honeytrap – Ein Meta-Honeypot zur Identifikation und Analyse neuer Angriffe

Tillmann Werner

CERT-Bund, Bundesamt für Sicherheit in der Informationstechnik

14. DFN-Workshop *„Sicherheit in vernetzten Systemen“*
8. Februar 2007

Übersicht

- Einleitung
- honeytrap – ein Meta-Honeypot
- Dynamische Server
- Identifikation neuer Angriffe
- Experimentelle Ergebnisse
- Zusammenfassung

Einleitung: BSI

- Bundesamt für Sicherheit in der Informationstechnik, Bonn
- Drei Fachabteilungen + eine Verwaltungsabteilung
- Abteilung 1: *Sicherheit in Anwendungen, KRITIS und im Internet*
- Referat 121 – *Computer-Notfallteam für Bundesbehörden*

- Aufgaben von CERT-Bund
  - Veröffentlichen von Advisories zu aktuellen Schwachstellen
  - Monitoring und Bewertung der Bedrohungslage im Internet
  - Incident Handling bei Vorfällen im Bereich Bund
  - Anlassbezogene Analyse von Schadprogrammen und Angriffen
  - Point-of-Contact für internationale CERTs

Einleitung: Honeypots

Definition

A honeypot is an information system resource whose value lies in unauthorized or illict use of that resource.[a]

[a] *Know Your Enemy*, The Honeynet Project, Addison-Wesley, 2004

- Unterscheidung zwischen **low-interactive** und **high-interactive** Honeypots je nach Interaktionslevel
- **Client-seitige Honeypots** als Ergänzung des passiven Ansatzes
- In den letzten Jahren vor allem **Grundlagenforschung**, heute **Verwertung der Ergebnisse** in der Praxis

Einleitung: Anpassung an Angriffstrends

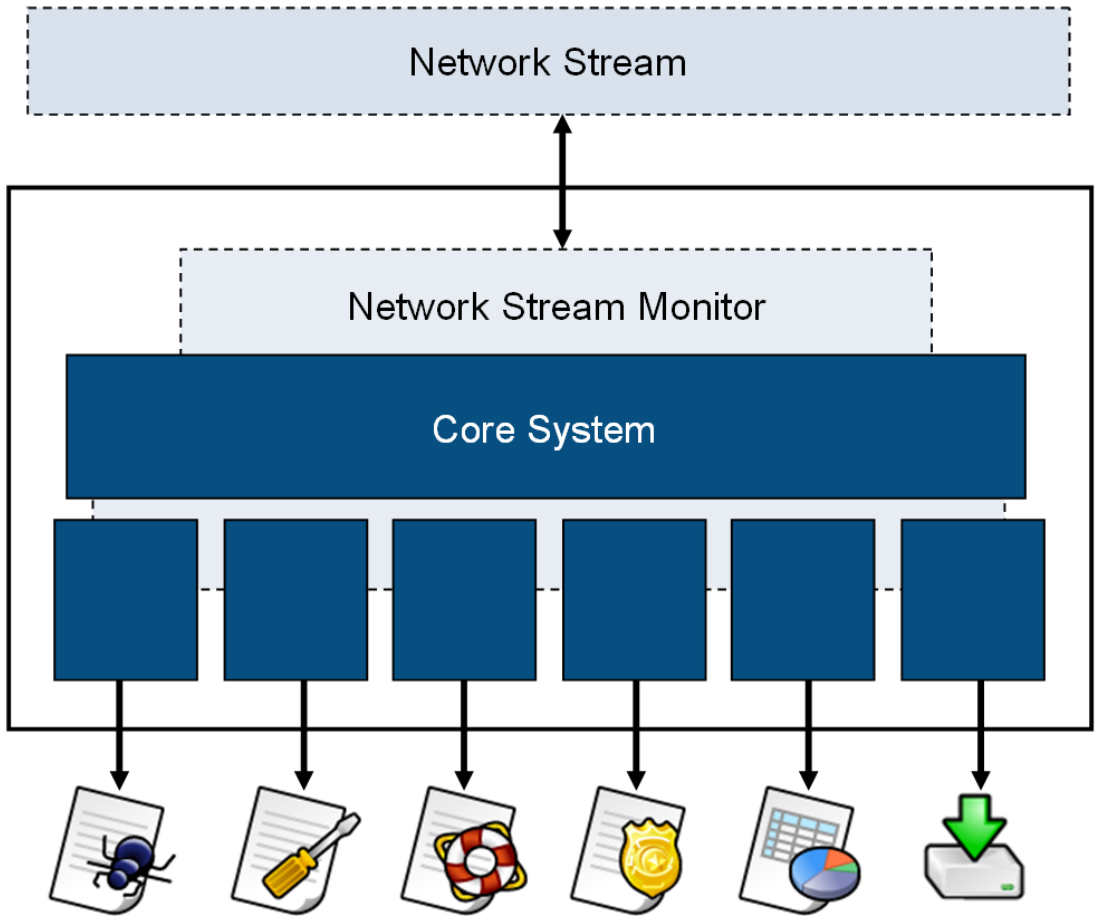
- Honeypots in einer Sicherheitsinfrastruktur: **Ableiten von Sicherheitsmaßnahmen** zum Schutz verwundbarer Systeme
- Sinnvoller Umgang mit neuen, bisher **unbekannten Angriffen** ist größte Herausforderung beim Einsatz von Honeypots
- Mögliches Maß für die Güte eines Honeypots:
  Flexibilität bei der **Anpassung an Angriffstrends**
- Diese Flexibilität ist in vielen Setups ein **Haupt-Schwachpunkt**

honeytrap: Idee

- **low-interaction Honeypot**,
  dynamischer Umgang mit netzbasierten Angriffen
- Einfach zu nutzende **Schnittstelle für die Analyse** von Angriffsdaten
- Programm läuft als **Daemon-Prozess** auf Unix-Systemen
- Angriffsdaten werden aufgezeichnet und können mit Plugins automatisiert weiter verarbeitet werden
- Projekt-Webseite: `http://honeytrap.sourceforge.net`

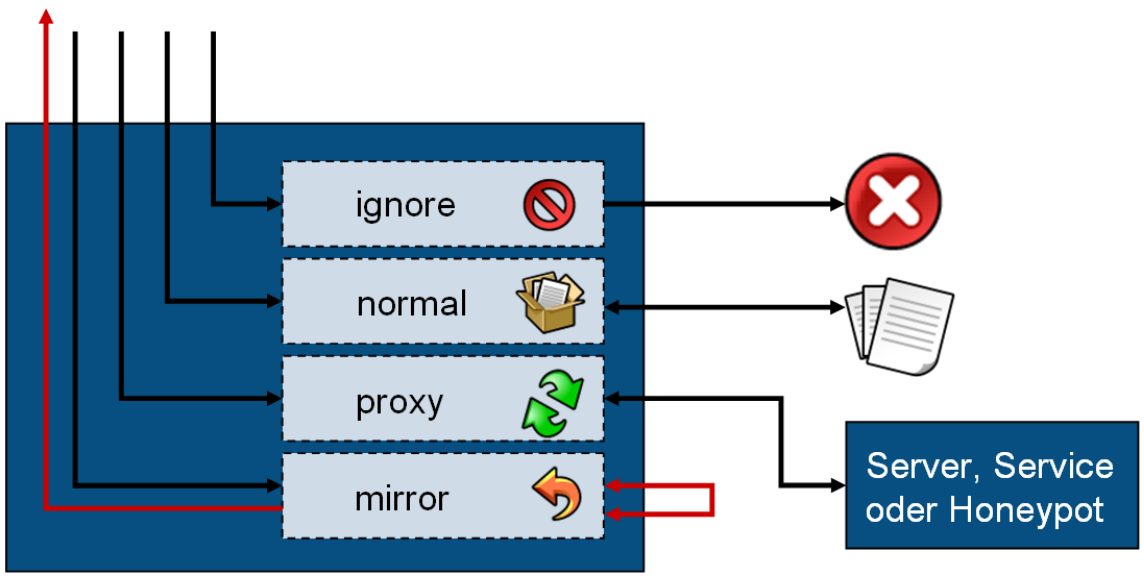
honeytrap: Architektur

- Verfügbare Plugins (unter anderem): Malware-Download via ftp und tftp, Base64-Decoder, VNC-Keystroke-Assembler



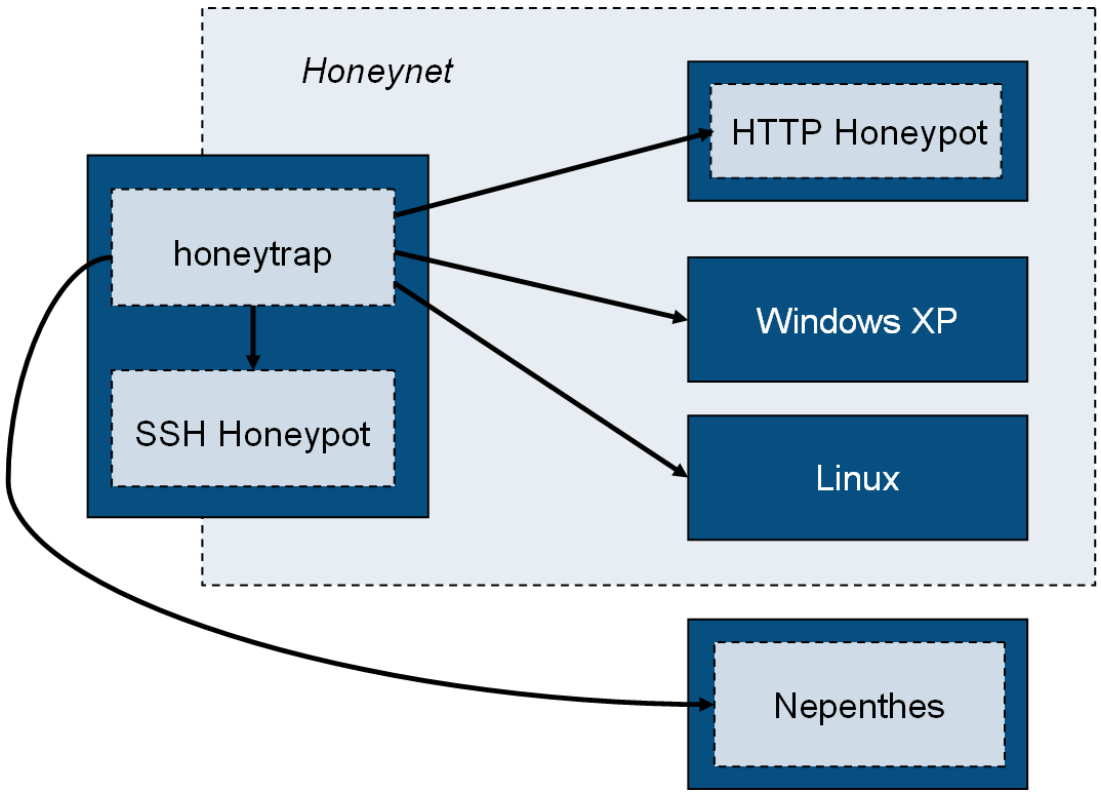
honeytrap: Operationsmodi

- honeytrap unterscheidet **vier Modi** zur Verarbeitung von Angriffen. Pro TCP- oder UDP-Port kann jeweils ein geeigneter Operationsmodus gewählt werden.



honeytrap: Beispiel-Setup mit Meta-Honeypot

- Als **Meta-Honeypot** kann honeytrap Angriffe entweder selbst verarbeiten oder an andere Honeypots vermitteln.

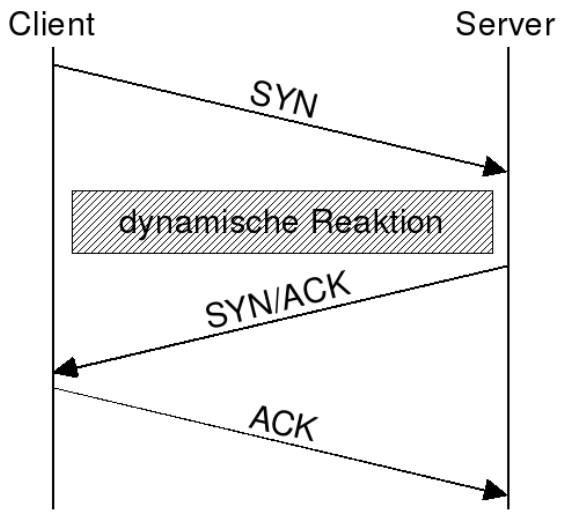


Dynamische Server: Idee

- `netcat`-Szenario: Verbindungen manuell einfangen ⇝ ☹
- **Automatische Annahme beliebiger Verbindungen**: Anfragen erkennen und dynamisch Server bereitstellen
- Start eines passenden Listeners mit begrenzter Lebenszeit (nachtriggerbar)
- Listener sind als **parallele Server** implementiert, die mehrere Verbindungen gleichzeitig verarbeiten können
- **Ressourcen werden freigegeben**, wenn ein Dienst für eine bestimmte Zeit nicht attackiert wurde

Dynamische Server: Aktives Modell

- **Aktive Einflussnahme** auf die Verarbeitung von Paketen: Verzögerung bis zur Bereitstellung eines passenden Listeners
- Trigger: TCP-SYN-Segmente und initiale UDP-Datagramme



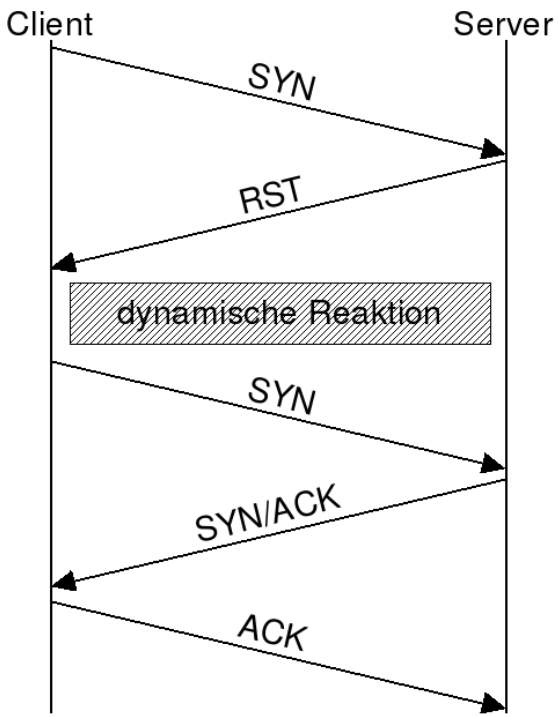
Dynamische Server: Konzept-Code für aktives Modell

ip_queue-basierter Netzwerk-Stream-Monitor:

```
 1  h = ipq_create_handle(0, PF_INET);
 2  ipq_set_mode(h, IPQ_COPY_PACKET, BUFSIZE);
 3  ipq_read(h, buf, BUFSIZE, 0);
 4  p = ipq_get_packet(buf);
 5  ...
 6  /* IP-Adressen, Protokoll-ID und Ports
 7     aus dem Puffer lesen */
 8  ...
 9  start_srv(src_ip, src_port, dst_ip, dst_port, proto);
10  ipq_set_verdict(h, p->packet_id, NF_ACCEPT, 0, 0);
```

Dynamische Server: Passives Modell

- Keine Einflussnahme, statt dessen **passives Abhören des Netzwerk-Streams**
- Trigger: TCP-RST-Segmente und ICMP-Port-Unreachable-Nachrichten mit UDP-Inhalt



Dynamische Server: Konzept-Code für passives Modell

pcap-basierter Netzwerk-Stream-Monitor:

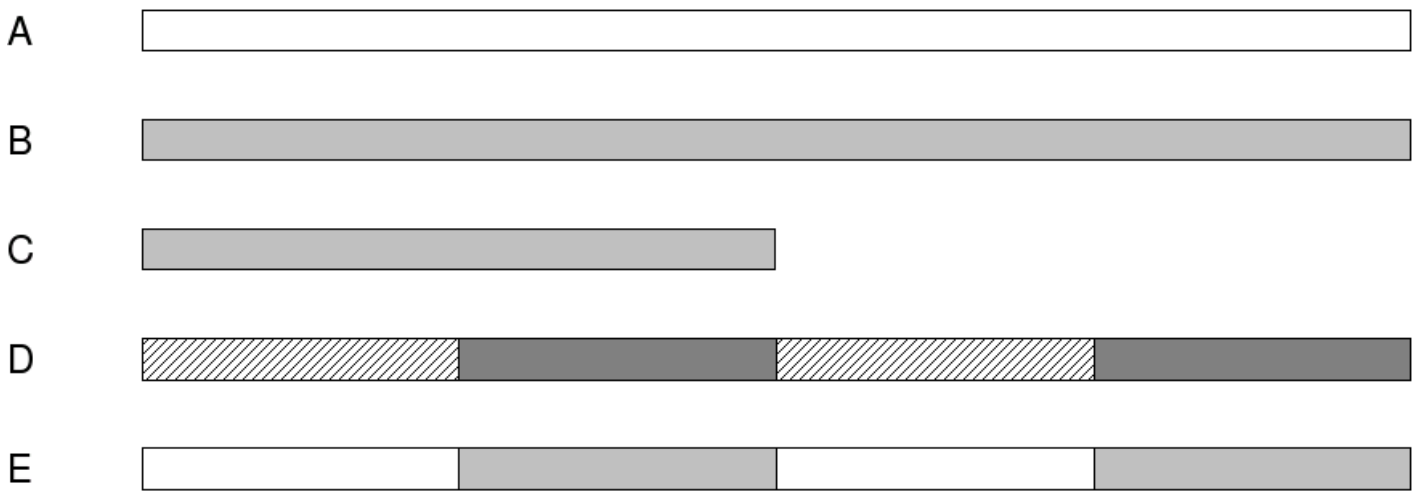
```
 1  char *pf = "(tcp[13]&4 != 0 and tcp[4:4] == 0) or "
 2             "(icmp[0] == 3 and icmp[1] == 3)";
 3  ...
 4  /* Bestimmung eines geeigneten, IP-fähigen
 5     Interfaces mit Netzmaske */
 6  ...
 7  netmon = pcap_open_live(dev, BUFSIZ, 0, 5, err);

 8  pcap_compile(netmon, &filter, pf, 1, net);
 9  pcap_setfilter(netmon, &filter);

10  pcap_loop(netmon, -1, (void *) start_srv, NULL);
```

Identifikation neuer Angriffe: Idee

- Analysieren des Byte-Streams (konkatenierter Payload) eines Angriffs und **Vergleich mit bekannten Attacken**
- **Berechnung eines Ähnlichkeitswerts** für alle bekannten und den neu observierten Angriff
- Bleibt der Wert **unter einer definierten Schwelle**, wird der Angriff **als neu klassifiziert**
- Schwierigkeiten
  - Geeignete Interpretation des Begriffs *ähnlich*
  - Effiziente Verfahren zur Berechnung dieser Ähnlichkeit

Identifikation neuer Angriffe: Interpretation von *Ähnlichkeit*

- Ohne Kenntnis der Beschaffenheit der Binärstrings sind **konkurrierende Ähnlichkeitsbegriffe** möglich
- Verfahren zur Ähnlichkeitsanalyse müssen wegen Eignung für unbekannte Angriffe **strukturunabhängig** sein



Identifikation neuer Angriffe: Formalisierung

- Wert aus dem Intervall $[0, 1]$ repräsentiert den Anteil der gleichen bzw. unterschiedlichen Information in Prozent

Ähnlichkeit zweier Binärstrings

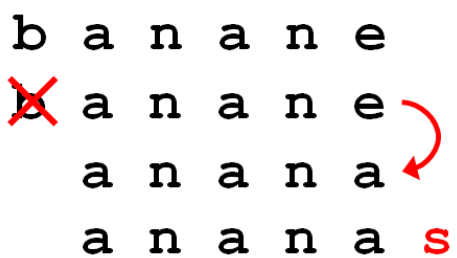
$$s : A \times A \to \mathbb{R},$$
$$s(a_1, a_2) \mapsto x, \qquad a_1, a_2 \in A, \quad x \in [0, 1] \subset \mathbb{R}$$

Unterschied zweier Binärstrings

$$d(a_1, a_2) := 1 - s(a_1, a_2)$$

Identifikation neuer Angriffe: Editierdistanz

- **Anzahl der Editier-Operationen** (Einfügen, Löschen, Ersetzen eines Zeichens) um einen String in einen anderen zu überführen
- **Normalisieren** liefert einen Wert zwischen 0 und 1, $s_{edit}(a_1, a_2) := 1 - d_{edit}(a_1, a_2)$
- Beispiel: Operationen zur Überführung von `banane` in `ananas`



Identifikation neuer Angriffe: N-Gramme

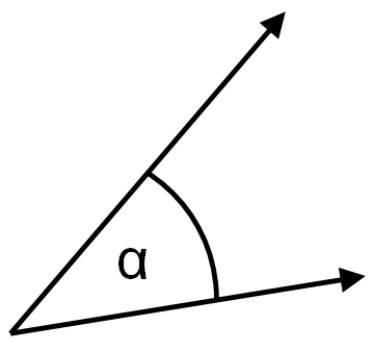
- Ansatz: **Überführen der Strings in Vektoren** und Berechnen des eingeschlossenen Winkels

Berechnung des Winkels und Normalisieren

$$\cos\theta = \frac{\langle \vec{a_1}, \vec{a_2} \rangle}{\|\vec{a_1}\| \|\vec{a_2}\|} = \frac{\sum_{i=1}^{n} a_{1_i} a_{2_i}}{\sqrt{\sum_{i=1}^{n} {a_{1_i}}^2} \sqrt{\sum_{i=1}^{n} {a_{2_i}}^2}}$$

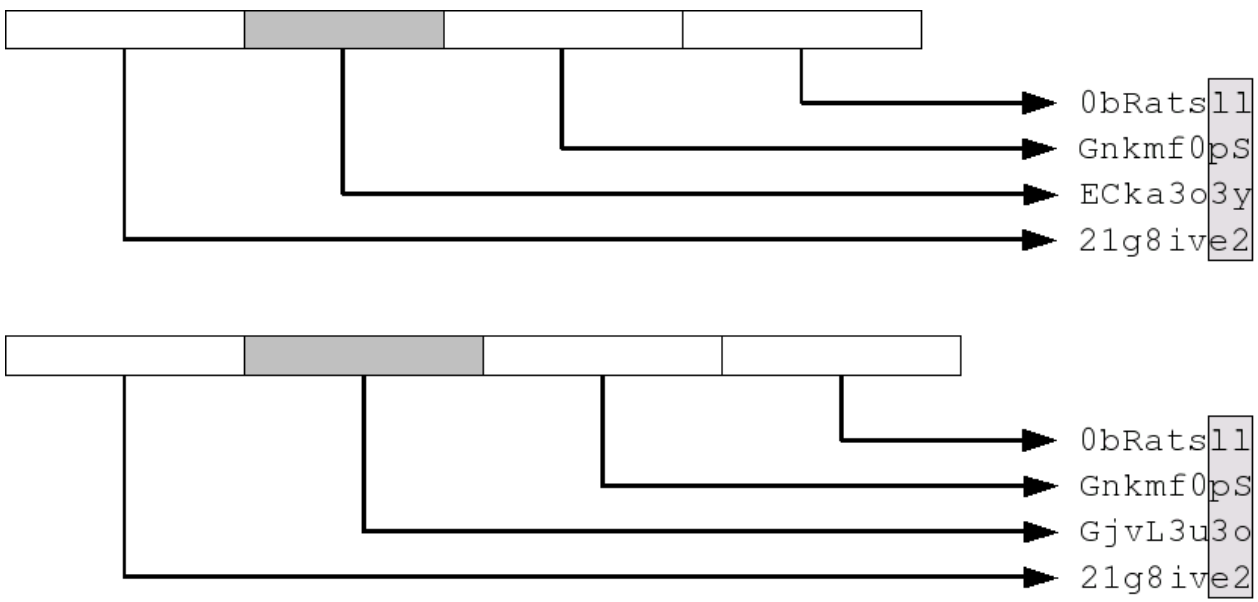
$$s_{ngram} := 1 - d_{ngram}, \qquad d_{ngram} := \arccos(\cos\theta) / \frac{\pi}{2}$$

aaa	3	2
aab	0	1
aac	4	0
...		
zzz	6	3



Identifikation neuer Angriffe: Locality Sensitive Hashing

- Berechnung **ähnlicher Hashes für ähnliche Strings**
- **Blockweise, klassische Hashes** ergeben entgültige Hash-Werte, deren Ähnlichkeit mittels Editierdistanz bestimmt wird
- Beispiel: Strings besitzen die Hashes `llpS3ye2` und `llpS3oe2`



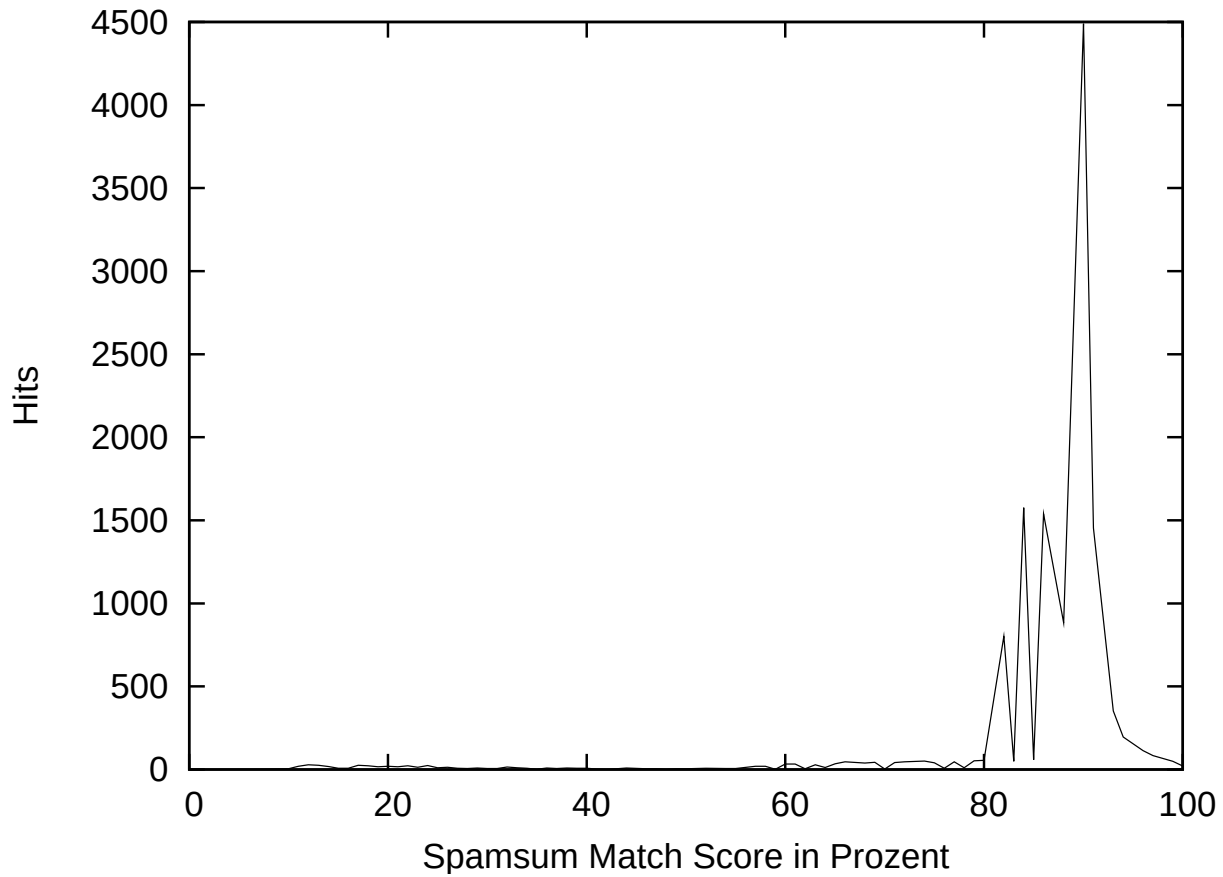
Experimentelle Ergebnisse: Vergleich von Bot-Generationen

Nr.	MD5-Checksumme	Bytes
1	02de24cfe3d31763e78bf6414010449b	1262592
2	40d5948ef1784115d898f84e93f2071f	1265664
3	5b9f4b16ccf11f4fc638e9a5197f06fa	1255424
4	7f4e623139695ef4b2f993e6fb8526d7	1256448
5	c1c875f9faa17d56d0c85af1d8bc4e9b	1260544
6	cad6fa541387f6c2748f31e30df5ccfc	1259520
7	ee5585af2667a7c194cfa617ac0b2610	1264640

Verfahren	durchschn. Zeit	Gesamtzeit
Editierdistanz	29121.10 s	611543.1 s
N-Gramm-Vergleich	13.39 s	281.10 s
Locality Sensitive Hashing	0.28 s	5.81 s

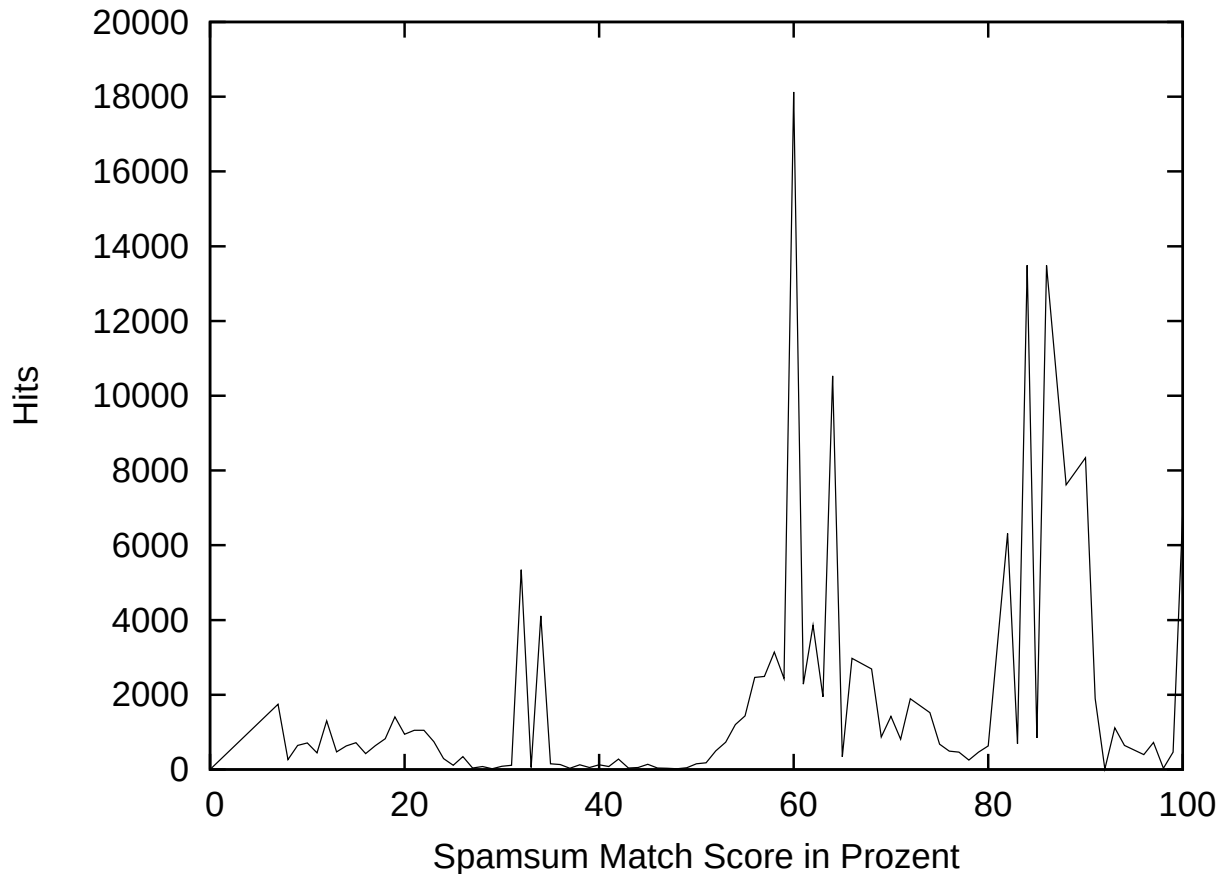
Experimentelle Ergebnisse: Angriffsanalyse (1)

- **Honeypot 1** – 28.092 Angriffe, davon 14.706 absolute Matches



Experimentelle Ergebnisse: Angriffsanalyse (2)

- **Honeypot 2** – 170.962 Angriffe, davon 9.440 absolute Matches



Zusammenfassung

- honeytrap kann netzbasierte **Angriffe als Meta-Honeypot flexibel koordinieren** und analysieren
- Honeypots können **Methoden zur Dynamisierung** einsetzen, um auch mit **unbekannten Angriffen** umgehen zu können
- Mit geeigneten Verfahren kann die **Ähnlichkeit zweier Byte-Streams** zu Angriffen bestimmt werden
- So lassen sich **neue Attacken automatisiert identifizieren**

Kontakt

Bundesamt für Sicherheit in der Informationstechnik (BSI)

Tillmann Werner
Referat 121 – CERT-Bund
Godesberger Allee 185-189
53175 Bonn

Tel.: +49 (0) 1888 9582–5492
Fax.: +49 (0) 1888 9582–5427

tillmann.werner@bsi.bund.de

http://www.bsi.bund.de
http://www.cert-bund.de