittelt. Die Prioritäten weisen den einzelnen Plugins einen Wert zu, anhand dessen die Datenqualität des Plugins bewertet werden kann. Das bedeutet, daß bei redundanten Daten jene Werte verwendet werden, deren Inventarisierungsprogramm den besten Wert hat. Das Element `<priority>` kann folgende Werte besitzen:

Priorität 1 .. die ermittelten Daten stimmen sicher mit den installierten Versionen überein und können von keiner anderen Applikation überschrieben werden

Priorität 2 .. die Daten sind über nicht hundertprozentig verlässliche Mechanismen gesammelt worden und können von Priorität 1 Plugins überschrieben werden

Priorität 3 .. der Wahrheitsgehalt der Daten kann nicht verifiziert werden, daher gelten diese Daten nur als Anhaltspunkt, und können von anderen Plugins überschrieben werden.

Daten, die von nicht dafür vorgesehenen Sensoren erfaßt und in die zentrale XML Datei geschrieben werden, können von spezialisierten Plugins überschrieben werden. Das bedeutet, daß die Priorität von Plugins, deren Typ nicht mit der Art der gesammelten Daten übereinstimmt, gleich Priorität 3 ist.
Als Grundlage für die Bewertung werden die verschiedenen Möglichkeiten und Mechanismen für die Datensammlung der jeweiligen Applikation verwendet. Dies kann gut am Beispiel des installierten Betriebssystems gezeigt werden. Die Informationen über das Betriebssystem werden aufgrund der Architektur der vorhandenen Applikationen typischerweise von *atc-host-inventory* ermittelt. Jedoch können auch *atc-net-inventory* Sensoren das Betriebssystem bestimmen. Nmap z.B. hat die Möglichkeit, über ein OS Fingerprinting das Betriebssystem zu ermitteln. Beim Aktivieren des Parameters für OS Fingerprinting schickt Nmap verschiedene Pakete an den Zielhost, und bestimmt anhand einer Analyse der Antwortpakete das Betriebssystem und die Betriebssystemversion [3]. Die Qualität dieser Daten ist jedoch sehr schlecht, da das Ergebnis nicht immer eindeutig und dementsprechend nicht verlässlich ist. Folgende drei Betriebssysteme wurden z.B. bei mehreren hintereinander durchgeführten Überprüfungen eines Apple Computers mit installierter Mac OS X Version 10.4.9 durch Nmap ermittelt:

- Apple Mac OS X 10.4.8 (Tiger)
- FreeNAS 0.671 (runs FreeBSD 6.1-STABLE)

- Apple Mac OS X 10.3.5 or 10.3.7

Da Nmap als *atc-net-inventory* Plugin definiert ist, hat es keine Priorität für das Bestimmen des Betriebssystems. Ist Nmap jedoch als *atc-host-inventory* Sensor definiert, besitzt Nmap bei dieser Verlässlichkeit eine Priorität 3. Verwendet man als weiteres Plugin z.B. OCS Inventory NG, verbessert sich die Qualität der Definition des installierten Betriebssystems. Bei OCS Inventory NG ermittelt der am Host installierte Softwareagent das installierte Betriebssystem anhand der vom System zur Verfügung gestellten Schnittstellen. Bei Computern mit einem installierten Microsoft Windows sind dies WMI und die lokale Registrierungsdatenbank. Die Qualität dieser Daten ist bereits sehr gut, da die Daten direkt vom Betriebssystem geliefert werden. Da die eingesetzte Software jedoch nicht vom Betriebssystemhersteller zertifiziert ist, wird eine Priorität von 2 definiert. Möchte man ein Priorität 1 Plugin bei Microsoft Windows Rechnern einsetzen, so kann man z.B. auf den Windows Server Update Service (WSUS) zurückgreifen. WSUS ermittelt ebenfalls über die definierten Betriebssystemschnittstellen wie WMI und die Remote Registry das installierte Betriebssytem, die Version und das Service Pack, ist jedoch vom Hersteller als vertrauenswürdig eingestuft. Die vom WSUS gesammelten Daten können von keinem anderen Plugin überschrieben werden.
Das Element `<status>` definiert anhand der Werte `active` oder `disabled`, ob das Plugin verwendet werden soll oder nicht. Das Element `<command>` definiert den von *atc-inventory* aufzurufenden Befehl. Der Befehl muß in einem Format sein, den das dem *atc-inventory* zugrundeliegende Betriebssystem interpretieren kann. Läuft das *atc-inventory* Programm unter Linux oder einem Unix basierenden Betriebssystem lautet der Befehl für Nmap z.B. `/usr/bin/nmap`, wird Windows als Betriebssystem eingesetzt, muß der in der Konfigurationsdatei definierte Befehl in der Windowsnotation stehen. Das Element `<parameter>` enthält zusätzliche Applikationsparameter, die dem im Element `<command>` definierten Befehl übergeben werden. Werden für die Ausführung des Befehls andere Rechte als die des *atc-inventory* Benutzers benötigt, kann in der Konfigurationsdatei anhand der Elemente `<user>` und `<password>` ein Benutzer für die Ausführung des Plugins definiert werden. Die Übergabe der Benutzerinformationen muß von der dem Plugin zugrundeliegenden Applikation unterstützt werden. Das Element `<timeout>` definiert die Zeitspanne, nach der das Programm abbrechen soll. Auch hier ist vorausgesetzt, daß ein Timeout dem vom Plugin verwendeten Programm übergeben werden kann. Dies ist nötig, da der *atc-inventory* Prozess beim Aufruf einer Applikation nicht immer auf Statusinformationen zur Laufzeit zurückgreifen kann. Sinnvoll ist der Einsatz dieses Parameters z.B. bei Überprüfungen von Hosts mit aktivierter Firewall, welche ein *drop* statt eines *rejects* auf eingehende Pakete durchführt. Greift ein Plugin nicht auf ein Programm zurück, sondern analysiert eine bestehende Ergebnisdatei, kann das Element `<reportfile>`

verwendet werden. Damit kann z.B. die Ausgabe von bestehenden third-party Produkten analysiert werden, ohne daß diese für die Analyse gestartet werden müssen. Wird statt der Analyse einer Ausgabedatei eine Verbindung mit einer Datenbank benötigt, kann die Verbindung anhand der Elemente `<dbhost>`, `<dbport>`, `<dbuser>`, `<dbpassword>` und `<dbname>` definiert werden. Bei der Verwendung dieser Parameter ist sicherzustellen, daß die für die Verbindung zur Datenbank benötigten Bibliotheken zur Verfügung stehen. Das ist z.B. bei Mysql und Postgresql Datenbanken in der Regel kein Problem, bei Microsoft SQL Server und Oracle kann dies mit einem größeren Installationsaufwand verbunden sein. Da die *atc-inventory* Konfigurationsdatei in XML zur Verfügung steht, sind pro Plugin nur jene Elemente zu definieren, die von der Applikation benötigt werden. Folgendes Listing zeigt ein Beispiel einer *atc-inventory* Plugin Konfigurationsdatei unter einem Microsoft Windows System:

```
<?xml version="1.0"?>
<pluginlist>
  <plugin name="atcnmap">
    <description>
      scans the target, set a suitable timeout to avoid a long delay
    </description>
    <type>net</type>
    <priority>1</priority>
    <status>active</status>
    <command>c:\programme\nmap\nmap.exe</command>
    <timeout>5m</timeout>
  </plugin>
  <plugin name="atcwsus">
    <description>
      looks for installed software in wsus report xml file. the report file has to be
      created on the wsus server. to create the wsus report file the ComputerStatusToXML
      Utility from the Windows Server Update Services API Samples and Tools can be used.
      The tools can be downloaded on
      http://www.microsoft.com/technet/windowsserver/wsus/20/downloads/tools/default.mspx
    </description>
    <type>host</type>
    <priority>1</priority>
    <status>active</status>
    <reportfile>wus_computerstatus.xml</reportfile>
  </plugin>
  <plugin name="atcmbsa">
    <description>
      looks for installed software with the mbsa tool. this plugin works only under
      Microsoft Windows. the mbsa utility can be downloaded on the microsoft website.
    </description>
    <type>host</type>
    <priority>1</priority>
    <status>disabled</status>
    <command>
      C:\Programme\Microsoft Baseline Security Analyzer 2\mbsacli.exe
    </command>
```

```
37      <user>administrator</user
38      <password>strongPassword</password>
39    </plugin>
40  </pluginlist>
```

Listing 5.2: Beispiel *atc-inventory* XML Plugin Konfigurationsdatei

6 Implementierung

Auf Basis des Designs von *atc-inventory* wird ein Prototyp für Microsoft Windows entwickelt. Dieses Kapitel beschreibt detailliert die Implementierung des Prototyps und die Programmierung der ausgewählten Plugins.

6.1 Implementierung des Prototypen

Das Konzept und die Implementierung von *atc-inventory* basieren auf einem offenen und modularen Ansatz. Der Prototyp wird in der Programmiersprache Python, einer objektorientierten und plattformunabhängigen Programmiersprache, entwickelt. Unter anderem werden Windows, Linux, Unix, Mac OS X, Palm Handhelds und Nokia Mobiltelefone unterstützt. Die Programmiersprache unterliegt einer offenen Lizenz und kann in freien wie in kommerziellen Produkten eingesetzt werden. Python bringt eine große Anzahl von Standardbibliotheken mit. Diese decken unter anderem Reguläre Ausdrücke, Dokumentationserzeugung, XML Verarbeitung, Threading und Datenbanken ab. Zuzätzlich gibt es sehr viele spezialisierte Bibliotheken, die auch für die Inventarisierung von Computersystemen relevant sind, wie z.B. für WMI, die RemoteRegistry, SNMP und SSL. Teilweise sind die einer Bibliothek zugrundeliegenden Technologien plattformabhängig, dementsprechend kann eine solche Bibliothek auch nur auf den unterstützten Plattformen verwendet werden.
Der *atc-inventory* Prototyp implementiert die Inventarisierung von Microsoft Windows Clients. Da das Programm in Zukunft nicht nur Microsoft Systeme inventarisieren soll, wird bei der Entwicklung Wert darauf gelegt, das System plattformunabhängig zu implementieren. Einzelne Plugins können jedoch aufgrund der verwendeten Bibliotheken nur auf einigen Plattformen lauffähig sein.
Das Programm *atc-inventory* basiert zu einem großen Teil auf der Verarbeitung von XML. Neben der Ausgabe der gesammelten hostbasierenden Daten in einer XML Datei, wird auch die Konfiguration im XML Format erstellt. Für die XML Verarbeitung kommt die Bibliothek `xml.dom.minidom`, eine Implementierung des *Document Object Model interfaces* zum Einsatz. Die Bibliothek versucht die Verwendung von DOM zu vereinfachen und ist wesentlich kleiner als die klassische DOM Implementierung von Python.
Das Inventarisieren ganzer Netzwerkbereiche erfolgt thread-basierend. Threads erlauben einer Applikation das parallele Abarbeiten von Aufgaben. Damit *atc-inventory* nicht bei

jedem zu inventarisierenden Computersystem die komplette Inventarisierung abwarten muß, werden die einzelnen Hosts parallel abgefragt. Dies bringt einen Performancegewinn mit sich. Als Thread-Bibliothek wird die Python-Bibliothek `threading` verwendet. Dieses Modul stellt ein *higher-level threading interface* auf Basis des Moduls `thread` zur Verfügung.
Beim Start von *atc-inventory* werden die Parameter für das zu überprüfende Ziel und die XML Ausgabedatei überprüft. Das Ziel kann entweder eine IP Adresse, ein Hostname oder ein Netzwerkbereich sein. Der Prototyp kann maximal ein Klasse C IP Subnetz inventarisieren. Der Netzwerkbereich muß in der Notation `a.b.c.d-e` angegeben werden, wobei `d` die Startadresse und `e` die Endadresse ist. Für die Überprüfung eines kompletten Klasse C Netzes muß man als Parameter z.B. 192.168.0.1-254 angeben. Die Parameter von *atc-inventory* sind folgendermaßen definiert:

```
atc-inventory [-h] [-t <a.b.c.d>|<hostname>] [-n <a.b.c.d-e>] [-X <xmlfile>]

-h                    print this message
-t <a.b.c.d|hostname> target to scan (example: 192.168.0.4, host.example.com)
-n <a.b.c.d-e>        targets to scan (example: 192.168.0.10-15)
-X <xmlfile>          output xml file
```

Wird als Parameter eine IP Adresse oder ein Adressbereich angegeben, wird die Gültigkeit der IP Addresse bzw. der einzelnen IP Adressen überprüft. Für den Prototypen wird hierfür kein kompletter Subnetzrechner implementiert. Es wird nur überprüft, ob die IP Adresse vier Tupel besitzt, und diese jeweils in dem Bereich zwischen 1 und 254 liegen. Folgende Methode überprüft die Gültigkeit der IP Adresse:

```
1  # check if the given ip is a valid ip
2  def validatetarget(ip):
3
4      # check if ip address has 4 numbers separated by dots
5      num=ip.split('.')
6      if not len(num)==4:
7          return False
8
9      # check if numbers are between 0 and 255
10     for n in num:
11         try:
12             if int(n) < 0 or int(n) > 255:
13                 return False
14         except:
15             return False
16
17     return True
```

Listing 6.1: Methode zum Überprüfen der Gültigkeit einer IP Adresse

Anhand des Zielparameters wird eine Liste der zu überprüfenden Computersysteme erstellt, und für jedes Ziel wird bei Vorhandensein eines DNS Servers oder einer lokalen `hosts`-Datei anhand der Thread-fähigen Klasse `gethostandaddress` die IP Adresse und der dazugehörige Hostname ermittelt. Dafür wird die von der Bibliothek `socket` zur Verfügung gestellte Methode `gethostbyaddr` verwendet. Die Bibliothek `socket` ist für alle modernen Betriebssysteme verfügbar und stellt den Zugriff auf das *BSD socket interface* zur Verfügung. Die Methode `gethostbyaddr` gibt für die angegebene IP Adresse den Hostnamen, bestehende Aliase und eine Liste der IP Adressen des Hosts zurück.

```
1  # get the dns hostname and the ip address of a given target
2  # for a better performance the class is threaded
3  class gethostandaddress(Thread):
4
5      # intialize the class
6      def __init__(self, target,ips):
7          Thread.__init__(self)
8          self.target = target
9          self.ips = ips
10
11     # define the commands of the thread
12     def run(self):
13
14         # define variables
15         iplist = []
16
17         # try to get the hostname and the ip address
18         # otherwise write the target ip in the variable
19         try:
20             self.hostandaddress = socket.gethostbyaddr(self.target)
21         except socket.error:
22             iplist.append(self.target)
23             self.hostandaddress = ([],[],iplist)
24
25         # append the hostname and the ip to the ips list
26         self.ips.append(self.hostandaddress)
```

Listing 6.2: Klasse zum Erstellen einer Liste von Hostnamen mit den dazugehörigen IP Adressen

Weiters wird überprüft, ob die angegebene XML Ausgabedatei existiert. Ist bereits eine gültige XML Datei vorhanden, wird diese in ein DOM Objekt eingelesen, ansonsten wird ein leeres DOM Objekt erzeugt. Nach dem Überprüfen der Parameter wird die Konfigurationsdatei `atcplugins.xml` gelesen und die Parameter der einzelnen Plugins ausgewertet. Die Plugins werden anhand des Status in host- und netbasierende Plugins aufgeteilt und nach der plugin-Priorität sortiert.
Nach der Auswertung der Parameter und dem Einlesen der konfigurierten Plugins werden

zuerst alle Zielsysteme anhand der *atc-net-inventory* Plugins überprüft. Dazu wird für jeden Zielhost eine Instanz der Thread-fähigen Klasse `netscantarget` mit den eingelesenen Parametern und Informationen zu den Plugins initialisiert. Die Überprüfungen werden parallel abgearbeitet, und erst nach dem Überprüfen aller Hosts werden die Host-Plugins aufgerufen. Hierfür wird wiederum für jeden Host eine eigene Instanz der Thread-fähigen Klasse `hostscantarget` gestartet und jeder aktive Host wird überprüft.
Die von den einzelnen Plugins zu den jeweiligen Computersystemen gesammelten Informationen werden von einem globalen DOM Objekt verwaltet. Dies hat den großen Vorteil, daß die parallel ausgeführten Überprüfungen die Änderungen in einem einzigen DOM Objekt durchführen. Damit alle Plugins das selbe DOM Objekt verwenden, wird das DOM Objekt einer globalen Variable zugewiesen. Diese globale Variable wird beim Instanzieren eines Plugins dem Plugin als Parameter übergeben. Die einzelnen Plugin sammeln die jeweiligen Daten und schreiben sie in das globale DOM Objekt. Erst wenn alle aktiven Threads beendet sind, wird das DOM Objekt in eine XML Datei geschrieben.

6.2 Implementierung und Beschreibung der ausgewählten Plugins

6.2.1 Nmap

Nmap ist ein Programm zum Erkennen und Auswerten von Computersystemen in einem Netzwerk und wird als Portscanner bezeichnet. Der Name Nmap steht für *Network Mapper*. Nmap verwendet *raw IP packets* zum Erkennen der aktiven Hosts im Netzwerk, der angebotenen Netzwerkdienste und der verwendeten Betriebssysteme. Nmap wurde für das effiziente und schnelle Scannen von ganzen Netzwerken entwickelt, funktioniert zum Analysieren von einzelnen Hosts jedoch auch sehr gut. Nmap ist:

- flexibel: Nmap unterstützt eine große Anzahl von fortgeschrittenen Technologien zum Analysieren von Netzwerken. Dazu zählen verschiedene Portscanning Mechanismen für TCP und UDP basierende Dienste und das Erkennen von Betriebssystemen.
- plattformunabhängig: es werden die meisten Betriebssysteme wie Microsoft Windows, Linux, Mac OS X und alle gängigen Unix Betriebssysteme wie Solaris, die verschiedenen BSD Derivate und HP UX unterstützt.
- frei: die Applikation unterliegt einer open source Lizenz und der Source Code ist frei verfügbar.

- gut dokumentiert

atc-inventory verwendet Nmap um die aktiven Computersysteme in einem Netzwerk zu finden. Zudem werden anhand des Sicherheitsscanners die offenen Netzwerkports ausgewertet, und es kann versucht werden, das Betriebssystem zu bestimmen. Nmap bringt eine sehr große Anzahl von verschiedenen Parametern für spezielle Aufgaben mit. Die gesamte Liste kann durch den parameterlosen Aufruf von Nmap ausgegeben werden. Für die Funktionalität von *atc-inventory* sind jedoch nur einige wenige relevant. Der *atc-inventory* Prototyp implementiert folgende 3 Möglichkeiten von Nmap:

1. `nmap -sP <target>`: ein einfacher *ping scan*, hier wird nur überprüft, ob der Host aktiv ist.

2. `nmap -P0 -sT <target>`: ein *tcp connect scan*. `-P0` bedeutet, daß nicht zuerst überprüft werden soll, ob der Host aktiv ist. Dies ist sehr nützlich, wenn z.B. ICMP Pakete durch eine Firewall geblockt werden. Der TCP Connect Scan ist die Standardmethode von Nmap. Für diese Art der Überprüfung werden keine Administratorenrechte benötigt. Nmap verwendet für diesen Scan das darunterliegende Betriebssystem, welches die Verbindung zum überprüften Host aufbaut. Auf die selbe Art bauen zum Beispiel Webbrowser oder Emailprogramme ihre Verbindungen auf. Nicht nur bringt diese API wesentlich weniger Möglichkeiten als die Verwendung von *raw ip packets* mit, auch ist ein solcher Verbindungsaufbau zeitintensiv und kann vom überprüften Computersystem wesentlich einfacher entdeckt und protokolliert werden.

3. `nmap -P0 -O -sS -sU <target>`: dieser Scan führt einen *tcp syn scan*, einen *udp scan* und eine Betriebssystemerkennung durch. Wiederum wird nicht überprüft ob der Host aktiv ist. Ein SYN Scan ist eine effiziente Möglichkeit, schnell sehr viele Hosts zu überprüfen. Bei einem solchen Scan wird die TCP Verbindung nie komplett aufgebaut. Es wird ein SYN Packet zum Host gesendet, und anhand der Antwort bestimmt Nmap den Status des überprüften Ports. Kommt ein SYN/ACK Packet zurück, bedeutet dies, daß der Port offen ist. Wird mit einem RST Packet geantwortet, ist das Port geschlossen. Kommt überhaupt keine Antwort zurück, ist das Port gefiltert. Ein UDP Scan ist generell schwieriger zu implementieren und langsamer als ein TCP Scan. Nmap schickt hierfür einen leeren *UDP header* an jedes UDP Port. Kommt ein *ICMP port unreachable* Fehler zurück, ist das Port geschlossen. Kommen andere *ICMP unreachable* Fehler zurück, ist das Port gefiltert. Kommt als Antwort ein UDP Packet zurück, ist das Port offen und kommt gar keine Antwort zurück, gilt das Port als offen oder gefiltert. In diesem Fall kann Nmap

nicht mit Sicherheit den Status bestimmen. Da eine UDP Kommunikation statuslos ist, muß ein UDP basierender Dienst nicht eine Antwort schicken.

Ein weiterer wichtiger Konfigurationsparameter für die Verwendung von Nmap ist der Parameter `-host-timeout`. Dieser Parameter definiert nach welcher Zeitspanne Nmap einen Zielhost von der Überprüfung ausschließen soll. Bei manchen Computersystemen dauert eine Überprüfung mit Nmap lange. Der Grund dafür können eine schlechte Netzwerkperformance, Probleme bei der Netzwerkhardware oder -software, Limitierungen bei der Bandbreite oder eine restriktiv konfigurierte Firewall sein. Mit diesem Parameter kann definiert werden, wielange die Überprüfung eines Systems maximal dauern darf und ab wann ein Host übersprungen werden soll. Um Fehler im Überprüfungsergebnis zu vermeiden, werden die bis zu diesen Zeitpunkt gesammelten Daten nicht in das Ergebnis mitaufgenommen. Die Bestimmung eines idealen Werts für diesen Parameter hängt vom zu überprüfenden Netzwerk ab.
Die Konfiguration der Nmap Plugins wird über die Datei `atcplugins.xml` durchgeführt.

```
<plugin name="atcnmapportscan">
  <description>
    a tcp connect scan, -P0 meens no host discovery  - useful if ping
    requests are blocked
  </description>
  <type>net</type>
  <priority>2</priority>
  <status>disabled</status>
  <command>c:\programme\nmap\nmap.exe</command>
  <parameter>-P0 -sT</parameter>
  <timeout>5m</timeout>
</plugin>
<plugin name="atcnmapping">
  <description>
    Ping Scan - go no further than determining if host is online
  </description>
  <type>net</type>
  <priority>3</priority>
  <status>disabled</status>
  <command>c:\programme\nmap\nmap.exe</command>
  <parameter>-sP</parameter>
  <timeout>5m</timeout>
</plugin>
<plugin name="atcnmapos">
  <description>
```

```
    TCP Syn scan, UDP scan and OS detection, due the given timeout nmap
    can report uncompleted informations
  </description>
  <type>net</type>
  <priority>1</priority>
  <status>active</status>
  <command>c:\programme\nmap\nmap.exe</command>
  <parameter>-P0 -O -sS -sU</parameter>
  <timeout>5m</timeout>
  </plugin>
```

Für die einfachere Auswertung der Nmap Scans, ruft *atc-inventory* Nmap mit dem Parameter `-oX` auf. Damit werden die Scanergebnisse in eine XML Datei geschrieben, die in Folge von *atc-inventory* ausgewertet wird. Folgendes Listing zeigt ein Beispiel eines XML Reports von Nmap:

```
<?xml version="1.0" ?>
<?xml-stylesheet href="/usr/share/nmap/nmap.xsl" type="text/xsl"?>
<!-- Nmap 4.10 scan initiated Sun Sep  9 16:43:59 2007 as: nmap -P0 -O -sS -sU -oX
  winscan.xml 192.168.0.3 -->
<nmaprun scanner="nmap" args="nmap -P0 -O -sS -sU -oX winscan.xml 192.168.0.3"
  start="1189349039" startstr="Sun Sep  9 16:43:59 2007" version="4.10"
  xmloutputversion="1.01">
<scaninfo type="syn" protocol="tcp" numservices="1679"
  services="1-1027,1029-1033,1040,1043,1050,1058-1059,1067-1068,1076,1080,1083-1084,
  1103,1109-1110,1112,1127,1139,1155,1158,1178,1212,1214,1220,1222,1234,1241,1248,
  1337,1346-1381,1383-1552,1600,1650-1652,1661-1672,1680,1720,1723,1755,1761-1764,
  1827,1900,1935,1984,1986-2028,2030,2032-2035,2038,2040-2049,2053,2064-2065,
  2067-2068,2105-2106,2108,2111-2112,2120-2121,2201,2232,2241,2301,2307,2401,
  2430-2433,2500-2501,2564,2600-2605,2627-2628,2638,2766,2784,2809,2903,2998,
  3000-3001,3005-3006,3049,3052,3064,3086,3128,3141,3264,3268-3269,3292,3306,
  3333,3372,3389,3421,3455-3457,3462,3531,3632,3689,3900,3984-3986,3999-4000,
  4002,4008,4045,4125,4132-4133,4144,4224,4321,4333,4343,4444,4480,4500,4557,
  4559,4660,4662,4672,4899,4987,4998,5000-5003,5010-5011,5050,5060,5100-5102,
  5145,5190-5193,5232,5236,5300-5305,5308,5400,5405,5432,5490,5510,5520,5530,
  5540,5550,5555,5560,5631-5632,5679-5680,5713-5717,5800-5803,5900-5903,
  5977-5979,5997-6009,6017,6050,6101,6103,6105-6106,6110-6112,6141-6148,6346-6347,
  6400-6401,6502,6543-6544,6547-6548,6558,6588,6666-6668,6699,6881,6969,7000-7010,
  7070,7100,7200-7201,7273,7326,7464,7597,7937-7938,8000,8007,8009,8021,8080-8082,
  8443,8888,8892,9090,9100,9111,9152,9535,9876,9991-9992,9999-10000,10005,10082-10083,
  11371,12000,12345-12346,13701-13702,13705-13706,13708-13718,13720-13722,13782-13783,
  15126,16444,16959,17007,17300,18000,18181-18185,18187,19150,20005,22273,22289,22305,
  22321,22370,26208,27000-27010,27374,27665,31337,32770-32780,32786-32787,38037,38292,
  43188,44334,44442-44443,47557,49400,50000,50002,54320,61439-61441,65301" />
<scaninfo type="udp" protocol="udp" numservices="1487"
  services="1-1025,1028,1030-1032,1034,1043,1058-1059,1067-1068,1080,1083-1084,1110,1155,
  1167,1212,1214,1222,1248,1346-1381,1383-1552,1600,1645-1646,1650-1652,1661-1672,1701,
  1812-1813,1900,1986-2002,2004-2028,2030,2032-2035,2038,2040-2049,2065,2067,2103-2106,
  2108,2201,2232,2241,2307,2401,2430-2433,2500-2501,2627,2784,2948,2967,3049,3130,3141,
```

```
34    3246,3264,3333,3401,3421,3455-3457,3531,3900,3984-3986,3996-3998,4000,4008,4045,
35    4132-4133, 4321,4343,4444,4500,4666,4672,4827,5000-5003,5010-5011,5050,5060,5145,
36    5190-5193,5236,5300-5305,5308,5428,5500,5540,5555,5632,5713-5717,6110-6111,6141-6148,
37    6346-6347,6502,6549,6558,6969,7000-7010,7100,7200-7201,7648-7651,9535,9876,10080,
38    16444,17007,17185,18000,22370,26000,26900,27015,27444,27500,27910,27960,28910,31335,
39    31337,32768,32770-32780,32786-32787,38037,38293,39213,45000,47557,54321" />
40  <verbose level="0" />
41  <debugging level="0" />
42  <host><status state="up" />
43  <address addr="192.168.0.3" addrtype="ipv4" />
44  <address addr="00:04:23:75:12:1A" addrtype="mac" vendor="Intel" />
45  <hostnames><hostname name="win01" type="PTR" /></hostnames>
46  <ports><extraports state="closed" count="3156" />
47  <port protocol="tcp" portid="135"><state state="open" />
48    <service name="msrpc" method="table" conf="3" /></port>
49  <port protocol="tcp" portid="139"><state state="open" />
50    <service name="netbios-ssn" method="table" conf="3" /></port>
51  <port protocol="tcp" portid="445"><state state="open" />
52    <service name="microsoft-ds" method="table" conf="3" /></port>
53  <port protocol="udp" portid="123"><state state="open|filtered" />
54    <service name="ntp" method="table" conf="3" /></port>
55  <port protocol="udp" portid="137"><state state="open|filtered" />
56    <service name="netbios-ns" method="table" conf="3" /></port>
57  <port protocol="udp" portid="138"><state state="open|filtered" />
58    <service name="netbios-dgm" method="table" conf="3" /></port>
59  <port protocol="udp" portid="445"><state state="open|filtered" />
60    <service name="microsoft-ds" method="table" conf="3" /></port>
61  <port protocol="udp" portid="500"><state state="open|filtered" />
62    <service name="isakmp" method="table" conf="3" /></port>
63  <port protocol="udp" portid="1025"><state state="open|filtered" />
64    <service name="blackjack" method="table" conf="3" /></port>
65  <port protocol="udp" portid="4500"><state state="open|filtered" />
66    <service name="sae-urn" method="table" conf="3" /></port>
67  </ports>
68  <os><portused state="open" proto="tcp" portid="135" />
69  <portused state="closed" proto="tcp" portid="1" />
70  <osclass type="general purpose" vendor="Microsoft" osfamily="Windows"
71    osgen="2003/.NET" accuracy="100" />
72  <osclass type="general purpose" vendor="Microsoft" osfamily="Windows"
73    osgen="NT/2K/XP" accuracy="100" />
74  <osmatch name="Microsoft Windows 2003 Server or XP SP2" accuracy="100"
75    line="13532" />
76  </os>
77  <tcpsequence index="9999999" class="truly random" difficulty="Good luck!"
78   values="895D4AA3,74994117,ECDE2B86,F2A32BAA,250C53F5,290659C2" />
79  <ipidsequence class="Incremental" values="D64,D65,D66,D67,D68,D69" />
80  <tcptssequence class="zero timestamp" values="0,0,0,0,0,0" />
81  </host>
82  <runstats><finished time="1189349049" timestr="Sun Sep  9 16:44:09 2007"/>
83    <hosts>
84  <runstats><finished time="1189349049" timestr="Sun Sep  9 16:44:09 2007"/>
85    <hosts up="1" down="0" total="1" />
86  <!-- Nmap run completed at Sun Sep  9 16:44:09 2007; 1 IP address (1 host up)
87    scanned in 14.947 seconds -->
88  </runstats></nmaprun>
```

Listing 6.3: XML Ausgabe von Nmap

Die Nmap Integration in *atc-inventory* findet in der Klasse `atcnmap` statt. Diese eigene Klasse führt den Nmap Scan durch. Für jeden Host wird mit dem Aufruf

```
netsensor = atcnmap(self.hostip,netplugin["command"],
  netplugin["parameter"],netplugin["timeout"],self.inventory_dom)}
```

ein eigenes Objekt mit den für den Scan und die Auswertung benötigten Informationen instanziert. Die benötigten Parameter werden aus den übergebenen Parametern des *atc-inventory* Aufrufs und aus der Konfigurationsdatei `atcplugins.xml` übernommen. Für die Überprüfung des Zielsystems wird mittels der Methode `popen` des Moduls `os` das Programm Nmap aufgerufen. Das Ergebnis wird in eine temporäre XML Datei geschrieben. Diese wird von der Methode `scan2xml` eingelesen, und die relevanten Informationen werden übernommen. Zu den für *atc-inventory* relevanten Informationen zählen folgende XML Elemente und Attribute (gezeigt am beschriebenen Beispiel einer Nmap XML Ausgabe) :

Status des Hosts:

```
<status state="up" />
```

die IP Adresse des Hosts:

```
<address addr="192.168.0.3" addrtype="ipv4" />
```

die Mac Adresse des Hosts:

```
<address addr="00:04:23:75:12:1A" addrtype="mac" vendor="Intel" />
```

der Hostname und der DNS Typ:

```
<hostnames><hostname name="win01" type="PTR" /></hostnames>
```

die offenen TCP und UDP ports:

```
<port protocol="tcp" portid="135"><state state="open" />
  <service name="msrpc" method="table" conf="3" /></port>}

<port protocol="udp" portid="123"><state state="open|filtered" />
  <service name="ntp" method="table" conf="3" /></port>}
```

das Betriebssystem:

```
<osmatch name="Microsoft Windows 2003 Server or XP SP2"
  accuracy="100" line="13532" />
```

Diese Informationen werden von `scan2xml` in das DOM Objekt von *atc-inventory* geschrieben. Die gesammelten Daten schauen im DOM Objekt folgendermaßen aus:

```
<state>up</state>
<addresses>
  <ipaddress>192.168.0.3</ipaddress>
  <macaddress>00:04:23:75:12:1A</macaddress>
</addresses>
<hostnames>
  <hostname>win01</hostname>
  <dnstype>PTR</dnstype>
</hostnames>
<os>Microsoft Windows 2003 Server or XP SP2</os>
<ports>
  <port>
    <status>open</status>
    <protocol>tcp</protocol>
    <id>135</id>
  </port>
  <port>
    <status>open|filtered</status>
    <protocol>udp</protocol>
    <id>123</id>
  </port>
</ports>
```

Konnte Nmap den Hostnamen nicht bestimmen, wird als Hostname `HOST-` zusätzlich der MAC Adresse in das *atc-inventory* DOM Objekt geschrieben. Konnte die MAC Adresse ebenfalls nicht bestimmt werden, lautet der Hostname `HOST-` zusätzlich der IP Adresse.

6.2.2 Windows Server Update Service

Der Windows Server Update Service (WSUS) ist eine Patch- und Updatemanagementsoftware von Microsoft, die die zentrale und automatische Aktualisierung von Microsoft Windows Servern und Arbeitsstationen in Windows Netzwerken ermöglicht. Der Dienst besteht aus einer Server- und einer Clientkomponente. Mit dem WSUS können Systembetreuer Microsoft Updates in lokalen Netzwerken überwachen und installieren. Der WSUS

lädt die Microsoft Updates von den Microsoft Update Servern im Internet und verteilt sie lokal auf die einzelnen Clients. Installiert werden nur die Updates, die der Administrator freigibt. Zusätzlich kann anhand von Computergruppen die Installation für einzelne Bereiche verwaltet und über Richtlinien der Zeitpunkt der Installation definiert und erzwungen werden. Da Microsoft den Windows Server Update Service kostenfrei zur Verfügung stellt, gibt es eine sehr große Verbreitung dieses Dienstes. Die Konfiguration und die Verwendung von WSUS basiert auf einem webbasierten Programm. Dieses Programm stellt alle benötigten Konfigurations- und Verwaltungsmöglichkeiten für den Einsatz dieser Lösung bereit.
Folgende Funktionen beinhaltet der Microsoft Server Update Service:

- Updates für Microsoft Windows, Internet Explorer, Microsoft Office, Microsoft Exchange Server, Microsoft SQL Server, Microsoft ISA Server, Microsoft SMS Server und Microsoft Visual Studio
- automatischer Download von Updates
- Freigabe von bestimmten Updates
- Möglichkeit vor der Installation zu überprüfen, ob das Update benötigt wird
- Definition von Computergruppen
- Synchronisierung von mehreren Update Servern
- Berichte für installierte Updates, fehlgeschlagene Updates, benötigte Updates und Computerstände
- Erweiterbarkeit durch eigene API
- automatisierbare Verwaltung der Updatepolicies
- automatisierbare Verwaltung der automatischen Update-Dienste auf Microsoft Clientsystemen
- automatische Aktualisierung von Clientcomputern
- automatische Ermittlung benötigter Updates

Decken die beschriebenen Möglichkeiten nicht die benötigte Funktionalität ab, gibt es ein eigenes *Platform SDK* [7] für den Microsoft Server Update Service. Anhand dieser API können eigene Programme für die Verteilung, Analyse und Auswertung erstellt werden.

Zur Veranschaulichung der zur Verfügung gestellten Bibliotheken findet man Beispielprogramme im Microsoft Developer Network.
Das Programm *atc-inventory* kann einen vorhandenen Windows Server Update Service in die Funktionalität integrieren. Ein direkter Zugriff auf die WSUS SQL Datenbank seitens third-party Applikationen ist vom Produkt nicht vorgesehen. Auch ist das SQL Schema nicht dokumentiert. Daher verwendet *atc-inventory* die zur Verfügung gestellte API und greift nicht direkt auf den Microsoft SQL Server zu. Für *atc-inventory* sind die pro Computer installierten Updates relevant. Genau für diese Abfrage stellt Microsoft in den WSUS Beispielapplikationen ein Programm bereit. Das Programm ComputerStatusToXML erstellt eine XML Datei mit einer Liste aller beim Windows Server Update Service registrierten Computern und dem Status aller freigegeben Updates für jeden Computer. Das bedeutet, in dieser XML Datei finden sich pro Computersystem alle installierten und nicht installierten Updates, die vom Server zur Verfügung gestellt werden. Einen Auszug eines solchen XML Reports zeigt folgendes Listing:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<ComputerStatus>
    <Computer Name="pc-352.akademie.bildendekunst.ac.at" LastReportedStatus="07.09.2007
       06:18:45">
        <UpdateStatus>
            <Update Title="Security Update for Windows Media Player 10 for Windows XP
              (KB917734)" Status="NotApplicable" />
            <Update Title="Security Update for Windows 2000 (KB888113)" Status=
               "NotApplicable" />
            <Update Title="Visio 2003 Service Pack 2" Status="Installed" />
            <Update Title="Security Update for Windows Server 2003 (KB923689)" Status=
               "NotApplicable" />
            <Update Title="Visio 2003 Service Pack 1" Status="NotApplicable" />
            <Update Title="Security Update for Windows XP (KB871250)" Status=
               "NotApplicable" />
            <Update Title="Security Update for Office 2007 (KB933688)" Status=
               "NotApplicable" />
            <Update Title="Security Update for Microsoft Office Publisher 2007
               (KB936646)" Status="NotApplicable" />
            <Update Title="Security Update for Windows XP (KB914389)" Status=
               "Installed" />
            <Update Title="Office 2003 Service Pack 1 for English User Interface Pack"
               Status="NotApplicable" />
            <Update Title="Security Update for Windows XP (KB824151)" Status=
               "NotApplicable" />
            <Update Title="Security Update for Windows 2000 (KB893086)" Status=
               "NotApplicable" />
            <Update Title="Security Update for Windows XP (KB923689)" Status=
               "Installed" />
            <Update Title="Security Update for Word 2003 (KB934181)" Status=
               "Installed" />
            <Update Title="Security Update for DirectX 9.0 (KB839643)" Status=
               "NotApplicable" />
```

```
34          <Update Title="Update for Office 2007 (KB934391)" Status="NotApplicable" />
35          <Update Title="Security Update for Microsoft Windows (KB824105)" Status=
36             "NotApplicable" />
37          <Update Title="Update for Windows XP (KB930916)" Status="Installed" />
38          <Update Title="Project 2002 Service Pack 1" Status="NotApplicable" />
39          <Update Title="Update for PowerPoint 2003 (KB933669)" Status=
40             "Installed" />
41          <Update Title="Security Update for Windows XP (KB908519)" Status=
42             "Installed" />
43          <Update Title="Security Update for Windows XP (KB914388)" Status=
44             "Installed" />
45          <Update Title="Security Update for Windows XP (KB935840)" Status=
46             "Installed" />
47          <Update Title="Security Update for Windows 2000 (KB923980)" Status=
48             "NotApplicable" />
49          <Update Title="Security Update for Windows XP (KB904706)" Status=
50             "Installed" />
```

Listing 6.4: Auszug XML Ausgabe der freigebenen Updates durch den Windows Server Update Service

Die Applikation *atc-inventory* integriert alle Informationen vom WSUS über diese XML Datei. Das bedeutet die Daten vom WSUS müssen mit der beschriebenen Applikation ComputerStatusToXML exportiert werden. *atc-inventory* liest diese XML Datei ein, und extrahiert die für die Überprüfung relevanten Daten. Dazu zählt das Element `Computer`, anhand dessen Attribut `Name` der Zielcomputer in der generierten XML Datei gesucht wird, und das Element `Update`, dessen Attribute `Title` und `Status` die verfügbaren Microsoft Updates und deren Installationsstatus definieren.
Die Konfiguration des `atcwsus` Plugin sieht folgendermaßen aus:

```
<plugin name="atcwsus">
  <description>
  looks for installed software in wsus report xml file. the report file has
  to be created on the wsus server. to create the wsus report file the
  ComputerStatusToXML Utility from the Windows Server Update Services API
  Samples and Tools can be used. The tools can be downloaded on
  http://www.microsoft.com/technet/windowsserver/wsus/20/downloads/
    tools/default.mspx.
  </description>
  <type>host</type>
  <priority>1</priority>
  <status>active</status>
  <reportfile>wus_computerstatus.xml</reportfile>
</plugin>
```

Der Parameter `reportfile` wird von *atc-inventory* zusammen mit dem Hostnamen des Ziels und dem globalen DOM Objekt der Klasse `atcwsus` übergeben.

```
hostsensor = atcwsus(self.hostname,hostplugin["reportfile"],
  self.inventory_dom)
```

Die Klasse durchsucht die angegebene Auswertungsdatei nach den Zielhost und schreibt die zu diesem System gefundenen Daten in das zentrale DOM Objekt. Die Größe dieser vom WSUS generierten XML Datei hängt von der Anzahl der im Netzwerk vorhandenen Computersysteme ab. Bei größeren Netzwerken kann die Datei schnell über 20 MByte groß werden. Dementsprechend relevant ist die Implementierung einer effektiven und performanten Suche nach Computersystemen und deren installierter Software in dieser XML Datei. *atc-inventory* verwendet dafür das Modul `xml.sax`. Das Durchsuchen der XML Datei ist in der Klasse `findsoftware` definiert. Diese Klasse sucht zuerst den Zielhost in der XML Ausgabe vom WSUS, und durchsucht dann die Elemente `Update` nach den Attributen `Status`. Ist der Status `Installed`, wird der im Attribut `Title` definierte Softwarename der Liste der installierten Software hinzugefügt.

```
1  # find the installed software
2  class findsoftware(handler.ContentHandler):
3
4      # initialize the class
5      def __init__(self, target, installed_software):
6          # initialize class variables
7          self.target = target
8          self.installed_software = installed_software
9          self.destinationhost = 0
10
11     # get the needed informations about installed software
12     def startElement(self, name, attributes):
13
14         # if it's the searched computer name, save the software title
15         if name == 'Computer':
16
17             # search the target in the xml file and set a variable destinationhost true,
18             # otherwise set it to false
19             if attributes.get('Name', None) == self.target:
20                 self.destinationhost = 1
21             else:
22                 self.destinationhost = 0
23
24         # if the xml entry is an update, check if it is installed on the destination host
25         # and save the Softwarename
26         elif name == 'Update':
27             if self.destinationhost == 1:
28                 SoftwareTitle = attributes.get('Title', None)
29                 if SoftwareTitle:
30                     SoftwareStatus = attributes.get('Status', None)
```

```
31                      if SoftwareStatus == "Installed":
32                          self.installed_software.append(SoftwareTitle)
```

Listing 6.5: Klasse zum Finden der installierten Software eines Hosts mittels dem WSUS Plugin

Die Klasse `atcwsus` übergibt der Klasse `findsoftware` neben dem Zielhost eine leere Liste für die installierten Applikationen und bekommt eine Liste mit den installierten Softwarepaketen zurück. Nach dem Aufruf schreibt `atcwsus` die installierten Applikationen in das zentrale DOM Objekt. Folgendes Beispiel zeigt einen Auszug des DOM Objekts:

```
<softwarelist>
  <software>
    <vendor>Microsoft</vendor>
    <product>Visio 2003 Service Pack 2</product>
    <version>2003</version>
    <patchlevel>Service Pack 2</patchlevel>
  </software>
    <software>
    <vendor>Microsoft</vendor>
    <product>Security Update for Windows XP (KB914389)</product>
    <version>XP</version>
    <patchlevel>KB914389</patchlevel>
  </software>
  <software>
    <vendor>Microsoft</vendor>
    <product>Security Update for Word 2003 (KB934181)</product>
    <version>2003</version>
    <patchlevel>KB934181</patchlevel>
  </software>
```

Ein zentraler Punkt aller hoch priorisierten Plugins wie dem `atcwsus` Plugin ist die Konsolidierung der gesammelten Daten. Da das `atcwsus` Plugin mit der definierten Priorität 1 am Ende des Überprüfungszyklus aufgerufen wird, muß das Plugin vor dem Schreiben der gesammelten Softwareinformationen überprüfen, ob die einzelnen Applikationen und Updates bereits im globalen DOM für das überprüfte System vorhanden sind. Und gegebenenfalls muß das Plugin die gefundenen Softwareeinträge löschen. Dies wird mit der Funktion `lookupsoftware` durchgeführt. Diese Funktion wird für jede gefundene installierte Software vor dem Schreiben in das globale DOM aufgerufen. Übergeben wird der Funktion der Softwarename, das Patchlevel der Software und der DOM Knoten des überprüften Hosts.

Folgendes Listing zeigt die Funktionen `lookupelement` und `lookupsoftware`. Die generische Funktion `lookupelement` wird von `lookupsoftware` aufgerufen.

```
1
2  # generic helperfunction to get a element in the dom (for example for lookupsoftware)
3  # but can used also for other functions
4  def lookupelement(self, tagname, dom, **kwargs):
5
6      # define variables
7      taglist = []
8
9      # check if there is a dom
10     if not dom:
11         return None
12
13     # get the dom element, check if the element has the right tagname
14     # and append it to the taglist list
15     tags = dom.getElementsByTagName(tagname)
16     for tag in tags:
17         i = 0
18         for attribute,value in kwargs.items():
19             tagelements = tag.getElementsByTagName(attribute)
20             if len(tagelements) != 1:
21                 continue
22             if tagelements[0].firstChild.data == value:
23                 i+=1
24         if i == len(kwargs):
25             taglist.append(tag)
26
27     return taglist
28
29 # search the softwareproduct in the dom
30 def lookupsoftware(self, product, patchlevel, dom):
31
32     # define variables
33     domelements = []
34
35     # set regexp query for Microsoft Patchlevels (KBxxxxxx) and search the product
36     # in the dom a microsoft patchlevel KB-number is enough to identify a installed
37     # software
38     re_patchlevel = "KB[0-9]{6}"
39     if re.search(re_patchlevel, patchlevel):
40         domelements = self.lookupelement("software", dom, patchlevel=patchlevel)
41         if len(domelements) == 1:
42             return domelements[0]
43
44     # set regexp query for Service Packs and search the product in the dom
45     # to identify a installed software with the servicepack, the patchlevel and the
46     # productname has to be analyzed
47     re_patchlevel = "Service Pack [0-9][a-z]?"
48     if re.search(re_patchlevel, patchlevel):
49         if product:
50             domelements = self.lookupelement("software", dom, product=product,
51                 patchlevel=patchlevel)
```

```
52              if len(domelements) == 1:
53                  return domelements[0]
54
55      # look if the productname is in the dom
56      domelements = self.lookupelement("software", dom, product=product)
57      if len(domelements) == 1:
58          return domelements[0]
59      else:
60          return None
```

Listing 6.6: Funktion zum Suchen von Softwareelementen im globalen *atc-inventory* DOM

Der im Prototyp implementierte Algorithmus zum Analysieren der im DOM bereits vorhandenen Software setzt voraus, daß die einzelnen Plugins den selben Softwarenamen für die einzelnen inventarisierten Applikationen an *atc-inventory* zurückgeben. Dieser kann jedoch aufgrund der eingesetzten Plugintechnologien variieren. Dementsprechend ist, bei der Erweiterung von *atc-inventory* durch zusätzliche Plugins, dieser Algorithmus an neu erstellten Plugins anzupassen.

6.2.3 Microsoft Baseline Security Analyzer

Der Microsoft Baseline Security Analyzer (MBSA) ist ein Sicherheitsscanner. Der Scanner ermöglicht das Überprüfen von Microsoft Windows Computersystemen auf Basis der von Microsoft veröffentlichten Sicherheitsratschläge. Hauptzielgruppe sind Klein- und Mittelbetriebe. Aufbauend auf die Windows Update Infrastruktur und dem lokalen Windows Update Agenten überprüft der MBSA den Updatestatus des Hosts und die Konfiguration des Betriebssytems und einzelner Microsoft Applikationen. Der Microsoft Baseline Security Analyzer beinhaltet folgende Funktionalitäten

- Überprüfen von fehlenden Updates für Microsoft Windows, Microsoft Office, Microsoft Exchange Server, Microsoft SQL Server, Microsoft IIS Server, Internet Explorer, Windows Media Player, Windows Messenger
- Überprüfen der Konfiguration der Windows Firewall, des Automatischen Update Dienstes, der Passwortrichtlinie und der Benutzergruppen
- Unterstützung lokaler und netzwerkbasierender Überprüfungen
- Ausgabe von Lösungsvorschlägen beim Auffinden von fehlenden Updates oder Konfigurationsfehlern
- Unterstützung von CVE-ID's

- Kompatibilität mit dem Windows Server Update Service
- über GUI und über die Kommandozeile verwendbar

Für *atc-inventory* sind jedoch nicht alle Funktionalitäten von MBSA relevant. *atc-inventory* greift nur auf das Überprüfen der installierten Updates zurück. Hierfür wird MBSA über das Kommandozeilenprogramm `mbsacli.exe` mit den Parametern `/n OS+IIS+SQL` aufgerufen. Zudem wird mit dem Parameter `/wi` definiert, daß die Microsoft Update Webseite als Updateverzeichnis verwendet werden soll. Dies ist nützlich, wenn kein lokaler WSUS Server vorhanden ist. Das Ergebnis von MBSA wird in eine XML Datei geschrieben. Das folgende Listing zeigt eine solche XML Ausgabe von MBSA.

```
<XMLOut>
  <Check ID="500" Grade="5" Type="5" Cat="1" Rank="1"
    Name="Office Security Updates" URL1="Help/Check5311.html"
    URL2="Help/Check5311fix.html" GroupID="477b856e-65c4-4473-
    b621-a8b230bb70d9" GroupName="Office">
  <Advice>No security updates are missing.</Advice>
  <Detail>
    <UpdateData ID="887622" GUID="e643dc24-bc71-47ff-a4b7-01a0555a2069"
      BulletinID="" KBID="887622" Type="2" IsInstalled="true" Severity="0"
      RestartRequired="false">
      <Title>Visio 2003 Service Pack 2</Title>
      <References>
        <InformationURL>
        http://www.microsoft.com/downloads/details.aspx?FamilyId=C36D4B49-1910-4FF3-
        A525-47368F6E5FDD&displaylang=en
        </InformationURL>
        <DownloadURL>
        http://www.download.windowsupdate.com/msdownload/update/v3-19990518/cabpool/
        visiosp2_f42c0b34da5850f40d70875b1bb19b9a0a2fc366.cab
        </DownloadURL>
      </References>
    </UpdateData>
    <UpdateData ID="MS06-039" GUID="872266b6-1140-446a-8778-3e2987df94b3"
      BulletinID="MS06-039" KBID="914455" Type="1" IsInstalled="true" Severity="2"
      RestartRequired="false">
      <Title>Security Update for Office 2003 (KB914455)</Title>
      <References>
        <BulletinURL>
        http://www.microsoft.com/technet/security/bulletin/MS06-039.mspx
        </BulletinURL>
            <InformationURL>
        http://www.microsoft.com/downloads/details.aspx?FamilyId=66C15CD1-A33B-4EB4-
        9D90-87DECF053768&displaylang=en
        </InformationURL>
        <DownloadURL>
        http://www.download.windowsupdate.com/msdownload/update/v3-19990518/cabpool/
        gpfiltff_63ddd23e9cd5f526ffdcd859e0052b436fb8b03b.cab
        </DownloadURL>
      </References>
```

```
40      </UpdateData>
41    </Detail>
42    </Check>
43  </XMLOut
```

Listing 6.7: Auszug XML Ausgabe der installierten Updates durch den Microsoft Baseline Security Analyzer

Um mit dem MBSA entfernte Rechner über das Netzwerk überprüfen zu können, muß in manchen Fällen ein Benutzername und ein Passwort dem Kommandozeilenprogramm übergeben werden. Diese Parameter können anhand der Datei `atcplugins.xml` konfiguriert werden. Die Konfiguration von MBSA sieht folgendermaßen aus:

```
<plugin name="atcmbsa">
  <description>
  looks for installed software with the mbsa tool. this plugin works
  only under Microsoft Windows. the mbsa utility can be downloaded
  on the microsoft website
  </description>
  <type>host</type>
  <priority>1</priority>
  <system>windows</system>
  <status>disabled</status>
  <command>
    C:\Programme\Microsoft Baseline Security Analyzer 2\mbsacli.exe
  </command>
  <user>username</user>
  <password>password</password>
</plugin>
```

Die Überprüfung eines Zielhosts anhand des Microsoft Baseline Security Analyzer wird in der Klasse `atcmbsa` durchgeführt. Dieser Klasse werden die in der *atc-inventory* Konfigurationsdatei definierten Parameter übergeben. Zudem benötigt `atcmbsa` den Zielhost und das globale DOM Objekt.

```
hostsensor=atcassets(self.hostname,hostplugin["command"],
  hostplugin["user"],hostplugin["password"],
  hostplugin["parameter"],self.inventory_dom)
```

Die Klasse `atcmbsa` führt zuerst einen Scan mit den übergebenen Parametern durch. Ob ein Username und das Passwort übergeben werden, wird in der Datei `atcplugins.xml` definiert. Das Programm führt die Überprüfung durch und speichert standardmäßig das Ergebnis im Windows Profil des ausführenden Benutzers. Die XML Datei wird im durch

die Installation von MBSA erstellten Verzeichnis `SecurityScans` im Windows Profil des Benutzers abgelegt. Die Ausgabe XML Datei wird von *atc-inventory* eingelesen und die benötigten Informationen werden übernommen. Zu den für *atc-inventory* relevanten Informationen zählen ausschließlich die Daten bzgl. installierter Software. Zusätzliche Informationen zu den einzelnen Applikationen werden nicht ausgewertet. *atc-inventory* sucht in der XML Ausgabedatei zuerst das Element `UpdateData`. Dieses Element hat ein Attribut `IsInstalled`. Hat dieses als Wert `true`, wertet das `atcmbsa` das Element `Title`, das den Softwarenamen enthält, aus.
Das Plugin `atcmbsa` hat wie `atcwsus` eine Priorität von 1. Dementsprechend muß das Plugin auch überprüfen, ob die gefundenen Applikationen bereits von einem weniger priorisierten Plugin in das globale DOM geschrieben wurden. Das Plugin verwendet hierfür die bereits beim Plugin `atcwsus` beschriebene Funktion `lookupsoftware` zum Suchen und Löschen von Softwareelementen im globalen DOM.

6.2.4 atc-assets

Als viertes Plugin kommt das selbst entwickelte *atc-assets* zum Einsatz. Dieses Plugin dient als Beispiel für die Integration von selbst entwickelten third-party Produkten. *atc-assets* ist ein Python Programm zum Auslesen der installierten Software auf Microsoft Windows Clients mittels WMI und der Remote Registry. *atc-inventory* ruft das Programm auf, wertet die von *atc-assets* gesammelten Daten aus und schreibt sie in die *atc-inventory* XML Ausgabedatei. *atc-assets* implementiert vier unterschiedliche Aufgaben:

1. das Auslesen der Betriebssysteminformationen mittels WMI
2. das Auslesen der über den Microsoft Installer installierten Software Pakete (msi-Pakete) mittels WMI
3. das Auslesen der installierten Hotfixes mittels WMI
4. das Auslesen der installierten Applikationen mittels der Remote Registry

atc-assets funktioniert nur unter Windows, da die Funktionalität mit nur für Windows verfügbaren Bibliotheken implementiert ist, und kann folgendermaßen aufgerufen werden:

```
atc-assets.exe [-h] [-t <a.b.c.d>[-e]] [-u <username> -p <password>]
  [-o] [-f] -[-m] [-r] [-v] [-X <xmloutputfile>]
```

`-h`: Ausgabe des Hilfetextes

`-t <a.b.c.d>[-e]`: das zu überprüfende Ziel. Als Parameter kann eine IP Adresse oder ein IP Adressen-Bereich angegeben werden (Beispiel: 192.168.0.4, 192.168.0.10-15)

`-u <username>`: Benutzernamen mit genügend Systemrechten auf dem Zielhost zur Durchführung der Überprüfung

`-p <password>`: Benutzerpasswort

`-o`: Überprüfung der Betriebssysteminformationen

`-f`: Überprüfung der installierten Hotfixes

`-m`: Überprüfung der mittels dem Windows Installer installierten Software

`-r`: Überprüfung der installierten Software mittels der Remote Registry

`-v`: Ausgabe der gesammelten Informationen auf dem Bildschirm

`-X <xmloutputfile>`: Ausgabe der gesammelten Informationen in eine XML Datei

Beim Start von *atc-assets* werden zuerst die Parameter überprüft. Es muß zumindest ein Ziel angegeben werden. Sind keine zusätzlichen Parameter angegeben, wird zumindest mittels WMI der Hostname, die IP Adresse und die MAC Adresse ausgelesen. Standardmäßig wird das Ergebnis am Bildschirm ausgegeben. Werden die Informationen in eine XML Datei geschrieben, muß zur zusätzlichen Ausgabe am Bildschirm der Parameter `-v` angegeben werden. Die Zielinformation muß entweder eine IP Adresse oder einen IP Adress Bereich in Form von `a.b.c.d-e` haben, wobei `d` die Startaddresse und `e` die Zieladresse ist. Um die Überprüfung effizienter und performanter durchzuführen, wird für jedes Ziel ein eigener Thread gestartet.
Die Überprüfungen selbst werden mittels WMI und mittels der Remote Registry durchgeführt. Für die Implementierung werden die Python Windows Module `win32com`, `wmi` und `_winreg` verwendet. Bei den beiden Verbindungsarten WMI und RemoteRegistry wird eine direkte Verbindung zum Zielsystem aufgebaut. Um bei ausgeschalteten Computersystemen nicht auf das normale WMI Verbindungstimeout warten zu müssen, wird jedes Ziel zuerst anhand des `ping`-Befehls überprüft. Ist ein Computersystem aktiv, wird überprüft, ob zu dem Zielhost eine WMI Verbindung aufgebaut werden kann.

```
1 # connect to the target with wmi
2 try:
3
4         pythoncom.CoInitialize()
5         wmiservice=win32com.client.Dispatch("WbemScripting.SWbemLocator")
6
7         # if user and password are given as parameter, connect with the given user
8         if self.user and self.password:
9                 swbemservices=wmiservice.ConnectServer(self.target,strUser=self.user,
```

```
10                      strPassword=self.password)
11
12          # else connect with the running user
13          else:
14                  swbemservices=wmiservice.ConnectServer(self.target)
15
16          c = wmi.WMI(wmi=swbemservices)
17
18  # check if the target is wmi enabled and has no firewall running
19  except com_error:
20          print "check if " + self.target + " is wmi enabled and has no firewall running"
21          self.assets_root = None
22          return
```

Listing 6.8: Aufbau und Überprüfung der WMI Verbindung zum Zielsystem

Kann die Verbindung nicht aufgebaut werden, wird das Programm beendet. Gründe dafür können eine aktivierte Firewall, nicht laufende Dienste wie die RemoteRegistry oder ein nicht Microsoft Windows System sein.
Die Informationen über das Betriebssystem werden mittels der Microsoft WMI Klasse `Win32_OperatingSystem` abgerufen. Für *atc-assets* sind folgende Informationen relevant:

`BuildNumber`: die *Build number* des Betriebssystems

`Caption`: die Beschreibung des Betriebssystems

`Version`: die Version des Betriebssystems

`CSDVersion`: das installierte Service Pack. Ist kein Service Pack installiert, ist diese Zeichenkette leer

Die installierten msi-Pakete sind anhand der `Win32_Product`-Klasse überprüfbar. Auch diese Klasse liefert wesentlich mehr Informationen, als von *atc-assets* benötigt. Folgende Informationen werden von *atc-assets* pro installierter Software gesammelt:

`Caption`: der Softwarename

`Description`: die Softwarebeschreibung

`Version`: die Produktversion

`Vendor`: der Name des Softwareherstellers

Die installierten Hotfixes können ebenfalls mittels WMI abgefragt werden. Diese Informationen finden sich in der Klasse `Win32_QuickFixEngineering`.

`Caption`: Name des Hotfixes

`Description`: Beschreibung des Hotfixes

`HotFixID`: eindeutige Bezeichnung des Hotfixes

Für die Abfrage der installierten Applikationen mittels dem RemoteRegistry Dienst werden die Uninstall Registry Einträge ausgewertet. Die Verbindung zur RemoteRegistry wird nicht mit der WMI Klasse, sondern mit dem Python Modul `_winreg` durchgeführt.

```
# connect to the target through the remote registry service
try:
        key = _winreg.ConnectRegistry(self.target, _winreg.HKEY_LOCAL_MACHINE)

# check if connection is possible, otherwise return
except WindowsError:
        print "check if " + self.target + " has remote registry configured"
        return

# define the software uninstall registry hive
hkey = _winreg.OpenKey(key,
r"Software\Microsoft\Windows\CurrentVersion\Uninstall")

# open the the connection to the defined hive
num = _winreg.QueryInfoKey(hkey)[0]
```

Listing 6.9: Verbindung zum RemoteRegistry Dienst des Zielsystems zum Auslesen der installierten Applikationen

atc-inventory bindet alle vier Möglichkeiten von *atc-assets* ein. Die vier *atc-assets* Plugins werden in der Datei `atcplugins.xml` konfiguriert:

```
<plugin name="atcassetssw">
  <description>
    looks for installed software with the atc-assets prototype tool through
    the remote registry service. this plugin works only under Microsoft
    Windows. to get the software information through the atc-assets utility,
    the running user must have enough privileges on the target host
  </description>
  <type>host</type>
  <priority>3</priority>
  <system>windows</system>
  <status>disabled</status>
  <command>D:\daten\inventory\atc-assets\atc-assets.exe</command>
  <parameter>-r</parameter>
```

```
</plugin>
<plugin name="atcassetsos">
  <description>
    looks for installed operating system through wmi. this plugin works only
    under Microsoft Windows.
  </description>
  <type>host</type>
  <priority>2</priority>
  <system>windows</system>
  <status>disabled</status>
  <command>D:\daten\inventory\atc-assets\atc-assets.exe</command>
  <parameter>-o</parameter>
</plugin>
<plugin name="atcassetsmsi">
  <description>
    looks for installed msi packages through wmi. this plugin works only
    under Microsoft Windows.
  </description>
  <type>host</type>
  <priority>2</priority>
  <system>windows</system>
  <status>disabled</status>
  <command>D:\daten\inventory\atc-assets\atc-assets.exe</command>
  <parameter>-m</parameter>
</plugin>
<plugin name="atcassetshotfix">
  <description>
    looks for installed hotfixes through wmi. this plugin works only
    under Microsoft Windows.
  </description>
  <type>host</type>
  <priority>2</priority>
  <system>windows</system>
  <status>disabled</status>
  <command>D:\daten\inventory\atc-assets\atc-assets.exe</command>
  <parameter>-f</parameter>
</plugin>
```

Bei dieser Pluginkonfiguration sieht man unterschiedliche Prioritäten bei den WMI basierenden Plugins und dem Plugin, das die RemoteRegistry verwendet. WMI basierende

Plugins haben die Priorität 2, da WMI über eine vom Betriebssystemhersteller definierte Schnittstelle seine Informationen liefert. Im Gegensatz dazu besitzt das Plugin für die Abfrage über die RemoteRegistry nur eine Priorität 3. Auch hier wird über eine Schnittstelle des Betriebssystemherstellers auf die Daten zugegriffen. Jedoch sind die Registry Daten für jeden lokalen Administrator leicht zu ändern. Daher sind diese Daten nicht hundertprozentig verlässlich.
atc-inventory bindet die Applikation *atc-assets* mit der Klasse `atcassets` ein. Dieser Klasse werden der Zielhost, die in der Datei `atcplugins.xml` definierten Parameter und das globale DOM Objekt von *atc-inventory* übergeben. Die Methode `scan` ruft die Applikation `atc-assets.exe` mit den übergegebenen Parametern auf. Zudem wird der Parameter `-X` mit einer temporären Datei übergeben. Damit werden die gesammelten Daten von *atc-assets* in eine XML Datei geschrieben. Die XML Datei kann die für *atc-inventory* relevanten Elemente `os`, `msisoftwarelist`, `hotfixes` und `softwarelist` enthalten. Beispiele für diese Elemente sind:

```
<os>
  <osname>Microsoft Windows XP Professional</osname>
  <osversion>5.1.2600</osversion>
  <osbuild>2600</osbuild>
  <servicepack>Service Pack 2</servicepack>
</os>

<msisoftwarelist>
  <software>
    <softwarevendor>Microsoft Corporation</softwarevendor>
    <softwarename>Windows Resource Kit Tools</softwarename>
    <softwareversion>5.2.3790</softwareversion>
    <softwaredescription>Windows Resource Kit Tools</softwaredescription>
  </software>
  <software>
    <softwarevendor>Adobe Systems Incorporated</softwarevendor>
    <softwarename>Adobe Reader 7.0.9 - Deutsch</softwarename>
    <softwareversion>7.0.9</softwareversion>
    <softwaredescription>Adobe Reader 7.0.9 - Deutsch</softwaredescription>
  </software>
  <software>
    <softwarevendor>Microsoft Corporation</softwarevendor>
    <softwarename>Microsoft Office Professional Edition 2003</softwarename>
    <softwareversion>11.0.7969.0</softwareversion>
    <softwaredescription>
      Microsoft Office Professional Edition 2003
```

```
    </softwaredescription>
  </software>
</msisoftwarelist>

<hotfixes>
  <hotfix>
    <hotfixname/>
    <hotfixid>KB923689</hotfixid>
    <hotfixdescription>
      Sicherheitsupdate für Windows XP (KB923689)
    </hotfixdescription>
  </hotfix>
  <hotfix>
    <hotfixname/>
    <hotfixid>KB873333</hotfixid>
    <hotfixdescription>Windows XP-Hotfix - KB873333</hotfixdescription>
  </hotfix>
<hotfixlist>

<softwarelist>
  <software>
    <softwarevendor>Adobe Systems, Inc.</softwarevendor>
    <softwarename>Adobe Photoshop 7.0</softwarename>
    <softwareversion>7.0</softwareversion>
  </software>
  <software>
    <softwarevendor>Microsoft Corporation</softwarevendor>
    <softwarename>Windows XP-Hotfix - KB873333</softwarename>
    <softwareversion>20050114.005213</softwareversion>
  </software>
  <software>
    <softwarevendor>Mozilla</softwarevendor>
    <softwarename>Mozilla Firefox (2.0.0.6)</softwarename>
    <softwareversion>2.0.0.6 (de)</softwareversion>
  </software>
  <software>
    <softwarevendor>Microsoft Corporation</softwarevendor>
    <softwarename>Microsoft Office Professional Edition 2003</softwarename>
    <softwareversion>11.0.7969.0</softwareversion>
  </software>
```

```
  <software>
    <softwarevendor>Adobe Systems Incorporated</softwarevendor>
    <softwarename>Adobe Reader 7.0.9 - Deutsch</softwarename>
    <softwareversion>7.0.9</softwareversion>
  </software>
</softwarelist>
```

Wie man an diesen Beispielen sieht, ist die Datensammlung über die RemoteRegistry die vollständigste.
`atcassets` bindet die Funktionalität von *atc-assets* als vier eigenständige Plugins ein. Dies bringt gegenüber einem einzigen Plugin für alle vier Funktionalitäten vor allem einen Performancegewinn, da der Benutzer *atc-assets* benutzerdefiniert aufrufen kann, und nicht alle zur Verfügung stehenden Möglichkeiten ausführen muß. Wird z.B. eine Hotfixabfrage über WMI nicht benötigt, wird eine WMI Verbindung pro Zielsystem weniger aufgebaut. Oder läuft auf den Zielsystemen kein RemoteRegistry Dienst, dann muß *atc-inventory* nicht auf die RemoteRegistry Verbindungstimeouts warten.
Obwohl die Funktionalität auf vier Plugins aufgeteilt ist, werden alle vier Plugins durch die Klasse `atcassets` abgebildet. Vor der Analyse der *atc-assets* XML Ausgabedatei definiert `atcassets` anhand der Pluginparameter die Bezeichnungen der für die Analyse relevanten XML Elemente. Diese Funktionalität ist in der Funktion `assets2xml` implementiert. Ist als Parameter `-o` angegeben, werden die Elemente `osname` und `servicepack` ausgewertet und in das globale DOM geschrieben. Für die Parameter `-m`, `-f` und `-r` werden Hilfsvariablen verwendet. Diese Hilfsvariablen definieren den Elementnamen des *atc-assets* Softwareeintrags und die Namen der untergeordneten Elemente.

```
# -m means get the softwareinformations through the msi database
if self.parameter == "-m":
    softwaretag = "msisoftwarelist"
    softwareentrytag = "software"
    softwarevendortag = "softwarevendor"
    softwarenametag = "softwarename"
    softwareversiontag = "softwareversion"

# -f means get the hotfix informations
elif self.parameter == "-f":
    softwaretag = "hotfixes"
    softwareentrytag = "hotfix"
    softwarevendortag = "Microsoft"
    softwarenametag = "hotfixdescription"
    softwareversiontag = "hotfixid"
```

```
# -r means get the softwareinformations through the registry
elif self.parameter == "-r":
    softwaretag = "softwarelist"
    softwareentrytag = "software"
    softwarevendortag = "softwarevendor"
    softwarenametag = "softwarename"
    softwareversiontag = "softwareversion"
```

Anhand dieser Hilfsvariablen wird dann die XML Ausgabedatei von *atc-assets* ausgewertet und die konsolidierten Informationen in das globale DOM geschrieben. Für die Datenkonsolidierung wird wiederum auf die bereits beschriebene Funktion `lookupsoftware` zurückgegriffen.

7 Test des Prototypen

Dieses Kapitel beschreibt den Test des Prototypen, der die Funktionalität und die Performance von *atc-inventory* untersuchen soll. Da *atc-inventory* auf die Integration verschiedener Applikationen basiert, hängt die Performance und die Funktionalität stark von der Auswahl und der Konfiguration der Plugins ab. Der Prototyp implementiert insgesamt neun verschiedene Funktionalitäten.

7.1 Testszenario

Die Tests von *atc-inventory* werden in zwei ausgewählten Netzwerkbereichen der Akademie der bildenden Künste Wien durchgeführt.
Für den Test der netzbasierenden *atc-inventory* Plugins wird der DHCP Bereich des Netzwerkbereichs 10.3.51.0/24 der Akademie der bildenden Künste verwendet. Der DHCP Bereich geht von 10.3.51.10 bis 10.3.51.250. In diesem Netzwerk gibt es insgesamt 27 Systeme mit Windows, Linux und Solaris, die durchgehend in Betrieb sind. Die Systeme sind folgendermaßen konfiguriert.

- Neun Windows Systeme, davon ein Host durch eine lokale Firewall geschützt
- Vier Solaris Systeme, wovon zwei virtuelle Systeme sind. Alle vier sind durch eine lokale Firewall geschützt
- Acht Linux Systeme, wovon vier virtuelle Systeme sind. Alle acht sind durch eine lokale Firewall geschützt
- Sechs Appliances mit einem herstellerspezifischen Betriebssystem

Als zweiter Netzwerkblock wird das Klasse C Netzwerk 10.3.3.0 verwendet. Dieses Netzwerk wird für den Test der hostbasierenden Plugins herangezogen. Auch hier wird wiederum der konfigurierte DHCP Bereich untersucht. In diesem Netzwerk befinden sich Windows und Mac OS X Systeme, die nicht durchgehend im Netzwerk aktiv sind. Insgesamt befinden sich in diesem Netzwerk bis zu zwanzig Windows Clientsysteme und bis

zu acht Apple Mac OS X Clients.
Alle Tests werden von einem System im untersuchten Netzwerkbereich durchgeführt, da bei einer Untersuchung von Systemen über eine Subnetzgrenze hinweg die Ergebnisse durch Filterregeln am Router bzw. durch den Einsatz einer Firewall als Router verfälscht werden können. Als Testsysteme wird jeweils ein Windows Client verwendet.

7.2 Test der netzwerkbasierenden *atc-inventory* Plugins

Bei diesem Test werden die Net-Plugins von *atc-inventory* getestet. Diese Plugins haben die längste Laufzeit, man kann aber auch anhand der verwendeten Parameter das Laufzeitverhalten beeinflußen.

7.2.1 Test des Nmap Plugins

atc-inventory implementiert drei verschiedene Net-Plugins anhand von Nmap:

- einen simplen Pingscan (`-sP`)
- einen TCP-Connect Scan (`-sT`) und
- einen TCP/UDP Scan inklusive OS Fingerprinting (`-O -sS -sU`).

Der Pingscan (`-sP`) sendet einen *icmp echo request* und ein TCP Paket zu Port 80, und analysiert, ob der überprüfte Host antwortet. Dieser Scan ist sehr schnell, gibt aber nur die Information aus, ob ein Zielsystem aktiv ist.
Der TCP-Connect Scan (`-sT`) hingegen versucht, die offenen TCP Ports anhand eines Verbindungsaufbaus zu ermitteln. Bei dieser Methode verwendet Nmap das Betriebssystem für den Verbindungsaufbau. Dies hat den Vorteil, daß Nmap nicht von einem administrativen Benutzer gestartet werden muß, der Zugriff auf *raw* Pakete hat. Jedoch bringt dieser Scan ein schlechtes Laufzeitverhalten mit sich.
Die dritte Option ermittelt die meisten Informationen. Einerseits werden anhand eines TCP SYN scans (`-sS`) die offenen TCP Ports ermittelt. Der TCP SYN scan ist relativ schnell, da er keine komplette TCP Verbindung öffnet. Vielmehr schickt dieser Scantyp ein SYN Paket zum Zielhost und analysiert das Antwortpaket. Anhand dieses Pakets kann Nmap bestimmen, ob der Port offen, geschlossen oder durch eine Firewall gefiltert ist. Zudem wird ein UDP Scan (`-sU`) durchgeführt. Bei diesem Scan Typ schickt Nmap einen leeren UDP Header an das Zielsystem und bestimmt wiederum anhand der Antwortpakete den Status der UDP Ports. Als dritte Funktionalität bringt diese *atc-inventory* Option ein OS Fingerprinting (`-O`) mit. Mit diesem Parameter schickt Nmap eine Serie von TCP

und UDP Paketen an das Zielsystem und analysiert die Antwortpakete auf spezielle Paketoptionen. Die Ergebnisse dieser Analyse vergleicht Nmap mit seiner Betriebssystem-Datenbank, die mehr als 1500 bekannte Betriebssystemeigenschaften beinhaltet.
Für alle drei Nmap Plugins kann zusätzlich der `-host-timeout` Parameter definiert werden. Dieser Parameter bestimmt, ab wann die Überprüfung eines Zielsystems abgebrochen werden soll. Ist die Überprüfung eines Zielsystems nach der in dem Parameter spezifizierten Zeitangabe nicht fertig, bricht Nmap die Überprüfung ab. Dementsprechend stark beeinflußt dieser Parameter das Laufzeitverhalten von *atc-inventory*. Beim ping-Scan verändert sich das Verhalten von Nmap durch diesen Parameter nicht. Bei den beiden anderen Scantypen kann eine Anpassung dieses Parameters den Zeitaufwand für die Überprüfung beeinflußen.
Für den Test werden zwei verschiedene Timeout Zeiten verwendet: 5 Minuten und 15 Minuten. Anhand der Testergebnisse wird analysiert, ob sich bei einem Timeout-Wert von 15 Minuten die Vollständigkeit der gesammelten Daten wesentlich verbessert, oder ob diese Anpassung in einem realen Netzwerk nicht relevant ist. Zudem werden die von *atc-inventory* durchgeführten Scans den Nmap-Überprüfungen, die über die Kommandozeile gestartet werden, gegenübergestellt. Bei den über die Kommandozeile gestarteten Überprüfungen werden die gleichen Parameter wie bei der Überprüfung durch *atc-inventory* verwendet. Das Überprüfen mehrerer Zielsysteme durch Nmap wird bei diesen Tests einmal parallel und einmal sequentiell aufgerufen. Dies ist nötig, da *atc-inventory* jeweils einen Nmap Prozess pro Zielsystem startet, und dies einem sequentiellen Aufruf von Nmap entspricht. Ziel dieser Gegenüberstellung ist es, den Overhead von *atc-inventory* zu bestimmen.
Der erste Testlauf ist ein TCP Connect Scan mit einem Timeout Parameter von 5 Minuten, durchgeführt über die Kommandozeile:

```
nmap -P0 -sT --host-timeout 5m -oX nmap_par_tcp_5m.xml 10.3.51.10-250
```

Dieser Scan hat insgesamt 24 Systeme gefunden, und keine offenen TCP Ports. Da dieses Ergebnis nicht den realen Zustand widerspiegelt, wird ein zweiter TCP Connect Scan mit einem Timeout Parameter von 15 Minuten durchgeführt.

```
nmap -P0 -sT --host-timeout 15m -oX nmap_par_tcp_15m.xml 10.3.51.10-250
```

Dieser Scanversuch findet insgesamt 24 Systeme und 69 offene TCP Ports. Auch dies spiegelt nicht das reale Testszenario wider.
Da es sich beim TCP Connect Scan um eine sehr langsame Scanart handelt, sieht man die Auswirkungen vom Parameter `-host-timeout` sehr gut. Ein Timeoutwert von 5 Minuten ist für diese Art der Überprüfung nicht genug. Aufgrund des Scanverhaltens des TCP Connect Scans sollte dieser Scan jedoch nur verwendet werden, wenn der Benutzer keine administrativen Rechte besitzt. Dies ist bei Sicherheitsüberprüfungen jedoch in der Regel nicht der Fall. Dementsprechend kann dieser Scan für den restlichen Performance- und

Funktionalitätstest der hostbasierenden Plugins vernachlässigt werden.
Die weiteren Testläufe werden als kombinierter Scan mit einem SYN Scan, einem UDP Scan und der Betriebssystemerkennung von Nmap durchgeführt. Zuerst wird ein paralleler Nmap Scan mit dem Befehl

```
nmap -P0 -O -sS -sU --host-timeout 5m -oX nmap_par_os_5m.xml 10.3.51.10-250
```

durchgeführt. Der selbe Scan wird zudem mit einem Timeout Parameter von 15 Minuten initiiert. Nach den parallelen Nmap Scans wird auf der Windows Kommandozeile ein sequentieller Scan von Nmap durchgeführt. Dafür wird der Befehl

```
for /L %i in (10,1,250) do
"c:\Program Files\nmap\nmap.exe" -P0 -O -sS -sU --host-timeout 5m 10.3.51.%i
```

aufgerufen. Auch dieser Test beinhaltet zwei Testläufe mit den unterschiedlichen Timeout Parametern.
In Anschluß wird zum Vergleichen der Ergebnisse der kombinierte Scan mit *atc-inventory* aufgerufen. Das einzige aktive Plugin ist *atcnmapos*, alle anderen Plugins werden deaktiviert. Ein Testlauf wird wiederum mit einem Timeout von 5 Minuten durchgeführt, der zweite mit einem Timeout von 15 Minuten.

```
python atc-inventory.py -n 10.3.3.145-150 -X atc-nmapos_5m.xml
```

Die Tabelle 7.1 fasst die Ergebnisse der einzelnen Überprüfungen zusammen. Für alle Scans wurde die Laufzeit, die gefundenen Systeme und die gefundenen Ports analysiert.
Das Ergebnis zeigt, daß unterschiedliche Timeout Parameter bei einem kombinierten Scan hauptsächlich die Laufzeit der Überprüfung beeinflußen. Da diese Scanart relativ schnell ist, beeinträchtigt dieser Parameter nur jene Hosts, die nicht aktiv sind, oder die eine restriktive Firewall einsetzen. Bei einem niederen Timeoutwert wird der Scan für diese Systeme früher abgebrochen, und dies führt zu kürzeren Überprüfungszeiten. Auch sieht man, daß der Zeitmehraufwand wenige zusätzliche Informationen bzw. keine besseren Ergebnisse liefert.
Die Gegenüberstellung der Laufzeit des sequentiellen Kommandozeilenaufrufs von Nmap und dem Aufruf über *atc-inventory* zeigt, daß *atc-inventory* einen vernachlässigbaren Overhead hat. Da *atc-inventory* für jeden Nmap Scan einen eigenen Thread startet, hat *atc-inventory* bei größeren Werten für den Timeout Parameter sogar Vorteile gegenüber dem sequentiellen Aufruf. Die Performance eines parallelen Nmap Scans wird jedoch nicht erreicht.
Das Ergebnis zeigt auch, daß keine der durchgeführten Überprüfungen ein komplett verlässliches Ergebnis liefert, denn kein Überprüfungsdurchlauf hat alle 27 aktiven Systeme erkannt. Dies kann verschiedene Gründe haben:

Aufruf	Dauer	Systeme	TCP Ports	UDP Ports
nmap -P0 -sT –host-timeout 5m	918 sec	25	0	na
nmap -P0 -sT –host-timeout 15m	1717 sec	24	69	na
nmap -P0 -sS -sU -O –host-timeout 5m (parallel)	618 sec	24	105	83
nmap -P0 -sS -sU -O –host-timeout 15m (parallel)	1816 sec	26	104	83
nmap -P0 -sS -sU -O –host-timeout 5m (sequentiell)	2703 sec	26	101	83
nmap -P0 -sS -sU -O –host-timeout 15m (sequentiell)	6000 sec	25	101	83
atc-inventory nmapos timeout 5m	2744 sec	26	104	83
atc-inventory nmapos timeout15m	4112 sec	24	104	83

Tabelle 7.1: Vergleich Nmap mit *atc-inventory*

- Netzwerkports können im laufenden Betrieb geöffnet und geschlossen werden
- die Antwortzeiten bei einer Überprüfung hängt von der aktuellen Auslastung des überprüften Systems ab
- restriktive Firewallkonfigurationen können Überprüfungen verlangsamen
- da die überprüften Systeme produktiv im Einsatz sind, kann das Verhalten durch die aktiven Benutzer verändert werden

Die bei den einzelnen Überprüfungen nicht gefundenen Hosts sind ausschließlich virtuelle Linux Systeme mit einer sehr restriktiven Firewall. Was jedoch verwundert, ist, daß Nmap trotz eines längeren Timeoutparameters nicht mehr Informationen, sondern teilweise sogar weniger Informationen sammelt. Dies ist nicht nur bei den *atc-inventory* Durchläufen der Fall, dieses Verhalten kann bei Nmap Scan über die Kommandozeile nachvollzogen werden. Die detaillierten Ergebnisse über die gefundenen Systeme sind für die beiden mit *atc-inventory* durchgeführten Testläufe in Tabelle 7.2 dargestellt.
Die Tabelle beinhaltet auch die Ergebnisse der Betriebssystemerkennung von Nmap. Auch bei dieser Überprüfung sieht man, daß die Ergebnisse von Nmap nicht verlässlich sind. Von den 27 aktiven Systemen wurde bei maximal 6 Systemen das korrekte Betriebssystem erkannt.
Der Test des Plugins zeigt, daß die Erkennung der aktiven Knoten sehr gut funktioniert,

produktive Systeme		*atc-inventory* 5min		*atc-inventory* 15min	
IP Adresse	System	Host inventarisiert	erkanntes System	Host inventarisiert	erkanntes System
10.3.51.10	Windows	ja	Windows	ja	
10.3.51.11	Windows	ja		ja	
10.3.51.12	Windows	ja		ja	
10.3.51.13	Windows	ja	Windows	ja	
10.3.51.14	Windows	ja		ja	
10.3.51.15	Windows	ja		ja	
10.3.51.16	Windows	ja		ja	
10.3.51.17	Windows	ja	Windows	ja	
10.3.51.18	Windows	ja		ja	Windows
10.3.51.20	Solaris	ja		ja	Solaris
10.3.51.22	Solaris	ja	Solaris	ja	Solaris
10.3.51.23	Solaris VM	ja	Solaris	ja	Solaris
10.3.51.24	Solaris VM	ja	Solaris	ja	Solaris
10.3.51.30	Appliance	ja		ja	
10.3.51.31	Appliance	ja	Panasonic	ja	Panasonic
10.3.51.41	Linux	ja		ja	
10.3.51.42	Linux VM	ja		nein	
10.3.51.44	Linux VM	nein		ja	
10.3.51.46	Linux VM	ja		nein	
10.3.51.48	Linux VM	ja		nein	
10.3.51.49	Linux	ja		ja	
10.3.51.50	Linux	ja		ja	
10.3.51.51	Linux	ja		ja	
10.3.51.201	Appliance	ja		ja	Windows
10.3.51.202	Appliance	ja		ja	
10.3.51.203	Appliance	ja		ja	
10.3.51.204	Appliance	ja		ja	

Tabelle 7.2: identifizierte Systeme mit *atc-inventory*

die Betriebssystemerkennung von Nmap jedoch nicht die gewünschte Wirkung zeigt. Für das Auffinden von aktiven Knoten kann ohne weiteres Nmap als *atc-inventory* Plugin eingesetzt werden. Zur Bestimmung des Betriebssystems sollte jedoch zusätzlich ein hostbasierendes Plugin zum Einsatz kommen.

7.3 Test der hostbasierenden *atc-inventory* Plugins

7.3.1 Test des Microsoft Baseline Security Analyzer Plugins

Bei den hostbasierenden Plugins wird zuerst das `atcmbsa` Plugin getestet. Dieses Plugin implementiert die Funktionalität des Microsoft Security Baseline Analyzer. Um *atc-inventory* mit dieser Microsoft Applikation vergleichen zu können, werden drei Testläufe durchgeführt:

1. ein paralleler, auf der Kommandozeile initiierter MBSA Scan des DHCP Bereichs 10.3.3.10-10.3.3.200:

   ```
   "c:\Programme\Microsoft Baseline Security Analyzer 2\mbsacli.exe"
   /r 10.3.3.10-10.3.3.200 /n OS+IIS+SQL /wi /unicode /o %IP%_mbsa_par.xml
   ```

2. ein sequentieller auf der Kommadozeile mit dem Befehl

   ```
   for /L %i in (10,1,200) do
   "c:\Programme\Microsoft Baseline Security Analyzer 2\mbsacli.exe"
   /target 10.3.3.%i /n OS+IIS+SQL /wi /unicode /o 10.3.3.%i_mbsa_seq.xml
   ```

   durchgeführter MBSA Scan des oben genannten DHCP Bereichs.

3. eine mit *atc-inventory* und dem aktivierten Plugin `atcmbsa` durchgeführte Überprüfung.

   ```
   python atc-inventory.py -n 10.3.3.10-200 -X atcmbsa_10-3-3.xml
   ```

Der Zeitaufwand der drei Überprüfungen ist in Tabelle 7.3 aufgelistet.
Wie bei Nmap wird von *atc-inventory* für jedes Zielsystem ein eigener Microsoft Baseline Security Analyzer Scan gestartet, was wiederum identisch mit einer sequentiellen Überprüfung ist. Auffallend ist der Geschwindigkeitsvorteil von *atc-inventory* bei dieser Art der Überprüfung. Der Grund hierfür liegt darin, daß im Gegensatz zu den auf der Kommandozeile gestarteten MBSA Überprüfungen, *atc-inventory* nur die von den netzwerkbasierenden Plugins bereits als aktiv erkannten Systeme anhand des `atcmbsa` Plugins

Aufruf	Dauer
paralleler MBSA Scan	4080 sec
sequentieller MBSA Scan	12540 sec
MBSA Scan mit *atc-inventory*	501 sec

Tabelle 7.3: Vergleich der Laufzeit des Microsoft Baseline Security Analyzers und *atc-inventory*

überprüft. Bei den über die Kommandozeilen gestarteten Überprüfungen muß der Microsoft Baseline Security Analyzer bei jedem nicht aktiven Zielsystem auf einen Timeout warten. Dies fällt mit *atc-inventory* weg.
Für die Überprüfung der Funktionalität wird stichprobenartig bei vier ausgewählten Zielsystemen überprüft, ob *atc-inventory* die selben Ergebnisse wie die mit dem Microsoft Baseline Security Analyzer manuell durchgeführten Überprüfungen liefert. Dazu wird die Anzahl der pro System gefundenen Softwareinstallationen analysiert. Das Ergebnis ist in Tabelle 7.4 zusammengefasst.

IP Adresse	paralleler MBSA Scan	sequentieller MBSA Scan	MBSA Scan mit *atc-inventory*
10.3.3.33	82	82	82
10.3.3.96	80	80	80
10.3.3.148	97	97	97
10.3.3.149	82	82	82

Tabelle 7.4: Vergleich der gefundenen Softwareapplikationen mit dem Microsoft Baseline Security Analyzer und *atc-inventory*

Wie man an den Ergebnissen sieht, liefern alle drei Überprüfungen die selben Ergebnisse. Dementsprechend kann dieses *atc-inventory* Plugin problemlos eingesetzt werden.

7.3.2 Test des Windows Server Update Service Plugins

Für den nächsten Testlauf kommt das hostbasierende Plugin `atcwsus` zum Einsatz. Bei diesem Test kann kein Performancevergleich durchgeführt werden, da die Originalapplikation von den auf den Clients installierten lokalen Agenten aktualisiert wird. Für den Funktionalitätstest werden wiederum die von *atc-inventory* gefundenen Applikationen den im WSUS gemeldeten Applikationen gegenübergestellt. Der *atc-inventory* Aufruf hat als einziges hostbasierendes Plugin das `atcwsus` Plugin konfiguriert. Obwohl kein Performancevergleich durchgeführt werden kann, wird beim Test zwecks der Vollständigkeit der Zeitaufwand mitaufgenommen.
Die Überprüfung des DHCP Bereichs des Netzwerk 10.3.3.0/24 anhand des Befehls

```
python atc-inventory.py -n 10.3.3.10-200 -X atcwsus_10-3-3.xml
```

dauert 72 Sekunden. Die vom Windows Server Update Service generierte und von *atc-inventory* analysierte XML Datei beinhaltet den Updatestatus aller Windows Clients der Akademie der bildenden Künste Wien und ist 27,1 MB groß. Mit einer Überprüfungszeit von knapp über einer Minute ist die Analyse mit dem `atcwsus` Plugin als genügend performant einzustufen.
Die Tabelle 7.5 zeigt die Ergebnisse des Vergleichs von *atc-inventory* mit der Windows Server Update Service Datenbank. Da die WSUS Überprüfung über den Computernamen und nicht über die IP Adresse durchgeführt wird, ist in der Tabelle der Computername angeführt. Die Analyse basiert wiederum auf Stichproben, als Basis dienen die bereits beim MBSA Plugin ausgewählten Zielsysteme.

IP Adresse	Computername	WSUS Datenbank	WSUS Scan mit *atc-inventory*
10.3.3.33	pc-352.akademie.bildendekunst.ac.at	98	98
10.3.3.96	pc-244.akademie.bildendekunst.ac.at	0	0
10.3.3.148	pc-373.akademie.bildendekunst.ac.at	119	119
10.3.3.149	pc-400.akademie.bildendekunst.ac.at	98	98

Tabelle 7.5: Vergleich der gefundenen Softwareapplikationen anhand des Windows Server Update Service und *atc-inventory*

Das Ergebnis zeigt, daß das Plugin die selben Ergebnisse wie die WSUS Applikation liefert. Daher kann das Plugin für Überprüfungen mit *atc-inventory* eingesetzt werden. Ein Problem dieser Überprüfung wird jedoch durch diese Ergebnisse sichtbar. Der für die Überprüfung mit dem `atcwsus` Plugin benötigte Computername wird von *atc-inventory* anhand eines DNS Servers oder anhand einer lokalen Datei für Hostnamen bestimmt. Die IP Adresse 10.3.3.96 wird vom DNS Server der Akademie der bildenden Künste Wien auf den Namen pc-244.akademie.bildendekunst.ac.at aufgelöst. Dieser DNS Name ist falsch und entspricht nicht dem wirklichen Computernamen. Dementsprechend kann *atc-inventory* diesen Computer in der vom WSUS Server generierten XML Datei nicht finden, und gibt keine Applikationen für diese IP Adresse aus. Auch die händische Überprüfung dieses Computernamens in der WSUS Datenbank liefert kein Ergebnis. Dieses Testergebnis zeigt, wie wichtig ein korrekt konfigurierter und aktualisierter DNS Server für hostbasierende Überprüfungen ist.

7.3.3 Test des *atc-assets* Plugins

Da das zugrundeliegende Programm eigens für die Integration in *atc-inventory* entwickelt ist, gibt es keine Möglichkeit eines objektiven Vergleichs der Performance zwischen dem Originalprogramm und der Verwendung durch *atc-inventory*. Für die Vollständigkeit des Tests wird der Zeitaufwand für jedes Plugin gemessen. In der Tabelle 7.6 ist die Laufzeit für die vier *atc-assets* Plugins aufgelistet. Die Dauer bezieht sich auf die Überprüfung des DHCP Bereichs 10.3.3.10-10.3.3.200.

Pluginname	Laufzeit
atcassetsos	80 sec
atcassetsmsi	149 sec
atcassetshotfix	71 sec
atcassetssoftware	90 sec

Tabelle 7.6: Zeitaufwand der einzelnen auf *atc-assets* basierenden Plugins

Wie bereits bei der Überprüfung der Performance kann die funktionelle Überprüfung der *atc-assets* Plugins nicht auf einen Vergleich zwischen *atc-inventory* und der den Plugins zugrundeliegenden Applikation basieren. Für den Test der Funktionalität werden alle vier Plugins stichprobenartig auf Basis ausgewählter Systeme überprüft. Folgende drei PCs werden für die einzelnen Tests und für die Verifizierung der gesammelten Daten herangezogen:

- pc-352.akademie.bildendekunst.ac.at (10.3.3.33)
- pc-373.akademie.bildendekunst.ac.at (10.3.3.148)
- pc-400.akademie.bildendekunst.ac.at (10.3.3.149)

Als erstes der vier Plugins wird die Überprüfung des Betriebssystems getestet. Zum Zeitpunkt der Überprüfung sind in diesem Netzwerkbereich nur sieben Systeme aktiv. Für die Überprüfung der Funktionalität wird stichprobenartig getestet, ob die gesammelten Informationen stimmen. Die drei ausgewählten Systeme laufen unter Microsoft Windows XP Professional. Auf allen drei PCs ist das Service Pack 2 für Windows XP installiert. Folgendes Listing zeigt die von *atc-inventory* gesammelten Betriebssysteminformationen für die drei ausgewählten Systeme in Form der Ausgabe XML Datei.

```
1 <inventory>
2   <desc>scanned hosts by atc-inventory</desc>
3   <host>
```

```
4     <lastscan>Mon Oct 15 08:19:46 2007</lastscan>
5     <state>up</state>
6     <addresses>
7       <ipaddress>10.3.3.33</ipaddress>
8       <macaddress/>
9     </addresses>
10    <hostnames>
11      <hostname>pc-352.akademie.bildendekunst.ac.at</hostname>
12      <dnstype>PTR</dnstype>
13    </hostnames>
14    <os>
15      <osvendor>Microsoft</osvendor>
16      <osgen>Microsoft Windows XP Professional</osgen>
17      <servicepack>Service Pack 2</servicepack>
18    </os>
19  </host>
20  <host>
21    <lastscan>Mon Oct 15 08:18:45 2007</lastscan>
22    <state>up</state>
23    <addresses>
24      <ipaddress>10.3.3.148</ipaddress>
25      <macaddress>00:13:D3:50:C1:21</macaddress>
26    </addresses>
27    <hostnames>
28      <hostname>pc-373.akademie.bildendekunst.ac.at</hostname>
29      <dnstype>PTR</dnstype>
30    </hostnames>
31    <os>
32      <osvendor>Microsoft</osvendor>
33      <osgen>Microsoft Windows XP Professional</osgen>
34      <servicepack>Service Pack 2</servicepack>
35    </os>
36  </host>
37  <host>
38    <lastscan>Mon Oct 15 08:18:47 2007</lastscan>
39    <state>up</state>
40    <addresses>
41      <ipaddress>10.3.3.149</ipaddress>
42      <macaddress>00:13:20:B7:72:BA</macaddress>
43    </addresses>
44    <hostnames>
45      <hostname>pc-400.akademie.bildendekunst.ac.at</hostname>
46      <dnstype>PTR</dnstype>
47    </hostnames>
48    <os>
49      <osvendor>Microsoft</osvendor>
50      <osgen>Microsoft Windows XP Professional</osgen>
51      <servicepack>Service Pack 2</servicepack>
52    </os>
53  </host>
54 </inventory>
```

Listing 7.1: Überprüfung der durch das Plugin atcassets gesammelten Betriebssysteminformationen

Die mittels Stichproben durchgeführte Überprüfung der von *atc-inventory* gesammelten Daten zeigt, daß die installierten Betriebssysteme korrekt inventarisiert werden.
Der Test der drei `atcassets` Plugins für die Inventarisierung der installierten Software gestaltet sich komplizierter. Die Tabelle 7.7 zeigt die Anzahl der installierten Applikationen und Updates, die von den jeweiligen *atc-assets* Plugins inventarisiert wurden.

IP Adresse	laut atcassetsmsi installierte MSI Pakete	laut atcassetshotfix installierte Updates	laut atcassetssoftware installierte Software
10.3.3.33	34	124	197
10.3.3.148	34	138	204
10.3.3.149	13	86	143

Tabelle 7.7: Anzahl der gefundenen Softwareinstallationen auf Basis der *atc-assets* basierenden Plugins

Anhand der Anzahl sieht man nicht, ob die gesammelten Informationen vollständig und richtig sind. Auch wenn die Gesamtanzahl Rückschlüsse auf den Gesamtzustand des Systems zulässt, muß die installierte Software gesondert überprüft werden. Dies wird anhand von Stichproben durchgeführt. Hierfür wird pro untersuchtem System geprüft, ob *atc-inventory* ausgewählte installierte Applikationen findet und in die zentrale XML Datei ausgibt. Für diesen Test werden zwei Applikationen, die standardmäßig auf allen Verwaltungsrechnern in der Akademie der bildenden Künste Wien installiert werden, und zwei relativ aktuelle Microsoft Updates für Windows und für den Windows Media Player ausgewählt. Folgende Applikationen bzw. Microsoft Updates werden überprüft:

- Microsoft Office
- Adobe Acrobat
- MS07-022: Sicherheitsupdate für Windows (KB931784) [8]
- MS07-047: Sicherheitsupdate für Windows Media Player (KB936782) [9]

Die zwei Softwarepakete und die zwei Updates sind auf allen drei im Detail überprüften Rechnern installiert. Die beiden Microsoft Updates müssen durch `atcassetssoftware` und `atcassetshotfix` inventarisiert werden. Die beiden Softwareinstallationen müssen von den Plugins `atcassetsmsi` und `atcassetssoftware` gemeldet werden. Die Tabelle 7.8 zeigt das Ergebnis der Überprüfung:
Diese Überprüfung zeigt, daß aufgrund der stichprobenartigen Überprüfung alle *atc-assets* basierenden Plugins korrekte Daten liefern. Jedoch wird anhand dieses Tests auch klar,

IP Adresse	Software	atcassetsmsi	atcassetshotfix	atcassetssoftware
10.3.3.33	Microsoft Office	11.0.8173.0		11.0.8173.0
	Adobe Acrobat	7.0.9		7.0.9
	MS07-022		KB931784	KB931784
	MS07-047		KB936782	KB936782
10.3.3.148	Microsoft Office	11.0.8173.0		11.0.8173.0
	Adobe Acrobat	7.0.9		7.0.9
	MS07-022		KB931784	KB931784
	MS07-047		KB936782	KB936782
10.3.3.149	Microsoft Office	11.0.8173.0		11.0.8173.0
	Adobe Acrobat	8.0.0		8.0.0
	MS07-022		KB931784	KB931784
	MS07-047		KB936782	KB936782

Tabelle 7.8: Überprüfung der auf *atc-assets* basierenden Plugins mittels ausgewählter Softwareinstallationen

daß die meisten Plugins alleinstehend kein vollständiges Bild der installierten Applikationen und des installierten Betriebssystems eines überprüften Systems geben. Dementsprechend wichtig ist es, bei Überprüfungen verschiedene hostbasierende Plugins zu verwenden.

7.3.4 Test von *atc-inventory* mit mehreren aktiven Plugins

Als letzter Test wird eine komplette Überprüfung des DHCP Bereichs des Netzwerkbereichs 10.3.3.0/24 durchgeführt. Folgende Plugins sind aktiviert:

1. `atcnmapping`: Net-Plugin Priorität 3
2. `atcassetssw`: Host-Plugin Priorität 3
3. `atcassetsos`: Host-Plugin Priorität 2
4. `atcassetsmsi`: Host-Plugin Priorität 2
5. `atcassetshotfix`: Host-Plugin Priorität 2
6. `atcwsus`: Host-Plugin Priorität 1

Die Überprüfung wird mit dem Befehl

```
python atc-inventory.py -n 10.3.3.10-200 -X complete_10-3-3.xml
```

um 8:37:31 Uhr gestartet und läuft bis 8:47:09 Uhr. Zu dieser Zeit sind acht Systeme aktiv, wovon fünf Windows PCs sind und drei unter Mac OS X laufen. Die aktiven Plugins

werten das Betriebssystem und die installierte Software aus. Die Tabelle 7.9 listet die Ergebnisse dieser Überprüfung auf. Untersucht wurde

1. ob ein System aktiv ist
2. ob das Betriebssystem korrekt erkannt wird
3. die Anzahl der installierten Software (Applikationen und Updates)

IP Adresse	Betriebssystem	Betriebssystem erkannt	Anzahl Softwareinstallationen
10.3.3.33	Windows XP	Ja	220
10.3.3.49	Mac OS X	Nein	0
10.3.3.89	Mac OS X	Nein	0
10.3.3.117	Windows XP	Ja	138
10.3.3.136	Mac OS X	Nein	0
10.3.3.137	Windows XP	Ja	204
10.3.3.148	Windows XP	Ja	246
10.3.3.149	Windows XP	Ja	162

Tabelle 7.9: inventarisierte Systeme bei Verwendung mehrerer hostbasierender Plugins

Die Ergebnisse spiegeln den realen Zustand des DHCP Bereichs 10.3.3.10-10.3.3.200 zum Zeitpunkt der Überprüfung wider. Da *atc-inventory* die Daten der einzelnen Plugins aufgrund ihrer Priorität konsolidiert, ergibt eine komplette Überprüfung mit mehreren aktiven Plugins eine unterschiedliche Anzahl installierter Software als eine Überprüfung mit einem aktiven Plugin. Hier ist wieder anzumerken, daß die reine Anzahl der gemeldeten Softwareinstallationen keinen Rückschluß auf die Vollständigkeit und die Korrektheit der Daten zuläßt.
Für die Überprüfung der Korrektheit der Daten werden wiederum die vier bereits oben genannten Applikationen und Updates bei den drei ausgewählten PCs mit den IP Adressen 10.3.3.33, 10.3.3.148 und 10.3.3.149 überprüft. Zudem wird anhand der vier Installationen überprüft, ob der Prototyp die Daten korrekt konsolidiert und keine Mehrfachnennungen pro System vorkommen. Da jede dieser vier Softwareinstallationen von mindestens zwei Plugins inventarisiert wird, kann damit die Funktionalität der Überprüfung und der Konsolidierung doppelter Einträge getestet werden. Die zwei Softwarepakete Microsoft Office und Adobe Acrobat werden von den Plugins `atcassetsmsi` und `atcassetssoftware` und die Microsoft Updates MS07-022 und MS07-047 von den drei Plugins `atcassetssoftware`, `atcassetshotfix` und `atcwsus` inventarisiert. Die Ergebnisse dieser Analyse finden sich in Tabelle 7.10.
Das Ergebnis zeigt, daß die gesammelten Daten den realen Installationsstand der einzelnen Rechner darstellen. Wie man bei dem PC pc-352.akademie.bildendekunst.ac.at mit

IP Adresse	Softwarename	Version	Anzahl Einträge
10.3.3.33	MS Office Prof.	11.0.8173.0	1
	Adobe Acrobat 7 Prof.	7.0.9	2
	MS07-022	KB931784	1
	MS07-047	KB936782	1
10.3.3.148	MS Office Prof.	11.0.8173.0	1
	Adobe Acrobat 7 Prof.	7.0.9	1
	MS07-022	KB931784	1
	MS07-047	KB936782	1
10.3.3.149	MS Office Prof.	11.0.8173.0	1
	Adobe Acrobat 8 Standard	8.0.0	1
	MS07-022	KB931784	1
	MS07-047	KB936782	1

Tabelle 7.10: Überprüfung ausgewählter Softwareinstallationen bei der Verwendung mehrerer Plugins

der IP Adresse 10.3.3.33 jedoch sieht, funktioniert die Konsolidierung der Daten der einzelnen Plugins noch nicht zufriedenstellend.
Bei der Überprüfung des Rechners 10.3.3.33 mit den aktiven Host-Plugins `atcassetssoftware`, `atcassetsos`, `atcassetsmsi`, `atcassetshotfix` und `atcwsus` werden folgende zwei Softwareinstallationen in die *atc-inventory* XML Datei ausgegeben:

```
<software>
  <vendor>Adobe Systems</vendor>
  <product>
    Adobe Acrobat 7.0.9 Professional - English, Français, Deutsch
  </product>
  <version>7.0.9</version>
  <patchlevel/>
</software

<software>
  <vendor>Adobe Systems</vendor>
  <product>
    Adobe Acrobat 7.0 Professional - English, Français, Deutsch
  </product>
  <version>7.0.9</version>
  <patchlevel/>
</software>
```

Da die Konsolidierung des Prototypen auf einem eindeutigen Patchlevel (wie zum Beispiel bei Microsoft Updates) oder Produktnamen basiert, werden diese beiden Einträge von *atc-inventory* nicht als eine Applikation erkannt. Der erste Eintrag wird vom Plugin `atcassetssoftware` erstellt, der zweite vom Plugin `atcassetsmsi`. Alleinstehend stimmen die beiden Einträge. Zusammengefaßt stellt dies jedoch einen doppelten Eintrag dar. Dies zeigt, daß der derzeit implementierte Algorithmus nicht der Komplexität dieses Themas genügt.
Bei den zwei weiteren im Detail untersuchten PCs stimmen der Produktname im Uninstall-Registryschlüssel und der Produktname in der MSI Datenbank überein. Dies ist auf eine unterschiedliche Initialinstallation des Adobe Acrobat auf den einzelnen Rechnern der Akademie der bildenden Künste Wien zurückzuführen.
Zusammenfassend kann trotz dieser Probleme in der Konsolidierung der Daten gesagt werden, daß der Prototyp von *atc-inventory* ohne weiteres in Windows Netzwerken eingesetzt werden kann. Der Prototyp kann Windows Systeme in Netzwerken korrekt identifizieren und inventarisieren.

8 Erfahrungen und weiterführende Arbeit

Dieses Kapitel beschreibt die verschiedenen Erfahrungen und Erkenntnisse, die durch die Entwicklung von *atc-inventory* und durch die Testergebnisse gewonnen werden konnten. Zudem wird ein kurzer Ausblick auf die Weiterentwicklung des Inventarisierungswerkzeugs gegeben.

8.1 Allgemeine Erkenntnisse

Für die Implementierung von Security Managementlösungen muß der Gesamtblick auf die bestehende Infrastruktur gegeben sein. Das umfasst die Inventarisierung aller im Netzwerk aktiver Komponenten, das Abdecken einiger weniger Betriebssysteme ist nicht ausreichend. Da sich in Unternehmen immer stärker alternative Desktop Betriebssysteme wie Apple's Mac OS X oder Linux durchsetzen, müssen diese Betriebssysteme ebenfalls vom zentralen IT Security Management berücksichtigt werden. Neben diesen weitverbreiteten Betriebssystemen gibt es zudem eine immer größer werdende Anzahl von PDA's und Mobiltelefonen, die in die Unternehmensnetzwerke integriert werden und aktiv Netzwerkdienste verwenden. Dementsprechend müssen die einzelnen Sicherheitskonzepte Methoden für das Management dieser zusätzlichen Systeme unterstützen.
Eine vollständige Inventarisierung der Netzwerk- und Computerinfrastruktur hängt neben der verwendeten Technologie auch von der Konfiguration der einzelnen Geräte ab. Für den Zugriff auf die einzelnen aktiven Komponenten genügt nicht mehr nur das Vorhandensein eines administrativen Benutzers, vielmehr benötigt man die Möglichkeit, Konfigurationen bei Bedarf für den Zugriff anzupassen. Dies sieht man sehr gut am Beispiel von Microsoft Windows. Damit ein Microsoft Windows System zentral administriert werden kann, müssen Dienste wie die RemoteRegistry aktiviert und lokale Firewallregeln für die Integration in das Unternehmensnetzwerk angepasst sein. Zudem muß sichergestellt werden, daß die Benutzer die für das zentrale Management erforderlichen Konfigurationen nicht ändern und die dafür benötigten Dienste nicht deaktivieren. Ohne diese Voraussetzungen ist es nicht möglich, eine zentrale Verwaltung sicherzustellen. Dies gilt für alle aktiven

Komponenten und für alle Betriebssysteme.
Hier ist vor allem zu berücksichtigen, daß lokale Benutzer nicht die Möglichkeit besitzen, sicherheitskritische Konfigurationsänderungen durchzuführen. In zentral verwalteten Netzwerken hängt dies sehr stark von der eingesetzten Technologie ab. Sehr viele Unternehmen setzen kommerzielle Verzeichnisdienste wie das Microsoft Active Directory oder das Novell eDirectory ein. Ein Microsoft Windows System lässt sich sehr gut in solche Verzeichnisdienste integrieren. Daher ist es bereits Standard, daß die Benutzer von Microsoft Clients in Unternehmensnetzwerken nicht mit lokalen Administratorenkonten arbeiten, sondern daß die Benutzerkonten von den zentralen Verzeichnisdiensten bereitgestellt werden. Dementsprechend einfach ist es, den jeweiligen Benutzerkonten nur die benötigten Rechte zu geben, und ihnen nicht wie voreingestellt alle Rechte zur Verfügung zu stellen. Anders sieht es bei Apple und Unix basierenden Systemen aus. Auch wenn sich Mac OS X und Linux in bestehende Verzeichnisdienste durch die Unterstützung von Kerberos und LDAP integrieren lassen, wird dies noch nicht standardmäßig aktiviert. Daher verwenden Benutzer dieser Systeme sehr oft lokale Benutzerkonten mit administrativen Rechten.
Für das Sicherstellen, daß alle aktiven Systeme in die Inventarisierung mitaufgenommen sind, muß zudem eine Möglichkeit geschaffen werden, die einzelnen Systeme bei der Anmeldung in das Netzwerk zeitnahe zu überprüfen. Dies ist sehr wichtig, da nicht erkannte Geräte potentielle Sicherheitslücken darstellen, auf welche die Systembetreuer und die Administratoren nicht reagieren können. Bei dezentral verwalteten Netzwerken stellt dies ein Problem dar. Daher wird für die Problemlösung sehr oft auf agentenbasierte Systeme zurückgegriffen. Bei diesen Systemen benachrichtigt der Client den Server über seinen Status. Dies löst jedoch nicht das Problem von unbekannten Geräten, die keinen lokalen Agenten installiert haben.
Diese Erkenntnisse zeigen, daß ein effizientes Asset Management ein sehr komplexes Thema darstellt. Einige der Voraussetzungen wie ein funktionierender Verzeichnisdienst oder ein korrekt konfiguriertes DNS Service werden nicht nur für Sicherheitslösungen benötigt und sind daher in den meisten Fällen vorhanden. Die zeitnahe Inventarisierung von Systemen, die sich im Unternehmensnetzwerk anmelden, stellt das wesentlich größere Problem dar. Es gibt einige Ansätze und kommerzielle Implementierungen, die versuchen, dieses Problem zu lösen. Diese Lösungen setzen jedoch immer voraus, daß die Systeme bereits vorher bekannt sind, und daß ein lokaler Agent auf den Systemen läuft. Sowohl das Risikomanagement von ATCERT als auch das Sicherheitsmanagement des Netzwerks der Universität Akademie der bildenden Künste Wien setzen voraus, daß unbekannte Geräte ebenfalls erkannt werden. Daher muß zusätzlich zur Inventarisierungslösung eine Möglichkeit implementiert werden, die diese zeitnahe Inventarisierung ermöglicht.

8.2 Erfahrungen bei Konzeption, Implementierung und Test von *atc-inventory*

Ein Ziel dieser Diplomarbeit war die Implementierung eines funktionstüchtigen Prototypen für die Host- und Softwareinventarisierung. Für das Design des Prototypen wurden verschiedene existierende Projekte und Applikationen analysiert, und die daraus gewonnenen Erkentnisse sind in das Design miteingeflossen. Hier ist vor allem das Projekt NetMap hervorzuheben. Im Gegensatz zu den anderen in der Diplomarbeit vorgestellten Lösungsvorschlägen basiert NetMap nicht auf einen agentenbasierenden Ansatz, sondern versucht ähnlich wie das Modell von *atc-inventory* verschiedene spezialisierte Applikationen in die Funktionalität mitaufzunehmen. In diesem Punkt geht der Ansatz von *atc-inventory* über NetMap hinaus. Der in dieser Diplomarbeit beschriebene Lösungsansatz versucht nicht nur einzelne spezialisierte Applikationen für einzelne Teilbereiche zu verwenden, sondern integriert bereits im Unternehmensnetzwerk bestehende Management- und Sicherheitslösungen. Dies bringt vor allem Vorteile im Zeitaufwand für die Implementierung, da die zu integrierenden Applikationen nicht zusätzlich installiert werden müssen. Dementsprechend kosteneffizient ist *atc-inventory* (Abbildung 8.1). Anhand dieses Diagramms sieht man auch die wesentlich kostenintensivere Implementierung von agentenbasierenden Lösungen. Sowohl die beiden freien Systeme OVAL und Integrated Vulnerability Management (IVM), als auch die beiden kommerziellen Lösungen von Novell und CA basieren auf einem agentenbasierten Ansatz. Dies schlägt sich in den Implementierungskosten nieder, da die Agenten auf den lokalen Systemen installiert werden müssen.

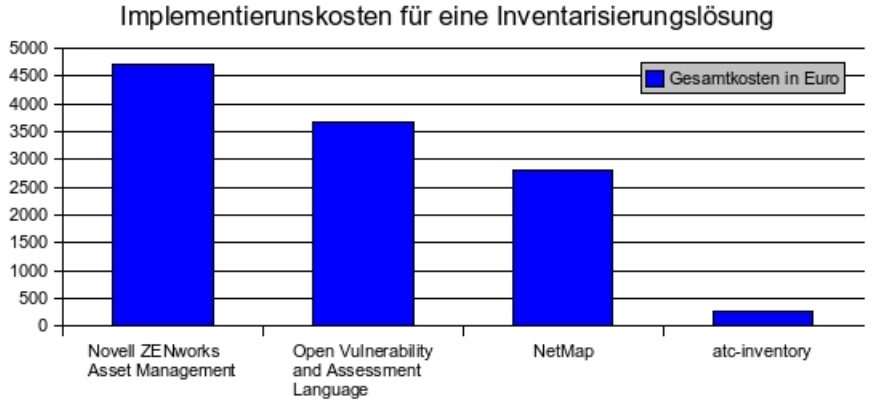

Abbildung 8.1: Kostenvergleich verschiedener Inventarisierungslösungen

Das Design von *atc-inventory* deckt einen Großteil der in der Fachliteratur beschriebenen Voraussetzungen für ein gesamtheitliches Security Management ab. Der entwickelte

Prototyp kann aufgrund der entwickelten Plugins IP fähige, aktive Netzwerkgeräte identifizieren. Derzeit können mit *atc-inventory* jedoch nur Windows Systeme inventarisiert werden. Dies hat sich während der Testphase als großes Manko herausgestellt, da damit kein Gesamtbild der eingesetzten IT erstellt werden kann (Tabelle 8.1).

Betriebssystem der überprüften Systeme	Anzahl der Systeme	von Host-Plugins inventarisiert	von Net-Plugins inventarisiert
Windows	9	9	9
Solaris	4	4	0
Linux	8	7	0
Appliance	6	6	0

Tabelle 8.1: Überprüfung eines Netzwerks mit proprietären Appliances, Windows-, Solaris- und Linux-Systemen

Für die gesamte Inventarisierung heterogener Netzwerke werden daher weitere Plugins benötigt. Für folgende Bereiche müssen zusätzliche Plugins entwickelt werden:

- für SNMP fähige Geräte wie Switches, Router und Drucker: ein generisches Plugin zum Sammeln von Daten über SNMP. Anhand dieser Schnittstelle kann mit angepassten SNMP MIBs jedes SNMP fähige Gerät überprüft werden.
- für Mac OS X Clients: ein Plugin zum Sammeln von Apple Client Informationen. Für diesen Zweck muß überprüft werden, ob Mac OS X eine geeignete Schnittstelle mitbringt, oder ob auf bestehende third-party Tools wie den Apple Remote Desktop zurückgegriffen wird.
- für Linux Systeme: ein Plugin zum Inventarisieren von Linux Systemen. Dies kann anhand verschiedener Technologien durchgeführt werden. Ob der Zugriff über SSH, SNMP oder anderen Schnittstellen durchgeführt wird, hängt vielfach vom zu überprüfenden Netzwerk ab. Dementsprechend werden mehrere Plugins benötigt, um Linux Systeme verschiedener Netzwerke effizient überprüfen zu können.

Zudem sollen weitere verbreitete Betriebssysteme wie zum Beispiel Solaris abgedeckt werden. Da der Zugriff auf Unix Systeme in den meisten Fällen dem Zugriff auf Linux Systeme ähnelt, kann für diesen Zweck in den meisten Fällen auf die für Linux entwickelten Plugins zurückgegriffen werden. Es ändern sich jedoch die Schnittstellen zu den Hard- und Softwareinformationen wie z.B. der Zugriff auf den verwendeten Paketmanager.
Als sehr effizient hat sich in der Testphase das Design und die Implementierung in Bezug auf die Laufzeit erwiesen. Hier liegen leider keine Vergleichsdaten zu den vorgestellten

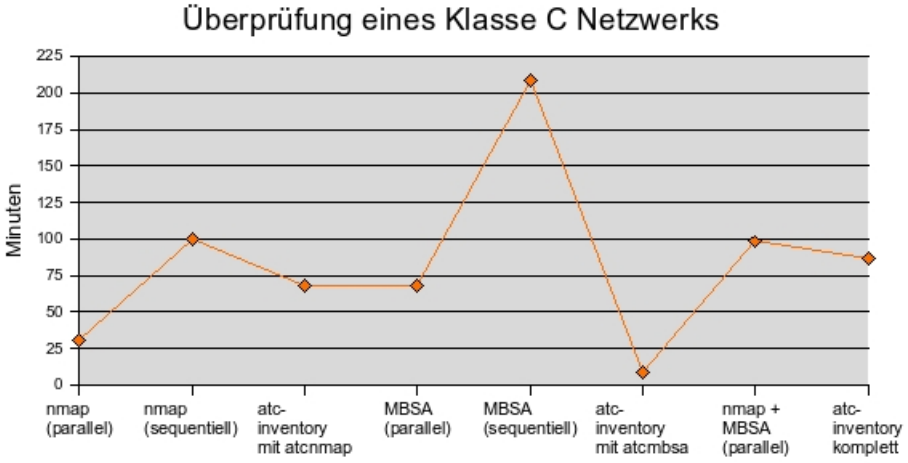

Abbildung 8.2: Zeitdauer für die Überprüfung eines Klasse C Netzwerks

alternativen Lösungen vor, jedoch kann der Prototyp sehr gut mit den orginalen, in *atc-inventory* integrierten Applikationen verglichen werden (Abbildung 8.2).
Der Zeitaufwand lässt keine Rückschlüsse auf die Qualität der gesammelten Daten zu. Dementsprechend wurde dies gesondert getestet. Auch dafür kann aufgrund der Testergebnisse eindeutig gesagt werden, daß der Prototyp den Anforderungen gerecht wird, und die gesammelten Daten dem realen Installationsstand entsprechen. Im Gegensatz zu den vorgestellten kommerziellen Lösungen benötigt *atc-inventory* für die korrekte Datensammlung keine eigene Datenbank bekannter Softwarelösungen für die Bestimmung der installierten Applikationen. Auch werden für die Überprüfung nicht wie bei OVAL und IVM permanent aktualisierte Konfigurations- und Definitionsdateien vorausgesetzt. Die Abbildung 8.3 zeigt die Effizienz der Integration verschiedener Applikationen sehr gut.
Anhand dieser Abbildung und Abbildung 8.2 sieht man auch, daß ein hintereinander durchgeführter Nmap und MBSA Scan eine längere Laufzeit als eine komplette Inventarisierung mit *atc-inventory* benötigt und trotzdem weniger Informationen als die in dieser Diplomarbeit vorgestellte Lösung sammelt.
Diese Testergebnisse zeigen zudem, daß für eine effiziente Inventarisierung von Systemen nicht unbedingt auf agentenbasierende Lösungen zurückgegriffen werden muß. So zeigt z.B. die Beispielimplementierung von *atc-assets* sehr gut, daß die von dem Betriebssystem zur Verfügung gestellten Schnittstellen für entfernte Abfragen die gleichen Informationen liefern wie ein lokaler Agent. Die Applikation *atc-assets* liefert im Hotfix Modus die selbe Liste der installierten Microsoft Windows Updates wie z.B. der Microsoft Baseline Security Analyzer. Einen Unterschied zwischen den beiden Applikationen gibt es sehr wohl. Die Lösung mit *atc-assets* liefert alle installierten Updates für das Betriebssystem, der Microsoft Baseline Security Analyzer konsolidiert Updates, die durch neuere

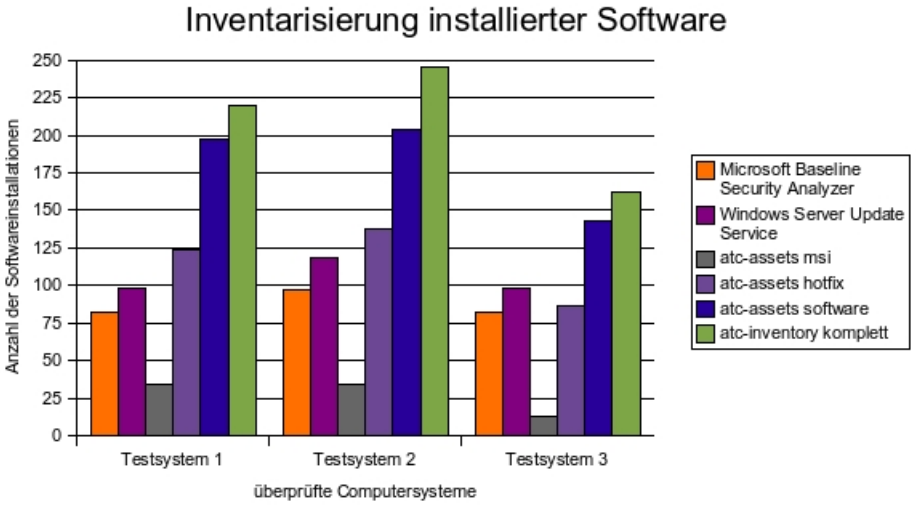

Abbildung 8.3: Anzahl der pro System inventarisierten Softwareinstallationen

Updates obsolet werden, und meldet diese nicht mehr.

Die verschiedenen Tests im Netzwerk der Akademie der bildenden Künste Wien lassen einige Rückschlüsse in Bezug auf die Skalierbarkeit und den Einsatz in größeren Netzwerken zu. Für den Einsatz in großen Netzwerken bietet ein agentenbasierender Ansatz Vorteile gegenüber dem agentenlosen. Bei den Tests des Prototypen hat sich gezeigt, daß der Einsatz von Firewalls und Paketfiltern die Ergebnisse signifikant verändern kann. Werden die für die jeweilige Überprüfung benötigten Ports und IP Adressen nicht auf den Firewalls und Routern zwischen den einzelnen Subnetzen freigeschaltet, kann nicht sichergestellt werden, daß die Qualität der Daten passt. Dieses Problem kann leider sehr schwer durch das Design einer Inventarisierungsapplikation gelöst werden, da für Konfigurationsänderungen auf Routern und Firewalls standardmäßig eigene Abläufe definiert sind. Bei agentenbasierenden Lösungen werden im Zuge der Installation der einzelnen Agenten auf den Unternehmensrechnern und eines abschließenden Tests in der Regel die erforderlichen Konfigurationsänderungen durchgeführt. Überprüft man zentral die Unternehmensinfrastruktur, ist das selten wie ein Rollout-Projekt geplant. Daher werden diese Änderungen eher vergessen, und die gesammelten Daten spiegeln nicht den realen Zustand wider. Werden die Filterlisten für die Anforderungen der Überprüfungssoftware geändert, kann aufgrund der Testergebnisse angenommen werden, daß sich der Aufwand für das Überprüfen mehrerer Subnetze linear zu den bestehenden Tests entwickelt.

Aufgrund dieser Erfahrungen kristallisieren sich folgende Eckpfeiler in Bezug auf eine Netzwerk- und Hostinventarisierung heraus:

- Eine Inventarisierung der Unternehmensinfrastruktur ist essentiell für die Imple-

mentierung von Sicherheits- und Netzwerkmanagementlösungen wie ein Update-, ein Risiko- oder ein Lizenzmanagement.

- Die Implementierung einer Inventarisierung rechnet sich auch für Klein- und Mittelunternehmen, da kostengünstige und an die Unternehmen angepasste Lösungen existieren. Zudem bringt die Einführung von Securitylösungen wie die eines Asset Management einen erheblichen Nutzen für Klein- und Mittelunternehmen im Bereich des IT Managements.

- Es genügt nicht, einzelne Geräte bzw. einzelne Betriebssysteme durch die Sicherheitslösung abzudecken. Um einen Gesamtüberblick über die Infrastruktur zu haben, müssen alle im Netzwerk aktiven Systeme berücksichtigt werden.

- Die Implementierung und die Einführung einer gesamtheitlichen Inventarisierungslösung ist ein komplexes Thema und hängt von etlichen Faktoren wie einem funktionierenden Verzeichnisdienst und einem korrekt konfigurierten DNS Server ab. Zudem muß das Asset Management in das bestehende Netzwerkdesign integriert werden, und Konfigurationen wie z.B. die Filterlisten von Firewalls müssen für den Einsatz angepasst werden.

- Eine effiziente IT Asset Management Lösung kann sowohl agentenbasiert als auch agentenlos implementiert werden. Beide Paradigmen bieten Möglichkeiten, alle benötigten Informationen zu sammeln. Welcher dieser beiden Ansätze effizienter ist, hängt einerseits von den Anforderungen an die Lösung ab, andererseits jedoch auch vom Aufbau der Unternehmensstruktur.

- Der modulare Ansatz von *atc-inventory* erweist sich als sehr effizient. Zudem ermöglicht ein solcher modularer Aufbau der Inventarisierung den Einsatz von agentenbasierenden und agentenlosen Technologien.

8.3 Ausblick und weiterführende Arbeit

Neben einzelnen Punkten wie das Abdecken verschiedener Betriebssysteme, mit denen die vorgestellte Lösung *atc-inventory* erweitert werden muß, sind durch den Test des Prototypen auch einige Bereiche aufgedeckt worden, die unbedingt verbessert werden müssen. Ein solcher Teil von *atc-inventory* ist die Datenkonsolidierung mehrfach gesammelter Informationen. Der Prototyp von *atc-inventory* hat die Funktionalität aufgrund der Priorisierung der Plugins bereits vorbereitet, der derzeit implementierte Algorithmus zum Konsolidieren von Mehrfacheintragungen ist jedoch nicht ausreichend. Dieser Algorithmus

ist für eine vollständige und fehlerlose Inventarisierung zu verbessern. Das Vergleichen von Produktnamen von Softwareinstallationen bei fehlenden eindeutigen Patchleveln oder Produktnamen ist nicht effizient. Daher muß hier ein Algorithmus entwickelt werden, der es ermöglicht, die von mehreren Plugins gesammelten Detailinformationen zu konsolidieren und auf einen Eintrag in der zentralen XML Datei zu reduzieren. Für die Entwicklung dieses Algorithmus ist zu prüfen, ob es eine effiziente Lösungsmöglichkeit ohne einer zugrundeliegenden Datenbank, die alle bekannten Applikationen beinhaltet, gibt.
Zudem ist der Prototyp derzeit noch nicht Unicode fähig. Das Analysieren von Systemen, die als Standardzeichensatz UTF-8 verwenden, kann zu Problemen und Fehlern führen. Dieser Punkt ist auf jeden Fall zu berücksichtigen, da sich UTF-8 verstärkt auch als Standardzeichensatz auf Desktop Systemen durchsetzt.
Ein weiterer Bereich, der derzeit weder vom Prototyp noch vom Design abgedeckt wird, ist die Inventarisierung der Hardwaredaten. Für ein gesamtheitliches Sicherheitsmanagement sind auch diese Daten von Bedeutung. Dementsprechend muß das XML Schema um die benötigten Elemente und Attribute erweitert werden. Zudem müssen Plugins entwickelt werden, die diese Daten sammeln. Hierfür kann vielfach wiederum auf die selben Schnittstellen wie für die Inventarisierung der installierten Applikationen zurückgegriffen werden. Bei Microsoft Windows Systemen kann zum Beispiel anhand von WMI die Hardware ausgelesen werden.

Nach der Lösung der offenen Fragen wie der Datenkonsolidierung und der Erweiterung der Funktionalität in Hinblick auf andere Betriebssysteme kann *atc-inventory* als Basiswerkzeug für verschiedene Managementlösungen eingesetzt werden. So ist geplant, *atc-inventory* als Sensor bei ATCERT einzusetzen. In der Akademie der bildenden Künste soll *atc-inventory* zuerst als Inventarisierungsprogramm für die eigenentwickelte Einsatz- und Supportdatenbank verwendet werden. Alle netzwerk- und hostbasierenden Informationen sollen durch *atc-inventory* gesammelt und in der zentralen Oracle Datenbank automatisiert gespeichert werden. In Folge soll die zentrale Datenbankapplikation mit den von *atc-inventory* gesammelten Informationen für das Lizenzmanagement der Akademie der bildenden Künste erweitert werden.
Weiters ist geplant, auf Basis des Inventarisierungsprogramms ein zentrales Update- und Risikomanagement für alle vier eingesetzten Betriebssysteme zu implementieren. Diese Lösung soll in Zukunft auch die Konfiguration und die Überwachung des verwendeten Intrusion Detection Systems beinhalten.

9 Zusammenfassung

Ziel dieser Arbeit war es, den Bedarf und die Implementierung einer effizienten Netzwerk- und Hostinventarisierung für das IT Sicherheitsmanagement am Fallbeispiel von *atc-inventory* zu zeigen und ein Inventarisierungstool zum Sammeln hostbasierender Daten zu entwickeln.

Netzwerk- und Hostinventarisierung - die Basis für IT Sicherheitslösungen

Das Thema der IT Sicherheit ist heute zentraler Bestandteil jedes größeren Netzwerks. Und das effiziente Management von sicherheitsrelevanten IT basierenden Aufgaben kann ohne die Bestandsaufnahme aller aktiven Geräte nicht durchgeführt werden. Daher ist es unumgänglich, eine Netzwerk- und Softwareinventarisierung einzuführen, auf welche weitere Security Lösungen wie z.B. das Update- oder das Risikomanagement aufsetzen. Solche zentrale Sicherheitslösungen sind meist für Klein- und Mittelbetriebe schwierig zu implementieren. Sehr oft fehlen ihnen das fachliche Know-How und die finanziellen Mittel für den Aufbau dieses Wissens. Mit diesem Problem beschäftigt sich unter anderem Secure Business Austria, ein Kompetenzzentrum mit verschiedenen akademischen und industriellen Partnern. Secure Business Austria entwickelt ein auf eine Security Ontologie basierendes semantisches CSIRT, welches die Anforderungen und die Bedürfnisse von Klein- und Mittelbetrieben berücksichtigt. Dieses Projekt ATCERT hat zum Ziel, ein effizientes und leistbares Risikomanagement zur Verfügung zu stellen, mit welchem Klein- und Mittelbetriebe auf die immer größer werdende Anzahl von Sicherheitsproblemen und Softwarelücken reagieren können. Zur Sicherstellung dieser Funktionalität muß die Infrastruktur des Unternehmens bekannt sein. Hierfür werden Sensoren benötigt, die diese Informationen sammeln, und aufbereitet in der dem CSIRT zugrundeliegenden Security Ontologie speichern. Im Zuge der Diplomarbeit wurde ein solcher Sensor entwickelt.

Kommerzielle und akademische Lösungsansätze

Da verschiedene Standards wie ITIL und ISO 17799 ein Asset Management voraussetzen, gibt es bereits verschiedene kommerzielle Lösung wie das CA Unicenter Asset Management oder das Novell ZENworks Asset Management. Diese kommerziellen Lösungen

sind in der Regel proprietär und decken nur einen bestimmten Bereich ab. Zudem sind sie nicht erweiterbar, und die durch diese Systeme gesammelten Daten sind nicht für andere Anwendungsgebiete verwendbar.
Neben diesen kommerziellen Produkten sind in dieser Arbeit auch einige offene und freie Lösungsansätze für die Inventarisierung beschrieben. Eine offene, industrienahe Lösung ist die Open Vulnerability and Assessment Language, die anhand von XML Schemas versucht, den Austausch sicherheitsrelevanter Informationen zu vereinheitlichen. Zum Sammeln der Informationen steht als Referenzimplementierung der OVAL Interpreter, eine lokale Agentensoftware, zur Verfügung.
Das aus dem akademischen Umfeld stammende Integrated Vulnerability Management System setzt auf diesen Standard auf, erweitert die Lösung jedoch dahingehend, daß zudem verschiedene Datenbanken und Informationsquellen für die Datenauswertungen verwendet werden können.
Eine weitere akademische Lösung ist NetMap, das verschiedene spezialisierte Applikationen für die Datensammlung in die Funktionalität integriert. Dieses modulare Design bringt Vorteile, da die meisten der am Markt verfügbaren Applikationen auf spezielle Aufgaben limitiert sind. Im Gegensatz zu der in dieser Diplomarbeit entwickelten Lösung liegt der Hauptfokus dieses Systems in der Inventarisierung der Netzwerktopologie und der angebotenen Netzwerkdienste, das Sammeln von hostbasierenden Daten wird nicht berücksichtigt.

Kosten und Nutzen von Sicherheitslösungen bei Klein- und Mittelbetrieben

In der Regel stellt die IT Security keinen direkten Nutzen für die Unternehmensziele dar. Dementsprechend wird bei Klein- und Mittelbetriebe wenig in diesen Bereich investiert. Die Verknüpfung zwischen den Unternehmenszielen bzw. den Kernunternehmensprozessen und der IT ist jedoch sehr wichtig. Meist ist Klein- und Mittelbetrieben gar nicht bewußt, wie stark ihre Kernprozesse von der EDV abhängen. IT Sicherheitsprobleme können daher den Unternehmensgewinn und -umsatz stark einschränken. Zudem können sie zu großen indirekten Kosten wie einem Reputationsverlust oder zum Verlust von Kunden führen. Daher ist es für Klein- und Mittelbetriebe wichtig, anhand von Kosten-Nutzen-Analysen den Wert von IT Sicherheit zu beziffern. Um den größtmöglichen Nutzen zu bringen, müssen Security Lösungen an die Anforderungen von Klein- und Mittelbetriebe angepasst werden. Aufgrund der begrenzten Investitionsbudgets dieser Unternehmen müssen die Lösungen auf diese Betriebe abgestimmt und mit einem Fixpreis implementierbar sein. Zudem müssen die laufenden Kosten überschaubar bleiben. Ohne Berücksichtigung dieser speziellen Anforderungen, ist die Einführung von IT Securitylö-

sungen bei Klein- und Mittelbetrieben mit zu hohen Kosten verbunden, und Investitionen werden daher nicht getätigt. Die Sicherheit bleibt unberücksichtigt.
Anhand einiger Fallbeispiele zeigt diese Arbeit den effektiven Nutzen einer Netzwerk- und Hostinventarisierung und von darauf basierenden Sicherheitslösungen wie z.B. dem Patchmanagement. Weiters werden in den Fallbeispielen die Kosten der einzelnen Lösungen dem Nutzen gegenübergestellt und die im Zuge dieser Diplomarbeit entwickelte Lösung *atc-inventory* mit den beschriebenen alternativen Lösungen verglichen.

Design und Implementierung von *atc-inventory*

Aufgrund der analysierten bestehenden Lösungsvorschläge und der Security Ontologie von ATCERT wurde das Inventarisierungswerkzeug *atc-inventory* entwickelt. Die Idee hinter *atc-inventory* ist ähnlich dem Modell von NetMap, berücksichtigt jedoch sehr viel stärker die Anforderungen von Klein- und Mittelbetrieben. Wie NetMap integriert *atc-inventory* externe third-party Applikationen in die Funktionalität. Diese als Plugins integrierten Programme sammeln die erforderlichen Daten. In diesem Schritt geht *atc-inventory* jedoch weiter als die vorgestellte Lösung NetMap. In den meisten Unternehmensnetzwerken gibt es bereits installierte Managementprodukte für die diversen Bereiche. So sind in vielen microsoftlastigen Netzwerken Windows Update Server implementiert, anhand derer die vorhandenen Microsoft Windows Systeme aktualisiert werden. *atc-inventory* versucht diese bereits bestehenden Security- und Managementsysteme anhand spezifizierter Schnittstellen zu integrieren. Dies bringt eine sehr große Aufwandsersparnis mit sich, da die für die Inventarisierung benötigten Programme bereits im Netzwerk implementiert und aktiv sind. Zusätzliche Applikationen werden nur implementiert, wenn eine benötigte Funktionalität nicht von bereits existierenden Lösungen bereitgestellt werden kann. Zudem hat *atc-inventory* den Anspruch, die gesammelten Informationen anderen Programmen für zusätzliche Arbeits- und Verwaltungsabläufe zur Verfügung zu stellen. Daher werden die gesammelten Daten in einem XML Format gespeichert. Das definierte XML Schema ist mit Standardmitteln weiterverarbeitbar und kann jederzeit erweitert werden. Werden neue Funktionalitäten benötigt oder durch zusätzliche Plugins und Applikationen bereitgestellt, kann das Schema problemlos erweitert werden. Die bestehenden Plugins aktualisieren nur die für sie relevanten Daten in der zentralen XML Datei. Durch den modularen Aufbau und die Verwendung unterschiedlicher third-party Applikationen können einige Daten durch mehrere Plugins gesammelt werden. Zur Konsolidierung dieser Mehrfachnennungen von Informationen können die einzelnen Plugins priorisiert werden, die Daten werden nur einmal im XML Schema abgebildet.
Die Design von *atc-inventory* ist offen und modular gehalten. Ziel ist es, das Programm so weit wie möglich plattformunabhängig zu gestalten. Leider werden teilweise betriebssys-

temspezifische Bibliotheken oder proprietäre Programme benötigt. Daher funktionieren gewisse Plugins nur unter den von der eingesetzten Technologie unterstützten Betriebssystemen.

Test von *atc-inventory*

Zum Test der Funktionalität wurde der *atc-inventory* Prototyp entwickelt. Der Prototyp deckt derzeit nur Microsoft Windows Systeme ab. Der Grund hierfür liegt darin, daß der Großteil der Klein- und Mittelbetriebe hauptsächlich Microsoft Windows auf den Unternehmenscomputern einsetzt. Der in Python entwickelte Prototyp wurde im Universitätsnetzwerk der Akademie der bildenden Künste Wien getestet. Die Testergebnisse zeigen, daß der gewählte Ansatz sowohl von der Performance als auch von der Funktionalität die Anforderungen von Klein- und Mittelbetrieben abdeckt. Der Prototyp kann demnach problemlos für die Inventarisierung von Microsoft Windows Systemen eingesetzt werden. Jedoch zeigt der Test auch, daß einige Teile des Prototypen verbessert werden müssen. So ist z.B. der Algorithmus für die Datenkonsolidierung nicht effizient genug, es kann zu Mehrfachnennungen von installierter Software kommen. Hier gibt es sicher noch ein Verbesserungspotential. Auch zeigt sich, daß das Abdecken von Microsoft Windows nicht für ein gesamtheitliches Sicherheitsmanagement genügt. Dementsprechend müssen Plugins für zusätzliche Betriebssysteme wie Mac OS X und Linux entwickelt werden. Damit auch Router und Switches mit abgedeckt werden können, sollte zudem ein generisches SNMP Plugin implementiert werden.

Vorteile von *atc-inventory*

Abschließend kann gesagt werden, daß *atc-inventory* einen effektiven Nutzen für Klein- und Mittelbetriebe bringt. Durch das modulare und offene Design werden die Anforderungen solcher Unternehmen abgedeckt und es hat gegenüber den vorgestellten alternativen Lösungen die Vorteile

- der kostengünstigen Implementierungsmöglichkeit,
- der einfachen Integration in bestehende Netzwerke,
- der Effektivität der Informationssammlung und
- der problemlosen Weiterverarbeitbarkeit der gesammelten Daten.

Für die Verwendung in heterogenen Netzwerken muß der Prototyp jedoch erweitert werden. Auch wenn das Design die Inventarisierung heterogener Netzwerke abdeckt, müssen für diese Funktionalität zusätzliche Applikationen in Form von Plugin integriert werden.

Damit kann das Inventarisierungswerkzeug *atc-inventory* effizient als Basis für weitere IT Sicherheitslösungen wie z.B. ein Update-, ein Lizenz- oder ein Risikomanagement eingesetzt werden.

Literaturverzeichnis

[ALRL04] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 01(1):11–33, 2004.

[And01] R. Anderson. Why information security is hard - an economic perspective. In *Computer Security Applications Conference. ACSAC 2001. Proceedings 17th Annual*, pages 358–365, Dec 2001.

[Bea04] J. Beale. *Snort 2.1 Intrusion Detection*. Syngress, second edition, May 2004.

[Bed99] John B. Bedunah. Xml: the future of the web. *Crossroads*, 6(2):5–10, 1999.

[BGJ+04] Y. Breitbart, M. Garofalakis, B. Jai, C. Martin, R. Rastogi, and A. Silberschatz. Topology discovery in heterogeneous ip networks: the netinventory system. *IEEE/ACM Trans. Netw.*, 12(3):401 – 414, 2004.

[BGSS06] M. Brenner, M. Garschhammer, M. Sailer, and T. Schaaf. Cmdb - yet another mib? on reusing management model concepts in itil configuration mangement. In *Large Scale Management of Distributed Systems (Proceedings of DSOM 2006, 17th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management)*, volume 4269/2006, pages 698 – 703. Springer, October 2006.

[BS03] B. Brykczynski and R. A. Small. Reducing internet-based intrusions: Effective security patch management. *Software (IEEE)*, 20(1):50 – 57, January – February 2003.

[CFSD90] J. D. Case, M. Fedor, M. L. Schoffstall, and J. Davin. Simple network management protocol (snmp). Request for Comments 1157, 1990.

[CWN02] R. Colville, R. Wagner, and M. Nicolett. Research note: Patch management benefits, challenges and prerequisites. *Decision Framework, DF-18-0680*, November 2002.

[DOFC00] M. Davis, W. O'Donovan, J. Fritz, and C. Childress. Linux and open source in the academic enterprise. In *SIGUCCS '00: Proceedings of the 28th annual ACM SIGUCCS conference on User services*, pages 65–69, New York, NY, USA, 2000. ACM Press.

[Don03] M. Donner. Toward a security ontology. *IEEE Security and Privacy*, 01(3):6–7, 2003.

[DRM+04] R. Deraison, N. Rathaus, HD Moore, R. Alder, G. Theall, A. Johnston, and J. Alderson. *Nessus Network Auditing*. Syngress, first edition, July 2004.

[EFK+06] A. Ekelhart, S. Fenz, M. Klemen, A Min Tjoa, and E. R. Weippl. Ontology-based business knowledge for simulating threats to corporate assets. In Ulrich Reimer and Dimitris Karagiannis, editors, *Practical Aspects of Knowledge Management (PAKM06)*, volume 4333 of *Lecture Notes in Artificial Intelligence*, pages 37–48, Vienna, Austria, Dec 2006. Springer.

[EFKW07] A. Ekelhart, S. Fenz, M. Klemen, and E. R. Weippl. Security ontologies: Improving quantitative risk analysis. In *40th Hawaii International Conference on System Sciences (HICSS07)*, pages 156–162, Los Alamitos, CA, USA, Jan 2007. IEEE Computer Society.

[GM04] T. Gerace and J. Mouton. The challenges and successes of implementing an enterprise patch management solution. In *SIGUCCS '04: Proceedings of the 32nd annual ACM SIGUCCS conference on User services*, pages 30 – 33, New York, NY, USA, 2004. ACM Press.

[GPFLC04] A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering*. Springer, first edition, 2004.

[JB04] D. Just and M. Bereszewski. It-governance für mehr effizienz. *Information Week*, May 2004.

[KRV04] C. Kruegel, W. Robertson, and G. Vigna. Using alert verification to identify successful intrusion attempts. *Practice in Information Processing and Communication (PIK)*, 27(4):219 – 227, October – December 2004.

[KV02] R. A. Kemmerer and G. Vigna. Intrusion detection: a brief history and overview. *Computer (IEEE)*, 35(4):27 – 30, Apr 2002.

[Köh06] P. T. Köhler. *ITIL. Das IT-Servicemanagement Framework*. Springer, second edition, October 2006.

[Lea99] A. C. Lear. Xml seen as integral to application integration. *IT Professional*, 1(5):12 – 16, Sep – Oct 1999.

[MBH05] P. Mell, T. Bergeron, and D. Henning. Recommendations of the national institute of standards and technology, creating a patch and vulnerability management program. *NIST Special Publications 800-40*, Nov 2005.

[Mer03] Rebecca T. Mercuri. Analyzing security costs. *Commun. ACM*, 46(6):15–18, 2003.

[MPS+03] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the slammer worm. *Security & Privacy Magazine (IEEE)*, 1(4):33 – 39, July – August 2003.

[NKB05] Thomas Neubauer, Markus Klemen, and Stefan Biffl. Business process-based valuation of it-security. In *EDSER '05: Proceedings of the seventh international workshop on Economics-driven software engineering research*, pages 1–5, New York, NY, USA, 2005. ACM Press.

[oST79] National Institute of Standards and Technology. Guidelines for automatic data processing risk analysis. *FIPS PUB 65*, Aug 1979.

[SR01] L. Seligman and A. Roenthal. Xml's impact on databases and data sharing. *Computer (IEEE)*, 34(6):59 – 67, Jun 2001.

[TFHC03] C. K. S. Tong, K. H. Fung, H. Y. H. Huang, and K. K. Chan. Implementation of iso17799 and bs7799 in picture archiving and communication system: local experience in implementation of bs7799 standard. In *CARS 2003. Computer Assisted Radiology and Surgery. Proceedings of the 17th International Congress and Exhibition*, volume 1256, pages 311 – 318. International Congress Series, June 2003.

[TG88] Sun Tzu and S. B. Griffith. *The Art of War (UNESCO Collection of Representative Works: European)*. Oxford University Press, 1988.

[Tho80] Mark S. Thompson. *Benefit-cost analysis for program evaluation*. Sage, first edition, 1980.

[VVZK02] G. Vigna, F. Valeur, J. Zhou, and R.A. Kemmerer. Composable tools for network discovery and security analysis. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC '02)*, pages 14 – 24, Las Vegas, NV, December 2002. IEEE Press.

[WYYR05] W. Wu, F. Yip, E. Yiu, and P. Ray. Integrated vulnerability management system for enterprise networks. In *e-Technology, e-Commerce and e-Service, 2005, EEE'05. Proceedings. The 2005 IEEE International Conference on*, pages 698 – 703. IEEE Press, March 2005.

Internetquellen

[1] CA. Unicenter asset management r11. Webseite, letzter Zugriff 30.05.2007. http://www.ca.com/files/DataSheets/unicenter_asset_mgmt_r11_ds.pdf.

[2] Inc. (DMTF) Distributed Management Task Force. Web services for management (ws-management). Webseite, letzter Zugriff 08.10.2007. http://www.dmtf.org/standards/published_documents/DSP0226.pdf.

[3] Fyodor. Remote os detection using tcp/ip fingerprinting (2nd generation). Webseite, letzter Zugriff: 08.10.2007. http://insecure.org/nmap/osdetect/index.html.

[4] H. E. Hauser. Smes in germany, facts and figures 2000. Webseite, letzter Zugriff: 08.10.2007. http://www.ifm-bonn.org/ergebnis/sme-in-germany.pdf.

[5] Microsoft. Ms02-039: Buffer overruns in sql server 2000 resolution service could enable code execution (q323875). Webseite, letzter Zugriff: 08.10.2007. http://www.microsoft.com/technet/security/bulletin/ms02-039.mspx.

[6] Microsoft. Ms03-027: An unchecked buffer in the windows shell could permit your system to be compromised. Webseite, letzter Zugriff: 08.10.2007. http://support.microsoft.com/kb/821557/en-us.

[7] Microsoft. Platform sdk: Windows server update services, microsoft.updateservices.administration namespace. Webseite, letzter Zugriff: 08.10.2007. http://msdn2.microsoft.com/en-us/library/ms825115.aspx.

[8] Microsoft. Ms07-022: Vulnerability in windows kernel could allow elevation of privilege (931784). Webseite, letzter Zugriff: 20.10.2007. http://www.microsoft.com/technet/security/bulletin/ms07-022.mspx.

[9] Microsoft. Ms07-047: Vulnerabilities in windows media player could allow remote code execution (936782). Webseite, letzter Zugriff: 20.10.2007. http://www.microsoft.com/technet/security/bulletin/ms07-047.mspx.

[10] Mitre. An introduction to the oval language version 5.0. Webseite, letzter Zugriff: 08.10.2007. http://oval.mitre.org/oval/documents/docs-06/an_introduction_to_the_oval_language.pdf.

[11] Novell. Get control of your software assets. Webseite, letzter Zugriff: 08.10.2007. http://www.novell.com/collateral/4622003/4622003.pdf.

[12] CMP Research. The state of it asset managment (itam) in north america. Webseite, letzter Zugriff 08.10.2007. http://www.ca.com/us/whitepapers/collateral.aspx?cid=92450.

[13] SANS. The ten most important security trends of the coming year, letzter Zugriff: 08.10.2007. http://www.sans.org/resources/10_security_trends.pdf.

[14] J. Shenk and D. Shackleford. Sourcefire real-time network awareness. Webseite, letzter Zugriff: 08.10.2007. https://www.sourcefire.com/resources/#cs.

[15] Unisys. Advancing open source for the enterprise. Webseite, letzter Zugriff 08.10.2007. http://www.unisys.com/eprise/main/admin/corporate/doc/Advancing_Open_Source_for_the_Enterprise_Positioning_Brief.pdf.

[16] M. Wild and S. Herges. Total cost of ownership (tco): ein Überblick. *Arbeitspapiere WI*, letzter Zugriff: 08.10.2007. http://geb.uni-giessen.de/geb/volltexte/2004/1577/.

Abbildungsverzeichnis

Tabellenverzeichnis

Listings