Zusammenfassung - NASS

SK

20. Januar 2016

> Korrektheit und Vollständigkeit der Informationen wird nicht gewährleistet.

Inhaltsverzeichnis

1 Introduction

1.1 Taxonomie der Angreifer

- einzelner Angreifer
  - sozialer Hintergrund
  - öffentliche Aufmerksamkeit als Antrieb
  - evtl pol. Statements
  - geht gewöhnlich niedrige Risiken ein
- organisierte Kriminalität
  - Geld als Antrieb
  - mittlere Risiken
- Terroristen
  - politische oder gesellschaftliche Motivation
  - hohe Risiken

  - Zerstörung/Verwirrung als Ziel

- Konkurrenten
  - möglichst niedriges Risiko der Aufdeckung(abhängig vom wert der Information)
  - Informationsdiebstahl oder Zerstörung als Ziel

- Regierungsorganisationen
  - Industriespionage zum Wohl einheimischer Firmen
  - Militärspionage udn hybride Kriegsführung

1.2 Angriffe gegen einen Computer

Informationsdiebstahl führt zu:

- Wettbewerbsvorteilen
- Verwirrung
- Erpressung

Zerstörung führt zu:

- Spaß und Selbstverherrlichung
- Politischen Stellungnahmen

Sammlung von Informationen

- Infos werden zu Angreifer gesendet
- an Netzwerk angeschlossene Rechner mit höherem Risiko
- Zugriff für Angreifer durch:
  - Social engineering
  - Viren/Trojaner/Würmer
  - Physischer Diebstahl von Datenträgern
  - Sniffing

Zerstörung von Infos

- Infos gehen verloren
- physische Angriffe/Feuer/Naturkatastrophen
- Beabsichtigte Löschungen durch
  - Social Engineering
  - Viren/Trojaner/Würmer

Viren

- Infektion von Dateien
- Infektion von System und Boot record
- Zerstörung, Verwirrung und öffentliche Aufmerksamkeit als Ziel

Würmer

- Mailing Worms - Verbreitung durch E-Mails
- Viren/Trojaner evtl als „Nutzlast“
- Network worms - Verbreitung durch Ausnutzung von Softwaremängeln(bspw Bufferoverflows)
- Ablauf:

  - Zielauswahl
  - ausnutzen(exploit)
  - Infektion
  - Verbreitung

Backdoors und Trojaner

- Schadsoftware wird in nützlicher Software versteckt
- mögliche Funktionen:
  - mitschneiden von Daten(logging)
  - Zerstörung
  - Installation weiterer Software(DoS Clients, root kits etc)
  - bedingter Start von Prozessen (time bombs)

Identitäts Spoofing

- Angreifer übernimmt die Identität von jemand anderem
- Angreifer und Ziel müssne normalerweise ein Netzsegment teilen
- Angreifer liefert evtl falsche Infos über Routen oder Namen
- Grundsätzlich sind alle Antworten eines Protokolls potentielle Spoofingsubjekte(subject of spoofing?)

DoS

- Angreifer möchte einen Dienst der von einem Rechner oder Gerät angeboten wird überladen
- Angriffe gegen Konkurrenten, als pol/gesellschaftliche Aussage oder um andere Aktivitäten zu verbergen
- bösartige Anfragen sind nicht von normalen Anfragen zu unterscheiden
- BSP: HTTP, DNS DoS, SYN Flooding

Bot Network

- Fernsteuerung mehrerer Rechner um bösartige Aktionen auszuführen
- bsp: DDoS, aufwändige Entschlüsselungen berechnen

Password/Schlüssel Attacken

- Brute Force
- Raten/ Wörterbuchangriffe
- Mängel in der Implementation(z.B. Password als Klartext gespeichert)

Port/Network Scanning

- während der Aufklärungsphase um Sicherheitslücken und geeignete Ziele zu finden
- Angreifer möchte Informationen über das System erlangen
- Sniffing/Mapping/Port Scans

Session Highjacking

- Angreifer bricht in einen bestehender Session ein ohne sich einloggen zu müssen.

ZSF: viele unterschiedliche Angriffsmöglichkeiten $\Rightarrow$ unüberschaubare Anzahl an verschiedenen Attacken Angreifer unterscheiden sich in Motivation und Möglichkeiten.

2 Einführung und Rekapitulation

2.1 Sicherheitskomponenten

- Firewalls
- Intrusion Detection/Prevention Systems
- Proxies
- interne oder private Netzwerke / Netzwerkzonen / Entmilitarisierte Zonen
- VPNs

2.2 Firewalls

entscheidet ob Verkehr ins Netzwerk gelangen darf oder nicht Typen:

- Paketfilter
- Zustandsbehaftete Firewall
- Proxyfirewall

2.3 Intrusion Detection Systems

Identifizierung von Attacken / verdächtigem Verkehr
Hilfe beim Einrichten/ konfigurieren von Firewalls
Normalerweise transparent für Nutzer und Angreifer.
hauptsächlich 2 Arten:

- Mustererkennung
- Anomalieerkennung

2.4 Proxies

strickte Trennung von internen und externen Netz
Üblicherweise auf Application Layer. Verhindert dass bestimmte Informationen(Viren, Pornos, illegale Infos) in das interne Netz gesandt werden.
Verhindert, dass bestimmte Informationen nach außen gesendet werden.
Kombinationen mit anderen Systemen(Virenfilter/ Spamfilter / IDS...)

2.5 VPN

VPNs erschaffen einen gemeinsamen Addressraum.
VPNs schützen die Kommunikation über ungesicherte Netzwerke als würde sie in einem Netzwerk stattfinden.
Gegenseitige Authentifizierung der Kommunikationspartner.
VPNs bieten signifikante Einsparungen über dedizierte Verbindungen.

2.6 Zonen - DMZ

kleine Netzwerke, welche öffentlich erreichbare Dienste beeinhalten (z.B. HTTP)
DMZ oft durch Firewalls etc geschützt.
DMZ befinden sich außerhalb des internen Netzes.
sind unsicherer als das interne Netz.
Abgeschirmte Teilnetze sind isolierte Netze innerhalb des internen Netzes

2.7 internes Netz

eingeschränkter Zugriff auf das externe Netz nur über gut bekannte Ports
Internes Angriffsrisiko hängt ab von:

- Anzahl der Nutzer
- Vertrauen in die Nutzer

- Zugriffswege der Nutzer(Notebooks?)
- Fähigkeiten der Nutzer

Hosts müssen trotzdem noch mit firewalls etc geschützt werden

2.8 Basis Kryptografie

Kerkhoff's Prinzip: Sicherheit hängt nur von Schlüssel ab und nicht von der Kenntnis der kryptografischen Funktion.

2.8.1 symmetrische Verschlüsselung

Beide Teilnehmer benutzen zum ver- und entschlüsseln denselben Schlüssel.
Stromchiffren: Klartext wird Zeichen für Zeichen ver- und entschlüsselt.
Blockchiffren: arbeitet mit festen Blockgrößen und entschlüsselt mehrere Zeichen in einem Schritt.

One-Time Pads Stromchiffre deren Schlüsselstrom ein Strom aus echten Zufallsbits ist
Uneingeschränkt sicher(einziges bisher „bewiesen" sicheres Verfahren).
Schlüssel muss zu verschlüsseln mindestens so lang sein wie der Klartext.
Jeder Schlüssel darf nur einmal verwendet werden.
Nachteil: viel Speicherbedarf für Schlüssel.

2.8.2 asymmetrische Verschlüsselung

Sender und Empfänger nutzen jeweils unterschiedliche Schlüssel.
Es ist schwierig den Entschlüsselungsschlüssel(k') aus dem Verschlüsselungsschlüsselungsschlüssel(k) zu berechnen
k kann öffentlich gemacht werden (public-key-Verschlüsselung).
Nachteil: Verteilung der Schlüssel

2.8.3 hybride Verschlüsselung

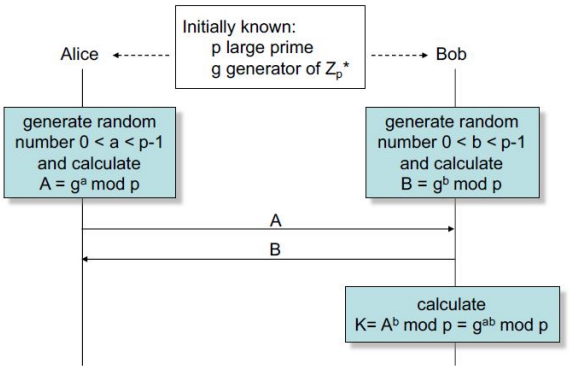
Kombination aus symmetrischer udn asymmetrischer Verschlüsselung.
symmetrischer Session Key mit dem die Daten symmetrisch verschlüsselt werden.
Session Key wird asymmetrisch mit public Key des Empfängers verschlüsselt.
löst Verteilungsproblem der asymmetrischen und behält Geschwidigkeit der symmetrischen Verschlüsselung

2.8.4 kryptografische Hashfunktion

Anforderungen:

- einseitig: wenn Hashwert y gegeben ist, ist es rechnerisch unmöglich eine Nachricht x zu finden, sodass $h(x) = y$
- schwacher Kollisionswiderstand: bei gegebener Nachricht ist es rechnerisch unmöglich eine andere Nachricht mit gleichem Hashwert zu finden
- starker Kollisionwiderstand: Es is rechnerisch unmöglich zwei Nachrichten mit gleichem Hashwert zu finden.

Hash und Signaturen Von Nachricht wird Hash gebildet. Dieser wird verschlüsselt und als Signatur an die Nachricht gehängt.
Empfänger entschlüsselt Signatur mit public key des Senders und vergleicht mit dem hash der Nachricht. Wenn gleich dann ist alles gut, wenn nicht dann wurde was verändert.



2.8.5 Diffie-Hellman Schlüsselaustausch

2.8.6 Zertifikate

Zertifikat ist eine Datenstruktur welche folgendes enthält:

- Öffentlichen Schlüssel
- Namen des Eigentümers des öff Schlüssels
- Namen des Ausstellers
- Ausstellungsdatum
- Ablaufdatum
- Möglicherweise andere Daten
- Signatur des Ausstellers

2.8.7 Certification Authorities (CA)

stellen Zertifikate aus.
sind normalerweise vertrauenswürdige Dritte.
Zertifikate werden über online Datenbanken verteilt(Certificate Directories) denen vertraut werden muss.

3 Paketfilter

3.1 Funktionsweise von Paketfiltern

Netzwerkpakete werden akzeptiert oder zurückgewiesen anhand von Parametern wie:

- Quelladresse/Ports
- Zieladresse/Ports
- Flags

3.2 Paketfilterregeln

Regeln können bzgl Flags, Adressen und Ports angewandt werden.
Paketfilter können auch bezüglich des Inhaltes von Paketen angewandt werden.
Zulassende Regeln:

- explizites Erlauben von Zugriff
- sämtlicher anderer Verkehr wird verhindert

Verhindernde/ablehnende Regeln:

- bestimmter Verkehr wird explizit abgelehnt.
- sämtlicher anderer Verkehr wird für gewöhnlich zugelassen.

Wichtig:

- Reihenfolge der Regeln ist wichtig.(erst alles verhindern und dann einige zulassen ist was anderes als erst einige zulassen und dann alles zu verhindern!)
- große Anzahl an Regeln kann verwirrend sein.
- „alles verbieten und solange es nicht explizit benötigt wird“ kann gute Herangehensweise sein.

3.2.1 Ingress-Filter

Filtern ankommende Pakete
blockieren Zugriff von verdächtigen Quelladdressen.

3.2.2 Egress-Filter

Filtern ausgehenden Verkehr.
Nur Pakete mit Quelladdresse im Netzwerk dürfen das Netzwerk verlassen so lange keine andere Regel greift.
Quellen von abgewiesenen Paketen sind gute Kandidaten für Überprüfung.

3.2.3 Protokollfilter

Dienste haben haben festgelegte Protokolle
Daumenregel: nur Verkehr zu Diensten zulassen die wirklich benötigt werden.

3.2.4 Probleme

Zugriff für bestimmte Netze zulassen
Gefahr des Spoofings: Angreifer nutzt evtl falsche Quelladressen
Source Routing:

- Pakete enthalten evtl Infos über die Route zurück zum Urheber
- Überschreiben die Routingtabelle des Routers

Gefahr das Filterregeln umgangen werden
Fragmentierung:

- Paketfilter untersuchen Headerinfos
- Paket wird so aufgeteilt dass der Header geteilt wird und Adresse und Ports nicht gefiltert werden können.

Löcher:

- Dienste müssen erreichbar bleiben für externe Netzwerke
- entsprechende Ports müssen geöffnet werden.

3.3 dynamische Paketfilter

Filterregeln werden on the fly so erstellt wie sie benötigt werden und nach schließen der Verbindung wieder gelöscht.
Filter beobachten ausgehenden Verkehr und erstellen zurückwirkende Regeln.(Ausgehender Verkehr zu einer Adresse bewirkt Regel dass eingehender Verkehr von dieser Adresse erlaubt wird.)
Probleme:

- Regeln sind angreifbar z.B. durch senden falscher reset Pakete
- ausgehender Verkehr wird nicht gefiltert. Gefahr von Trojanern/Viren.

4 Zustandsbehaftete Firewalls

4.1 Funktionsweise

Kennen den Zustand von Verbindungen und wissen welche Pakete in welchem Zustand erwartet werden.
Es können Regeln angewandt werden die nur in bestimmten Zuständen wirksam sind.
Untersuchen hauptsächlich OSI 4 (transport layer), aber auch höhere Schichten.

4.2 Probleme

Hohe Leistung benötigt teilweise geclusterte Hardware. Zustandsbehaftete Firewalls lassen sich nicht einfach clustern.
Zustandslose Protokolle (UDP, ICMP, DNS, HTTP)

4.2.1 Zustandslose Protokolle

Zustandslose Protokolle definieren trotdem welche Pakete erwartet werden.
Timeouts werden genutzt um Pseudo-Verbindungen zu erzeugen.

4.3 Multi-Layer Inspection

Die meisten Protokolle basieren auf Protokollen aus niedrigeren Layern. BSP: HTTP nutzt TCP Verbindungen. Zustandsbehaftete Firewalls können beide Layer beobachten.

5 Proxy Firewalls

Proxies verhalten sich für den inneren Client wie der äußere Server und andersherum. Client und Server werden niemals direkt miteinander agieren. Der Proxy ist außerdem intransparent. Auf dem Proxy selbst können ein paar Programme laufen, die sicher sind und denen vertraut werden kann.
Proxies können die interne Struktur eines Netzwerkes nach außen hin verstecken und gegen Protokoll-Angriffe schützen. Forward Proxies sind ein Mittel, um ausgehenden Traffic zu kontrollieren - Reverse Proxies für eingehende Verbindungen.

Beispiel: Der Nutzer fragt eine HTTP-Ressource an. Dessen Software leitet die Anfrage an den Proxy weiter. Der Proxy baut die Verbindung auf und gibt sich als Client aus, der die HTTP-Ressource beim Server anfragt. Sämtlicher Traffic zwischen dem internen Nutzer und dem externen System wird durch den Proxy geleitet.

5.1 Bastion Host

Unter einem **Bastion Host** versteht man in diesem Kontext einen Proxy der auf das öffentliche Internet zugreift und daher besonders gegen Angriffe geschützt und abgehärtet werden muss. Dies ist notwendig, da der Proxy-Server von außerhalb des Netzwerks sichtbar ist. Da Proxies kein IP-Forwarding machen, besitzen sie üblicherweise mit zwei Network-Interfaces. Die interne Struktur des Netzwerks bleibt vor äußeren Einblicken gesichert. (Passive Fingerabdrücke sind nicht möglich)

5.2 Arten von Proxies

- **Forward Proxy**: Weitverbreitetste Form, bei der die Verbindung vom internen Client initiiert wird
- **Reverse Proxy**: Hier wird die Verbindung von der externen Seite initiiert. Wird von Sicherheitsservices genutzt: Der ankommende Datenverkehr wird vom Proxy überwacht.
- **Application-Level Proxy**: Bietet für jeden Service Policies, die bestimmten Traffic genehmigen (bspw. Nutzer, Adressen, Computer,...)
- **Transparent Proxy**: Ein transparenter Proxy (intercepting proxy, inline proxy, or forced proxy) leitet die normale Kommunikation für beide Seiten normal auf dem Network-Layer weiter. Es ist keine spezielle Konfiguration notwendig. Der Client muss sich der Existenz des Proxies nicht bewusst sein.
- **Intercepting Proxies** Diese „abfangenden“ Proxies werden üblicherweise benutzt um Policies durchzusetzen, ohne dass eine clientseitige Browserkonfiguration notwendig wäre. Bspw. Nacktfilter
- **Intransparente Proxies** Für diese Art muss der Nutzer sein Gerät anpassen, um den Proxy nutzen zu können.
- **Circuit-level Proxy** Die Filterung dieses Proxies wird durch speziellere Regeln definiert. Der Inhalt wird auf Schlagwörter, Größe, Viren, Datentypen, Bilderkennnung, Passwörter oder ähnliches geprüft. Des Weiteren ist Authentifikation/ Legitimation möglich, um bestimmten Nutzern Rechte einzuräumen.

5.3 SOCKS

Bei SOCKS handelt es sich um ein Proxy-Toolkit. Es ermöglicht, dass Anwendungen sich ohne spezielle Client-Software mit Proxies verbinden können. Ein SOCKS-Server führt Client Authentifizierung und Authorisierung durch. Zur Kommunikation mit einem SOCKS-Server sind jedoch Modifikationen notwendig.

Der Client sendet einen Request an den Server. In diesem Request sind die Identität des Clients, die Ziel-Adresse (Ausgehende Verbindungen) ODER der Port (Eingehende Verbindungen.
Der Proxy prüft (als Server), ob der Request bewilligt werden soll. Im positiven Fall, antwortet er mit einem Reply-Paket, welches den Return-Code der Operation beinhaltet.
Protokoll:

- Der Client möchte sich mit einem externen Service verbinden und sendet diesen Request:
  | VN | CD | DSTPORT | DSTIP | USERID | ... | NULL |
  (VN-Versionsnummer, CD-Command)
- Reply | VN | CD | DSTPORT | DSTIP |
  (90: request granted, 91: request rejected or failed, 92: request rejected becasue SOCKS server cannot connect to identd on the client, 93: request rejected because the client program and identd report different user-ids)
- Client bietet eine eingehende Verbindung an und sendet einen Bind-Request
  | VN | CD | DSTPORT | DSTIP | USERID | ... | NULL |

5.4 Umgang mit Verschlüsselung

Wenn die übertragenen Daten verschlüsselt sind, ist eine Ende-zu-Ende-Verschlüsselung nicht möglich. Der Proxy kann die übertragenen Daten nicht verändern, wenn diese verschlüsselt sind. Deshalb ist es unmöglich den Inhalt zu überprüfen.
Es kann zu Problemen bei der Authentifizierung kommen, wenn zwischen Server und Client ein Proxy sich befindet. Diese Probleme treten insbesondere dann auf, wenn dieser versucht die Daten zu entschlüsseln.
Der Proxy kann jedoch als Client agieren. In diesem Fall überträgt er die Daten entschlüsselt zum Client und verschlüsselt zum Server.

5.5 Diskussion

Vorteile:

- Schutz der internen Struktur nach Außen
- Datentraffic kann einfach überwacht werden
- Nutzerbezogene Sicherheit ist möglich
- Authentifikation kann implementiert werden
- Schützt vor Spoofing (Der Proxy generiert sämtlichen ausgehenden Traffic für einen Außenstehenden)

Nachteile:

- Leistungseinbußen
- Single Point of Failure/Attack
- Anwendungsspezifische Proxies müssen für jede einzelnen Anwendungen entwickelt werden
- Software muss angepasst werden
- Bastion Host muss gehärtet werden

6 Policies

6.1 Was sind Policies?

Eine Sicherheitsrichtlinie(security policy) beschreibt was getan werden muss um auf einem Rechner gespeicherte Infos zu schützen.
Richtline definiert was gemacht werden muss und und wie es ausgewertet werden kann.
Werden normalerweise aufgeschrieben.

6.2 Komplexität von Richtlinien

werden von Menschen definiert.
müssen verständlich formuliert sein.
zu komplexe (aber auch zu einfache. bsp „keine rechner benutzen!“) Richtlinien sind nicht durchsetzbar.

6.3 Entwicklung von Richtlinien

beste Vorgehensweise:

- Risiken identifizieren
  - Sicherheitsanalyse
    - kritische Daten und Systeme identifizieren
    - normale Nutzung des Netzwerkes feststellen
  - aufschreiben
- Funde kommunizieren
  - Dem Management berichten
    - einfach
    - ausgeglichen
    - präzise
    - Zeigen auf einzelne vermeiden
    - Allgemein halten
- Richtlinie erstellen oder aktualisieren
  - aufschreiben
  - Genauheit und Klarheit
    - Was muss getan werden?
    - Warum?
    - Wer ist verantwortlich?
  - Knappheit: Niemand liest mehr als 10 Seiten.
  - Realismus
- Einhaltung der Richtline kontrollieren
  - Wenn die Einhaltung einer Regel nicht kontrolliert werden kann ist sie nicht durchsetzbar
  - Stichproben, Log Analysen, Festplattendurchsuchungen
- Versuchen eine „Kultur“ zur Einhaltung der Richtlinie einzuführen
  - Anpassung an die Richtlinie hängt stark vom Verhalten der Nutzer ab
  - Mit Nutzern über Risiken sprechen
  - Richtlinien vor der Einführung erklären
  - Anweisungen und autoritäres Verhalten vermeiden

6.4 ungeschriebene Richtlinien

versteckte Regeln existieren.
nutzen des gesunden Menschenverstandes
Versuchen ein Sicherheitsbewusstsein im Betrieb zu etablieren.
Wissen verbreiten, aber vorsichtig und sensibel.

7 Intrusion Detection Systems (IDS)

Sind dazu bestimmt Angriffe zu erkennen und nicht um diese zu verhindern.
Netzwerk IDS Sensor liest Traffic mit uns analysiert diesen indem nach Zeichen für:

- Scans/Sonden
- Aufklärungsaktivitäten
- Exploits

Ist normalerweise komplett transparent (nicht zu entdecken).

7.1 Motivation

Ohne IDS würde ein Admin die meisten Angriffe nicht bemerken und kann auf diese somit nicht reagieren.
Fehlende Infos ohne IDS:

- Welche Hosts wurden angegriffen?
- Welche Daten wurden kompromittiert?
- Mit welcher Methode wurde angegriffen?

Nachfolgende Schritte eines Angriffes können verhindert werden.

7.2 Methoden- Anomalieerkennung

Statistische Analyse um „unnormalen“ Verkehr erkennen zu können.
Parameter wie Herkunft, Datenrate, Ports und Zeit werden berücksichtigt und gegen eine Statistik geprüft, jedoch nicht nach bestimmten Mustern.
Berechnung der Wahrscheinlichkeit dafür das Verkehr unnormal ist mithilfe von Bayesschen Filtern.
Training der Filter mit Verkehr der als normal betrachtet wird.
Wenn eine bestimmte Grenze überschritten wird, wird Alarm ausgelöst.

7.3 Methoden- Signaturerkennung

Analyse von Paketen anhand von gegebenen Mustern.
Adressen und timing Pattern werden berücksichtigt.
bestimmte Mustern lösen einen Alarm aus.

7.4 Probleme mit IDS

Fehlalarme (false positives) und nicht erkannte Alarme (false negatives).

- Viele Fehlalarme werden zu einer bedeutend höheren Ignoranz seitens des Admins führen.
- Reduzierung der false positives führt gewöhnlich zu mehr false negatives.
- IDS Evasion um false positives zu verringern
- Als ersten Filter ein allgemeines Muster Nutzer
- Spezifischere Untersuchung der Pakete die den ersten Filter durchlaufen haben.

7.5 Ansätze für IDS

globale vs lokale Ansätze:

- betrachtet die Auflösung der Referenzmenge bezüglich ob ein bestimmtes Datenobjekt als Außenseiter erkannt wurde.
- globale Ansätze:
  - Referenzmenge enthällt alle Daten.
  - Basisannahme: es gibt nur einen normalen Mechanismus (normal mechanism?).

  - Problem: andere Ausreißer sind auch in der Menge und können das Ergebnis verfälschen.

- lokale Ansätze:
  - Die Referenz enthält lediglich eine Teilmenge der Daten.
  - keine Annahme über die Anzahl von normalen Mechanismen (normal mechanism?).
  - Problem: Wie fählt man eine geeignete Referenzmenge

Einige Ansätze befinden sich irgendwo dazwischen.
Die Auflösung der Referenzmenge kann automatisch oder durch eine Nutzereingabe verändert werden.

7.6 Statistische Tests

Idee:

- Gegebene Wahrscheinlichkeitsverteilung (z.B. Gauss)
- Parameter berechnen unter der Annahme, dass alle Datenpunkte durch eine solche Wahrscheinlichkeitsverteilung generiert wurden.
- Ausreißer sind die Punkte, die eine geringe Wahrscheinlichkeit haben von der Verteilung generiert worden zu sein (z.B. weicht mehr als 3 mal von der Standardabweichung ab).

Annahme:

- Normale Daten folgen einer Verteilung und treten in einer Region des Models mit hoher Wahrscheinlichkeit auf.
- Ausreißer weichen stark von dieser Verteilung ab.

Viele verschiedene Tests (unterschiedliche Verteilung, Menge der Variablen, Menge der Verteilungen....)

7.7 Tiefenbasierte Ansätze

Idee:

- Suche nach Ausreißern an den Grenzen des Datenraums aber unabhängig von Wahrscheinlichkeitsverteilungen.

- organisieren Daten in Schichten von konvexen Hüllen
- Ausreißer sind Objekte in äußeren Schichten.

Annahme:

- Ausreißer befinden sich an den Grenzen des Datenraums und normale Daten im Zentrum.

7.8 Abweichungsbasierte Ansätze

Idee:

- Gegeben: Menge von Datenpunkten
- Ausreißer sind Punkte die nicht in die allgemeinen Charakteristiken der Menge passen.

Annahme:

- Ausreißer sind die äußersten Punkte der Datenmenge.

7.9 Abstandsbasierte Ansätze

Idee:

- Punkte werden basierend auf ihrem Abstand zu ihren Nachbarn bewertet.

Annahme:

- normale Daten haben eine dichte Nachbarschaft
- Ausreißer sind weit von ihren Nachbarn entfernt.

Beispiel: k-Nearest-Neighbours

7.10 Dichtebasierte Ansätze

Idee:

- Vergleiche Dichte um einen Punkt mit der Dichte um seine lokalen Nachbarn
- Die relative Dichte eines Knotens im Vergleich zu der seines Nachbarn wird als Ausreißerwert berechnet.
- Ansätze unterscheiden sich in der Bewertung der Dichte

Annahme:

- Die Dichte um normale Daten ist gleich der Dichte um seine Nachbarn.
- Die Dichte um einen Ausreißer ist erheblich anders als die Dichte um seine Nachbarn.

8 Honeypots and Tarpits

8.1 Honeypot

Antivirus Software Hersteller möchten neue Viren und Varianten kennenlernen

- Forschungshoneypots
- gewöhnlich große verteilte Netzwerke von Fallen

Netzwerkadmin möchte Informationen über aktuelle Bedrohungen erlangen

- gewinnbringende (productive?) Honeypots
- um Infos über bekannte Angriffstypen zu erlangen

8.1.1 Honeypots um Angreifer abzulenken

Admin möchte Angreifer vor verletzlicheren Zielen ablenken Aus Netzwerksicherheitstechnischer Sicht keine akzeptable Methode!

8.1.2 Arbeitsprinzipien

Honeypots verhalten sich wie reale Systeme.
Angreifer sollen nicht mitbekommen können, dass sie mit einem Honeypot interagieren.
Honeypot zeichnet alle Nutzer/Client Handlungen auf.
Admin möchte den Spuren eines Angreifers folgen.
Später folgt eine Analyse der Angriffsmuster.

8.2 Honeynets

Reale Systeme hinter einem Gateway beobachten alle Netzwerkaktivitäten.
Es können nur Netzwerkaktivitäten beobachtet werden, lokale Aktivitäten (Viren etc.) sind nicht interessant.

8.3 Spam Traps

E-Mail Adressen werden in gut sichtbaren Bereichen platziert.
Alle Nachrichten die an diese Adressen geschickt werden, werden als Spam betrachtet, da die Köder nicht für echt Anwendungen benutzt werden.
Später Blacklisting der entsprechenden Adressen und Sender.

8.4 Virus Traps

Antivirus Forscher zeichnen alle Virusaktivitäten in einer „Sand Box“ auf.
Große Vielfalt an Systemkonfigurationen ist notwendig.
Gegenmaßnahmen müssen sehr schnell (normalerweise innerhalb von 6 Stunden) entwickelt werden.

8.5 Tarpits

Wie Honeypots, jedoch mit aktiver Funktion.
Verlangsamen Angreifer indem Warteperioden in Protokolle eingeführt werden.
Reduzieren die Verbreitungsgeschwindigkeit von Würmen.
Erhöhen die Kosten für die Angreifer.
BSP: Tarpit verzögert die SMTP Antworten solange, dass fast der Timeout eintritt.

8.6 Proof-of-work Systeme

Proof-of-work (POW) fügt einem Dienst einen Preis hinzu.
ökonomische Maßnahme um DoS Angriffe oder Spam zu verhindern.
benötigen für gewöhnlich Rechenzeit.
Können als Nebenprodukt für praktische Rechenaufgaben benutzt werden (Jeder Sender einer Mail rechnet also ein kleines bisschen an einem Problem das eh gelöst werden soll weiter).

9 Public Key Infrastructure

Ziele/Zwecke von PKI:

- Erzeugung,
- Verteilung und
- Widerrufung von Zertifikaten.

- Unterstützung bei sicherer Kommunikation und in der Handhabung von rechtlich bindenden Dokumenten (Signaturen, Nichtabstreitbarkeit).
- Komponente im DRM System

Public Key Kryptographie: Ein Schlüssen zum Verschlüsseln, ein anderer zum Entschlüsseln.
bekannte Algos sind: Diffie-Hellman(DH) und RSA.

9.1 Zertifikaterstellung

Nutzer erstellt Schlüsselpaar(Als Datei oder auf einer Smartcard).
Public Key muss authentifiziert werden.

- CA signiert den Public Key und generiert damit ein Zertifikat.

9.2 PKI Protokolle

X.509:

- ermöglicht Interoperabilität zwischen mehreren Anwendungen (Webserver, Mail tool, VPN Gateway).

LDAP:

- wird herkömmlicherweise genutzt um X.509 Zertifikate und Revocationlists der PKI zu verteilen.

9.3 Zertifikatwiderruf

Zertifikate können öffentlich verteilt hochverfügbar gemacht werden.
Private Keys können verloren gehen oder gestohlen werden.

- verlorene Keys können erneut generiert werden.
- gestohlene Keys stellen eine Sicherheitsbedrohung dar.

CA unterhält eine Liste der widerrufenen Keys- Certificate Revocation List (CRL)

9.4 Zertifikatsverwaltung

normalerweise stark reguliert (durch Firmenrichtlinien oder nationale Gesetze).
Funktionen:

- öffentlicher Aufbewahrungsort für Zertifikate (LDAP).
- Ablage für Revocation List.

9.5 Schlüsselverwaltung

private Keys müssen gegen Diebstahl und Verlust geschützt werden.
Manche PKI Systeme generieren Keys für Nutzer und liefern diese zu ihnen.
Chronik der public Keys muss gepflegt werden.

9.6 Client Software

interagiert mit Server Komponenten.
erlaubt Schlüssel- und Zertifikatsverwaltung.

9.7 Hardware Tokens

private Keys sind gefährdet wenn die auf Festplatten gespeichert sind.
private Keys können nicht einfach verteilt werden.
Tokens schützen Schlüssel und bieten zusätzliche Sicherheit, da der Nutzer ein physikalisches Token vorliegen hat.
Formen: Smartcards, RFID Tags, USB Token, iButton.

10 Enterprise Authentication

11 Securing Hosts and Appliances

12 Organisational Aspects

13 Computer Forensics