Aktuelle Techniken und Methoden der verdeckten Kanäle

Angriff, Detektion und Prävention – wie weit ist die Forschung?
Chaos Singularity (CoSin), 25. Jun. 2011, Biel, CH

Steffen Wendzel
(www.wendzel.de)

Anmerkung zur veröffentlichten Version des Foliensatzes

Einige Slides der ursprünglichen Präsentation sind in diesen Folien leider nicht enthalten, da sie in eine andere Veröffentlichung einfließen (inkl. Copyright Transfer). Ich wollte sie zumindest im kleinen Rahmen in Biel dennoch vorstellen :-)

Ab spätestens Mitte Oktober ist dann auch der Rest verfügbar als:

S. Wendzel, J. Keller: *low-attention forwarding for mobile network covert channels*, In. Proc. Communication and Multimedia Security (CMS) 2011, Ghent Belgium, LNCS, Springer, 2011 → www.springerlink.com bzw. www.cms2011.net.

Eine Einführung in Mikroprotokolle, sowie ausführliches zu PHCC und PC findet sich in meiner online verfügbaren Diplomarbeit (pdf).

Das Buch „Tunneling und verdeckte Kanäle“ wird in Q4-2011 (spätestens jedoch Q1-2012) erscheinen und eine sehr ausführliche Einführung in das Thema der verdeckten Kanäle, deren Techniken und Methoden bieten.

Agenda

- Einführung und bekannte Techniken
- Protocol Channels & Protocol Hopping Covert Channels
- Vorstellung Präventions- und Detektionsmethoden
- OpenCCD

Teil I

Einführung

Covert Channel

- dt. *verdeckter Kanal*
- *Hacking-Community: Tunneling, teilweise auch „Side Channel“ (beides falsch)*
- B. Lampson: *A Note on the Confinement Problem, Comm. ACM 16(10),pp. 613–615,* 1973:

  „*not intended for information transfer at all*“
- Multilevel Security (NRU, NWD)
- Kein Seitenkanal

Relevanz

- Journalisten
- Unternehmen

  Fisk et. al.: 4 GByte/Tag möglicher, verdeckter Information Leakage eines Unternehmens über größere Webseite mit 500 Mio. Pkts/Tag möglich (2003).
- Würmer etc.
- evtl. Geheimdienste

Grobe Geschichte

- 1973: Lampson: Conf. Problem (bereits erwähnt)
- 1983: Kemmerer: SRM-Methodology
- 1984: Simmons: Prisoner's Problem
- 1987: Girling: Covert Channels in LANs
- 1991: Kemmerer+Porras: Covert Flow Trees
- 1991: Hu: Fuzzy Time (VAX Sec. Kernel)
- 1993: Kang+Moskowitz: Pump
- 1997: Rowland: Covert Channels in TCP/IP

Typische Techniken

- Packet Ordering
  - buffer=1,2,3,4, out=1,4,3,2
- Packet Timings
- Verwenden diverser Headerbestandteile

  etwa IPv4-Reserved-Bit, TTL oder DF-Flag
- Unzählige Bereiche in diversen Headern
  - Man denke nur an Plaintext-Protokolle (wie HTTP) mit diversen Headerbestandteilen
    - User Agent, Modification Time usw.

Lokale Covert Channels

- Beispiel: Filename

Zeitpunkt	Datei vorhanden?	Empfangenes Bit
1	Ja	1
2	Ja	1
3	Nein	0
4	Ja	1

- Weitere Möglichkeiten: CPU-Auslastung, Memory Consumption, Ausführungszeit eines Prozesses, Nice-Level, … , …, …

Passive Covert Channels

- J. Rutkowska (2004)
  - „NUSHU“ (21C3)
- ISN-basierter Covert Channel
  - Implementiert in Linux-Kernel
- Später durch Murdoch/Lewis detektiert
  - Dazu später mehr …

Teil II

Protocol Channels und
Protocol Hopping Covert Channels

Vorgeschichte

- Ziel: Detektion und Aufdeckung übertragenen Inhalts erschweren
- Erste Implementierung
  - „LOKI2“, Phrack Mag. Vol. 7/51, 1997
  - Autor: ”daemon9”

LOKI 2

- Tunneling via UDP und ICMP
  - Manueller Protokollwechsel via '/swapt'

”Swapping protocols is broken in everything but Linux. (...) This is why this feature is 'beta'.”

(Auszug aus dem LOKI2-Artikel im Phrack Mag.)

swendzel@steffenmobile: ~

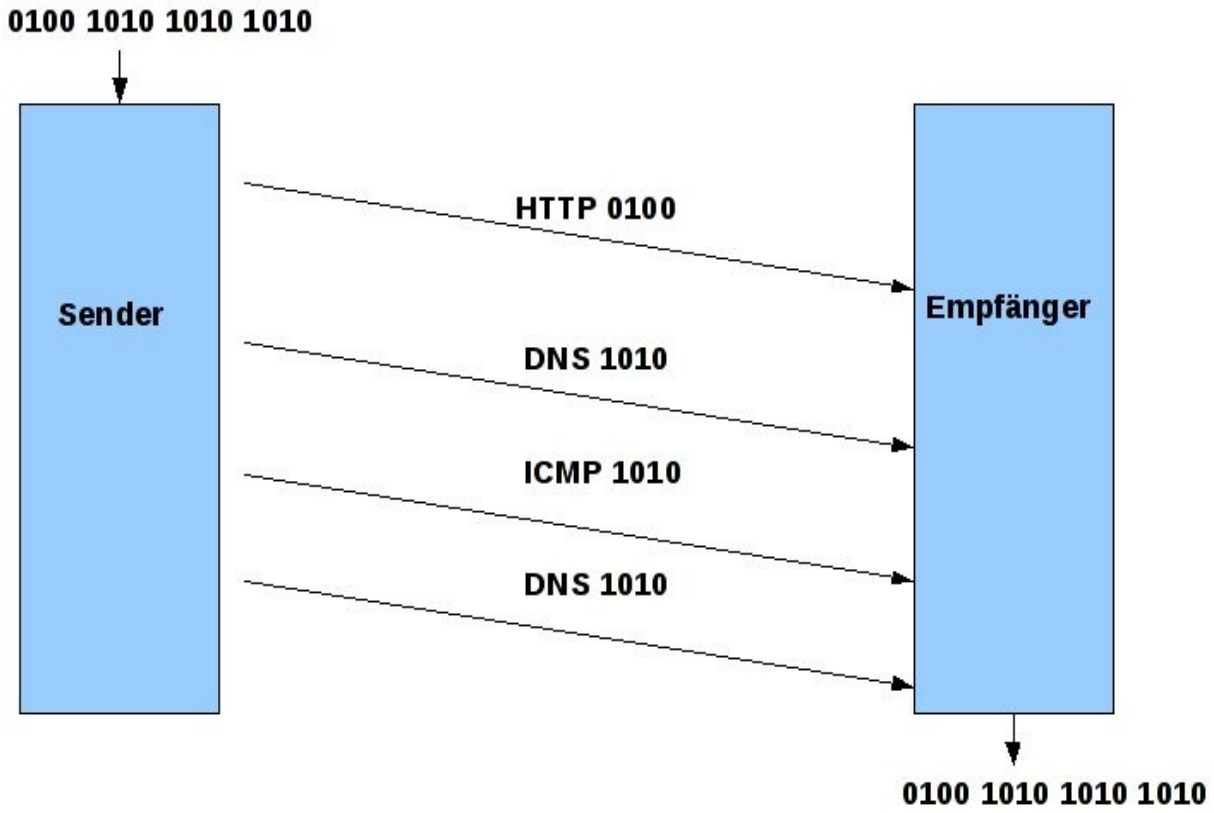
```
#define SWAP_T      "/swapt"        /* Swap protocols  */
...
    if (signal(SIGUSR1, swap_t) == SIG_ERR)
        err_exit(1, 1, verbose, L_MSG_SIGUSR1);
...

void d_parse(u_char *buf, pid_t pid, int ripsock) {
    ....
    if (!strncmp(buf, SWAP_T, sizeof(SWAP_T) - 1))
    {
        if (kill(getppid(), SIGUSR1))
            err_exit(1, 1, verbose,
                "[fatal] could not signal parent");
        clean_exit(0);
    }
...
void swap_t(int signo) {
    ...
    close(tsock);

    prot = (prot == IPPROTO_UDP) ? IPPROTO_ICMP : IPPROTO_UDP;
    if ((tsock = socket(AF_INET, SOCK_RAW, prot)) < 0)
        err_exit(1, 1, verbose, L_MSG_SOCKET);
    pprot = getprotobynumber(prot);
    sprintf(buf, "lokid: transport protocol changed to %s\n",
        pprot -> p_name);
    fprintf(stderr, "\n%s", buf);
                                                  24,1          Anfang
```

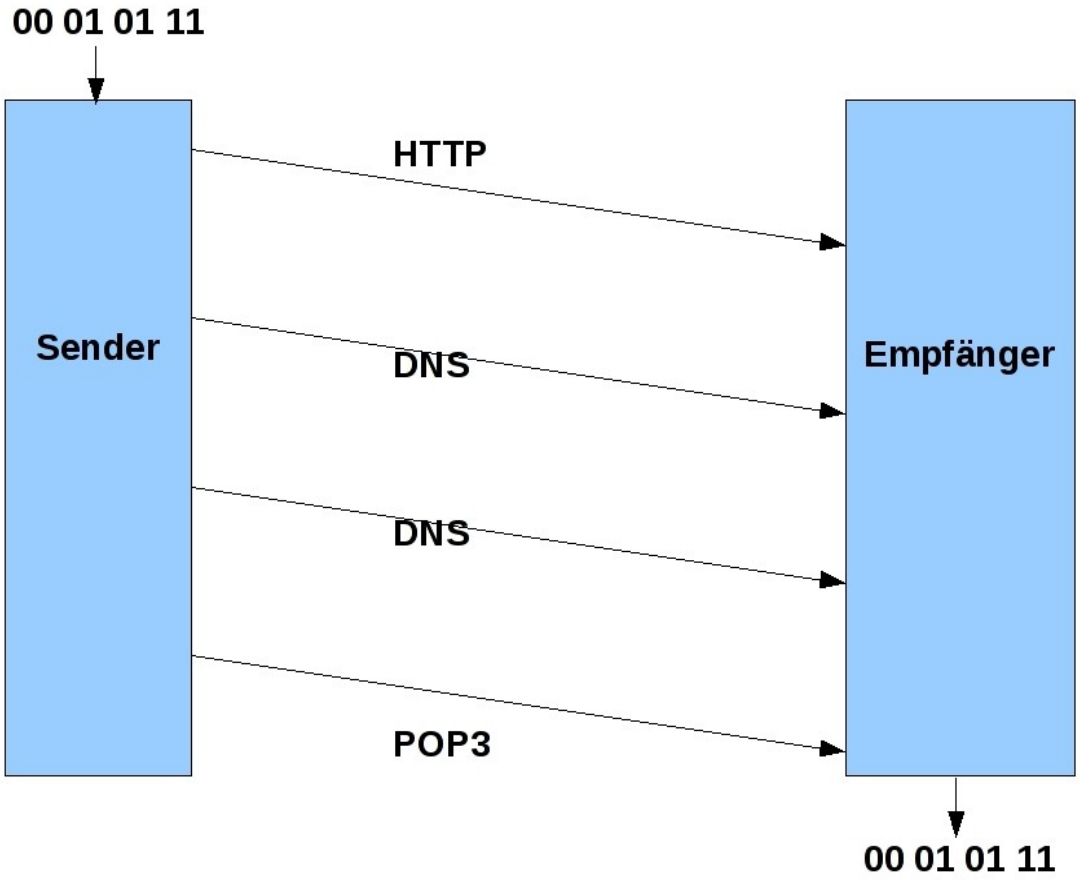
Protocol Hopping Covert Channels

- Sind Covert Storage Channels
- Wechsel des verwendeten Protokolls *jederzeit* und auch *zufällig* möglich. → Transparenz!
- Detektion und Analyse erschweren.
- Mikroprotokoll empfehlenswert (Sortierung)



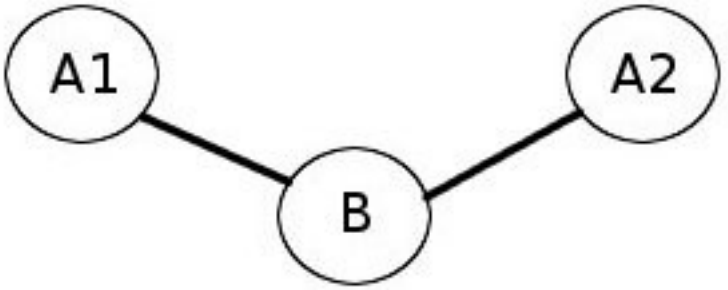
Protocol Channels

- Signalisierung der zu übertragenden Informationen *ausschließlich* durch die Information des verwendeten Protokolls.
- Nur statistisch detektierbar



Protocol Channels (2)

- Kein Platz für Korrekturinformationen
- Keine Bestätigungspakete sinnvoll möglich
  - Two Army Problem



- Störung durch eigentlichen Traffic → De-Sync!
- Fehlererkennung schwierig
  - PoC ”pct” nutzt Parity Bit

Bildquelle: Wikipedia: http://de.wikipedia.org/wiki/Datei:2_armeen_problem.jpg

Teil II

Präventions- und
Detektionsmethoden

Shared Resource Matrix Methodology

- Über welche Attribute können durch welche Prozeduren Covert Storage Channels etabliert werden?
  - Kemmerer, R. A.: *Shared resource matrix methodology: an approach to identifying storage and timing channels*, ACM Transactions on Computer Systems (TOCS), ACM, Vol. 1, Issue 3, pp. 256–277, 1983.

	read	**write**	**delete**	**create**
Dateiexistenz	R	R	R,M	R,M
Eigentümer	-	-	R	M
Dateiname	R	R	R	M
Dateigröße	R	M	M	M

McHugh: Erweiterung der SRM

- Versuch, Pseudo-Covert Channels zu sichten
  - 1. Unterscheidung: Ist Output/Input von/in eine Funktion durch den User eigentlich möglich?
  - 2. Visualisierung *unabhängiger Informationsflüsse* innerhalb von Funktionen („*Operation Splitting*“)
  - 3. Fallunterscheidungen in Informationsflüssen („*Guard Expansion*“)

	Operation A		
Attribut	Op1	Op2,Guard1	Op2,Guard2
a	R	-	-
b	-	M	M
c	-	R	-
User-In	R	R	R
User-Out	M	M	M

SRM/eSRM: Analyse

- Lt. Kemmerer anwendbar auf verschiedenste Bereiche des SDL, bspw. auf in englischer Sprache formulierte Requirements
- Bishop: „Nicht detailliert genug“ (etwa für Code-Analyse → CFT), da zu generalisiert
- Sowohl Kemmerer, als auch Bishop: Es können alle Storage Channels gefunden werden
- eSRM: Kann durch SRM detektierte Channels als nicht existent herauskristallisieren

Covert Flow Trees

- R. Kemmerer und P. Porras: „Covert Flow Trees: A Visual Approach to Analyzing Covert Storage Channels“, IEEE Tr. on SE, Vol. 17, No. 11, Nov. 1991.
- Verfahren zur Erkennung von Covert Storage Channels

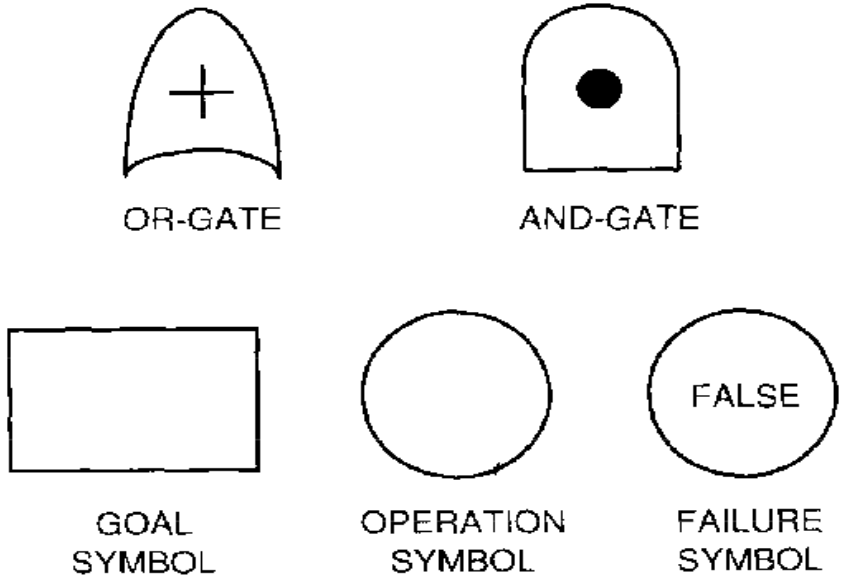
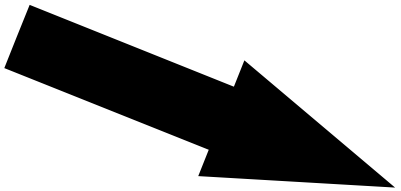


Bild aus M. Bishop: „*Computer Security. Art and Science*“

Covert Flow Trees (2)

- Code wird ähnlich analysiert, wie bei SRM:

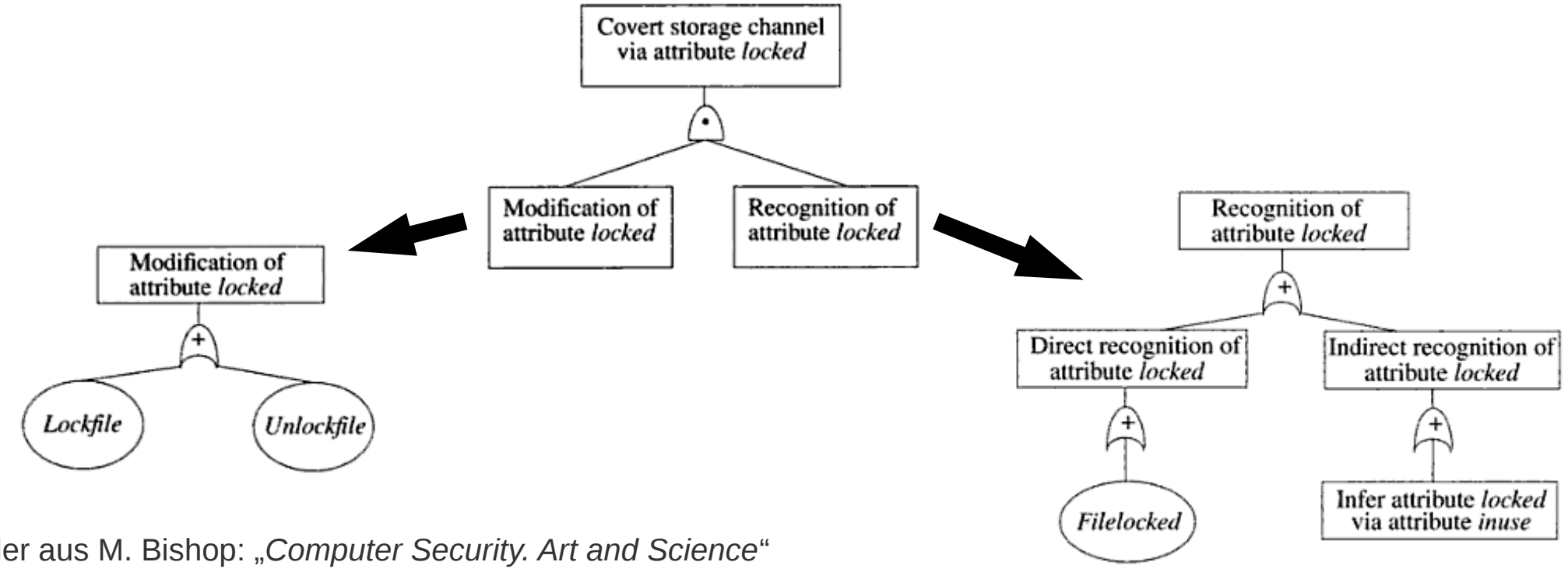
```
1  (* Datei locken, wenn noch nicht geöffnet u. gelockt *)
2  procedure Lockfile(f: file);
3  begin
4     if not f.locked and empty(f.inuse) then
5       f.locked := true
6  end;
7
8  (* Datei unlocken *)
9  procedure Unlockfile(f: file);
10 begin
11    if f.locked then
12      f.locked := false
13 end;
14
15 (* Prüfen, ob Datei gelockt ist *)
16 function Filelocked(f: file): boolean;
17 begin
18    Filelocked := f.locked;
19 end;
```



	Lockfile	Unlockfile	Filelocked
reference	locked,inuse	locked	locked
modify	locked	locked	-
return	-	-	locked

Covert Flow Trees (3)

	Lockfile	Unlockfile	Filelocked
reference	locked,inuse	locked	locked
modify	locked	locked	-
return	-	-	locked



Bilder aus M. Bishop: „*Computer Security. Art and Science*“

Covert Flow Trees (4)

- Information Flow-Listen erstellen
  - 1. Liste: Operationsfolgen, die Attribute modifizieren
  - 2. Liste: Operationssequenzen, die Modifizierungen prüfen können

    List 1 = ((Lockfile), (Unlockfile))

    List 2 = ((Filelocked))
- Damit ist der Informationsfluss für den Covert Storage Channel:
- Lockfile → Filelocked bzw. Unlockfile → Filelocked

Covert Flow Trees: Analyse

- Anwendungsgebiet auf Quellcode beschränkt
- Grafische Darstellung leicht interpretierbar
- Automatische Generierung des CFTs möglich
- Indirekte Informationsflüsse darstellbar!
- Kemmerer & Porras weisen darauf hin, dass Detektion von Timing Channels nicht geklärt ist (dazu keine weiteren Veröffentlichungen auffindbar).

Agat: Eliminierung von Timing Channels

- Agat: Bestimmter Input = bestimmte Timings (ähnlich Side-Channel, nur mit explizit gewünschtem Senden und MLS-Kontext)
- „*branches are observationally equivalent to the attacker*“ → no Timing Leaks

```
volvoValue := 0
i := 1
while (i<=DBsize) {
  let share := sharesDB[i].name in
  let value := lookupVal(share)*sharesDB[i].no in
    if (isVolvoShare(share))
      volvoValue := volvoValue+value;
    else
      skipAsn volvoValue (volvoValue+value);
    i := i + 1
}
```

Instruction Cache-Verhalten:

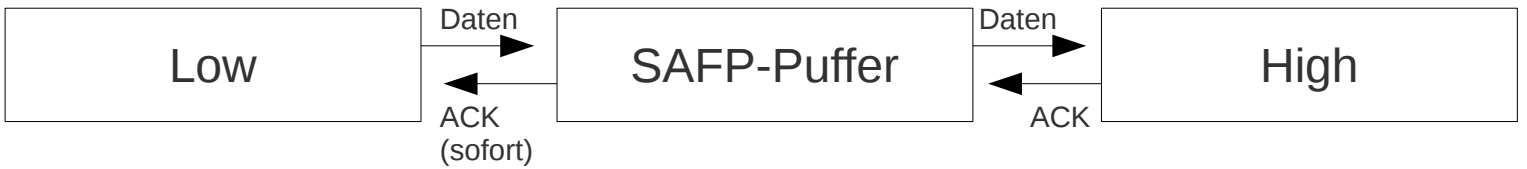
```
big_method();
if (h)
   big_method();
else
   big_method_LoSl();
```

Analyse von Agats Ansatz

- Agat: „einfach und realistisch“
  - Begründung: PoC-Implementierung (jedoch ohne Unterstützung für Java-Exceptions und Objekte)
- In welchem physikal. Speicher ist ein Code-Fragment gerade abgelegt? → Instruction Cache-Problematik
- (Myers,Liskov, 2000): Für prozedurale Sprachen geeignet.
- Performance-Einbußen durch Low-Slices

ACK-Filter, SAFP, Pump, Blind Write-Up, Upwards Channel, Quantized Pump

- Vielzahl ähnlicher Verfahren
  - Grundidee: Limitierung bzw. Vermeidung von ACKs
  - Vermeidung/Kapazitätsreduktion von Covert (Timing) Channels
  - Bekanntestes Beispiel: Die Pump (Kang u. Moskowitz '93)
    - Low-Data Caching → Flush; zudem ACK-Verzögerung
  - Umgesetzt in NRL Pump (mehrere Empfänger und Sender gleichzeitig möglich)
- Storage Channels werden ganz vermieden, Timing Channels werden unterbunden (Blind Write-Up und Upwards Channel) bzw. deren Kapazität wird reduziert (Pump).



Spurious Processes Approach

- Fadlalla,'96: Ansatz für Systeme mit TCB
- SPs sollen die Covert Channel-Prozesse stören: Jede Benutzung eines shared Attributes wird „vor-ausgeführt“.

P_1-**Verhalten**	P_1 erstellt Datei		P_1 erstellt Datei nicht	
SP-**Verhalten**	create()	create() +remove()	create()	create() +remove()
Resultat	Datei existiert	Datei existiert (den Syscalls von SP fehlen die Rechte)	Datei existiert	Datei existiert nicht
P_2-**Empfangswert**	1 (unsicher, ob P_1 oder SP die Datei erstellt hat)			0er Bit (sicher)

Spurious Processes Approach (Analyse)

- Verhindert einige Covert Storage Channels
- Schlecht beschrieben, nicht weiter von wissenschaftl. Community beachtet
- Diverse offensichtliche Sonderbedingungen nicht beschrieben (create → readdir() oder auch create() auf Systemdateien)
- Systemstabilität unter Umständen beeinträchtigt
- Performanceeinbußen (SPs können auch mehrfach ausgeführt werden; prob(SP) sinkt!)
- Nicht SMP-kompatibel!

Detektion in Geschäftsprozessen

- Accorsi & Wonnemann:
  - *Detective Information Flow Analysis for Business Processes*, 2009
  - *Informationsfluss-Mechanismen zur Zertifizierung von Cloud-basierten Geschäftsprozessen*, 2011
  - Leaking confidential business data

Detekt. in Geschäftsprozessen (2)

- InDico Framework mit IFnet-Modell (erweitertes, gefärbtes Petrinetz).
  - Input: Geschäftsprozess (etwa BPMN)
- Zertifikat
- Usage und Storage Conflict

Detekt. in Geschäftsprozessen (Analyse)

- Automatische Detektion von Interferenzen zwischen Prozessen möglich!
  - Fortschritt für die Detektion verdeckter Kanäle!
- Bereits für reale GPs (auch in Clouds) verwendet
- Bereits erkannte CCs konnten durch das InDico-Framework bestätigt werden

Normalisierung

- Typische Felder wie DF-Flag, Reserved Flag im IP-Header → decken nur bekannte Bits ab!
- Could-Start Problem
- Inconsistent TCP-Transmissions (Handley et.al.'01) → „nice“ || „root“
- pf scrub, Snort, norm

Bildquelle: Handley et. al. Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics, 2001.

Timing Channel Detection

- Diverse Verfahren zur Auswertung von Inter Packet Gaps (Inter-Arrival Times)
- Epsilon-Similarity (Cabuk et. al.)
- Compressibility (Cabuk et. al.)
- CC Detection gem. Berk et. al.

Fuzzy Time (Hu'91)

- Fuzzy Timings für virtuelle Maschinen im VAX-Security Kernel
- Event Time und Notification Time mit rand()-Verzögerung.

Passive ISN-Cov. Chan. Detection

- Murdoch/Lewis '05: Detektion von Rutkowskas TCP-ISN PCC „Nushu“
- Unterschiedliche Verteilung der ISNs
  - Current ISN → next ISN
- Unterschiedliche Wertebereiche der ISNs

Teil III

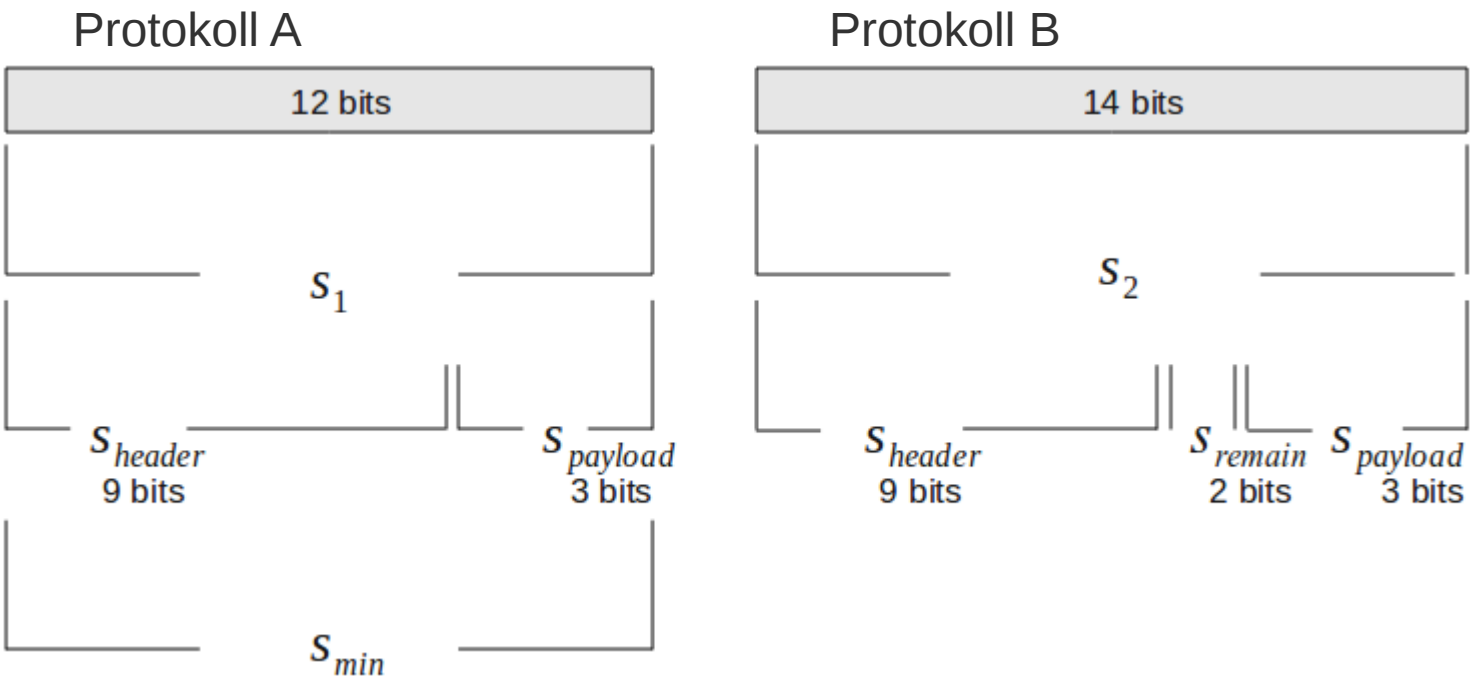
Mikroprotokolle in
verdeckten Kanälen

Anmerkungen zu Microprotokollen

- µP ist in den verdeckten Daten untergebracht
- Reliability, Transaktions-Flags, Identifier, usw.
- µP muss klein sein!

  → mehr Platz für Payload und weniger Traffic
- Detektion möglich (etwa via ID-Feld), daher zusätzlich zum eigentlichen CC: Obfuscation

Platzbedarf für Mikroprotokolle



Yaochkin et. al.

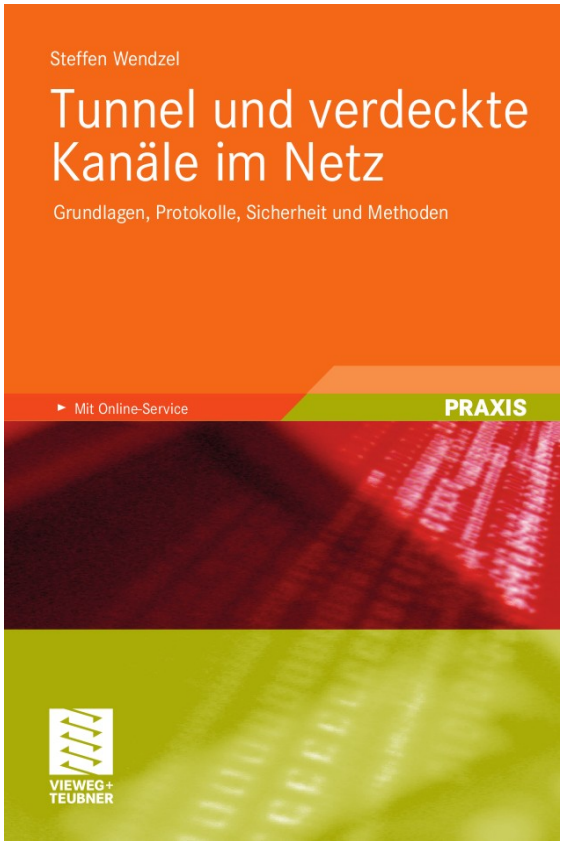
- Adaptive Covert Channels
- Zwei-Phasen-Kommunikationsprotokoll durch Yarochkin et. al. eingeführt.
- 1) Network Environment Learning Phase
  - Protokolle, die nicht geroutet werden oder
  - geblockt werden: abziehen
- 2) Communication Phase
  - Protokoll-Schnittmengen bilden
  - Kommunikation wird abgewickelt
  - Parallel läuft NEL-Phase

Zusammenfassung

- Relevanz
- PHCC: Verteilung der geheimen Daten
- Protocol Channel (PHCC ohne Storage-Attribute)
- Es gibt diverse Methoden zur Prävention und Detektion, hauptsächlich für Storage Channels
- Mikroprotokolle → www.cms2011.net
  - Reliability
  - Kontrolle des Channels

Vorschau

- Tunnel und verdeckte Kanäle im Netz
  - Vieweg+Teubner, 2012
  - www.linux-openbook.de
  - Noch dieses Jahr:
    - Linux. Das umfassende Handbuch, 5. Aufl.
      >1300 Seiten 4 free als Download!
    - Einstieg in Linux, 5. Auf.



- Open Covert Channel Detector
  - Erstes Covert Channel-NIDS
  - www.openccd.org
  - Call for Developers!