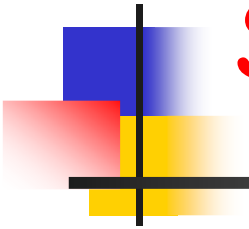
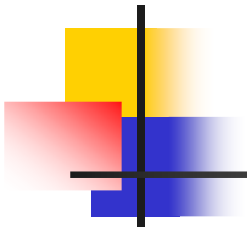


Detecting Backdoors and Stepping Stones

Yin Zhang
Cornell University
yzhang@CS.Cornell.EDU

Vern Paxson
ACIRI/LBNL
vern@aciri.org

9th USENIX Security Symposium
Denver, CO, August 2000

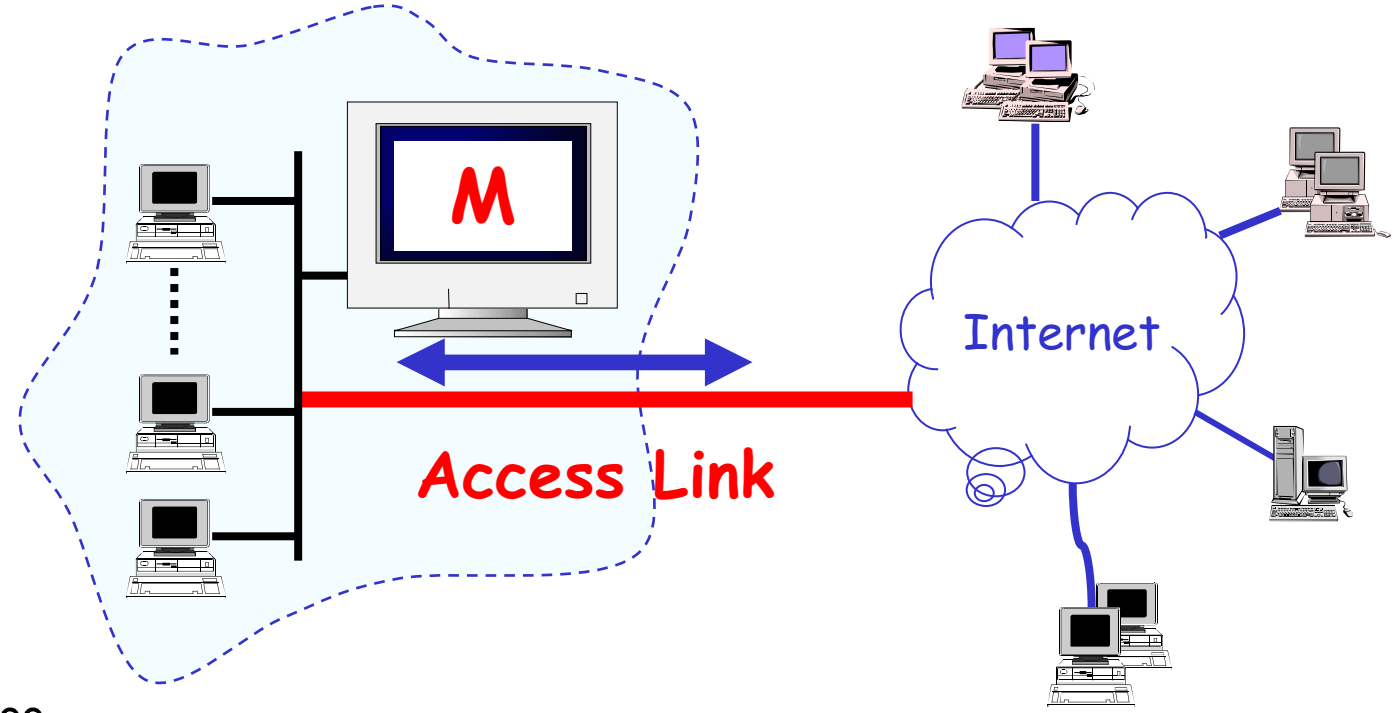


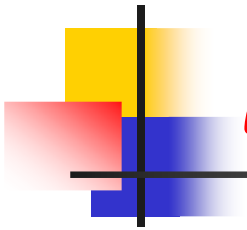
Backdoors & Stepping Stones

- Two big headaches for intrusion detection
  - Ease of returning to a compromised system
  - Ease of hiding attacker's identity
- Backdoors
  - Standard service on non-standard port, or on standard port associated with different service
- Stepping stones
  - Compromised, intermediary hosts used during attacks to hide attacker's identity

Targeted Environment

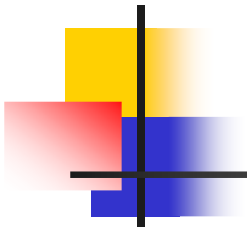
- Monitor captures inbound/outbound traffic
- Assume single ingress/egress point for stepping stone detection





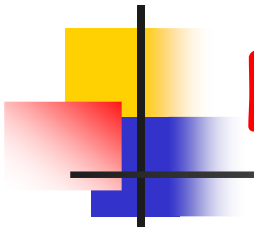
Methodology

- Design space
- Trace investigation
- General algorithms
- Refinements
- Trace-based evaluation:
  - FP, FN, efficiency



Backdoor Methodology

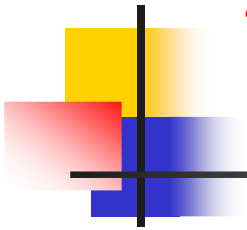
- Design space
  - A lot in common
- General algorithm: Pkt size + timing
  - doesn't require content
- Protocol specific algorithms
  - Stateless filter → highly efficient
- Performance Evaluation



Design Space

- Open vs. evasive attackers
  - raising the bar, "Arms race"
- Passive vs. active monitoring
- Accuracy: FP vs. FN
- Content vs. timing
  - Timing: can be very cheap, robust against encryption
- Real-time vs. off-line analysis
  - Off-line algorithms: full stream reassembly, baseline for how good you might do
- Filtering
  - Lots skipped in kernel → huge reduction in load

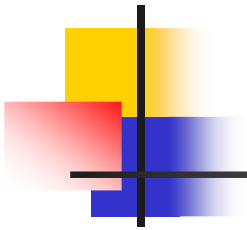
A General Algorithm for Detecting Interactive Backdoors



- Leveraging large number of small pkts
  - $(S - G - 1) / N \geq 0.2$
    - S: number of small packets
    - G: number of gaps in small packets
    - N: total number of packets
- Leveraging large number of long pauses
  - $\#interarrivals \in [10ms, 2s] / \#interarrivals \geq 0.2$
    - Almost the same performance when 2 sec → 100 sec.
- Filtering
  - Only small packets (e.g. with ≤ 20 bytes payload)
  - Need some guesses for G and N

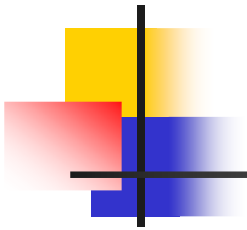
Protocol-Specific Algorithms

Backdoor	Optimal Algorithm	Stateless Algorithm
SSH	Ssh-sig, ssh-len	Ssh-sig-filter
Rlogin	Rlogin-sig	Rlogin-sig-filter
Telnet	Telnet-sig	Telnet-sig-filter
FTP/SMTP	Ftp-sig	Ftp-sig-filter
Root shell	Root-sig	Root-sig-filter
Napster	Napster-sig	Napster-sig-filter
Gnutella	Gnutella-sig	Gnutella-sig-filter



Detecting SSH

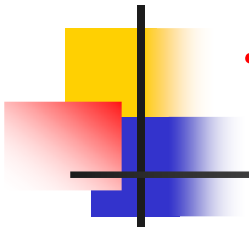
- Ssh-sig
  - Signature: SSH version string `^SSH-[12]\.'
- Ssh-len (mainly for partial connections)
  - Interactive according to the general algorithm
  - Most packets have 8N ($N \geq 2$) bytes payload, or most packets have (8N+4) bytes payload
- Ssh-sig-filter
  - Implemented by a stateless tcpdump filter
    - `tcp[(tcp[12]>>2):4] = 0x5353482D and (tcp[((tcp[12]>>2)+4):2] = 0x312E or tcp[((tcp[12]>>2)+4):2] = 0x322E)`



Detecting Others

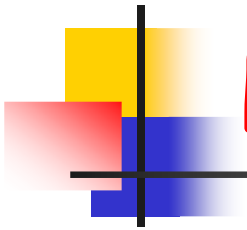
Backdoor	Signature	Equivalent Pattern
Rlogin	Username terminal dialog, <NUL> terminated	`'\x00$'`
Telnet	Option negotiation	`'^\xFF[\xFB-\xFE]'`
FTP/SMTP	Server status codes	`'^(220|421)[ -]'`
Napster	SEND/GET directives	`'^(SEND|GET)$'`
Gnutella	Connection negotiation	`'^GNUTELLA '`
Root shell *	Root shell prompt	`'^# '`

* A hack, but works surprisingly well



Trace Descriptions

- ssh.trace (194 MB, 380K pkts, 905 conns)
  - A half hour snapshot of SSH traffic at UCB
- lbnl.mix1.trace (54MB, 134K packets, 4.6K conns)
  lbnl.mix2.trace (421MB, 863K packets, 14.7K conns)
  - 1 hour of aggregate traffic at LBNL with high volume protocols filtered out
- lbnl.inter.trace (389MB, 3.5M packets, 5.5K conns)
  - 1 day's worth of Telnet/Rlogin traffic at LBNL



Performance Evaluation

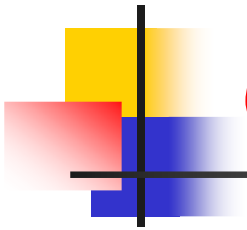
Algorithm	FP	FN	% bytes captured
Ssh-sig	0/16,938	0/546	NA
Ssh-sig-filter	0/16,938	0/546	0.057%
Ssh-len	5/16,938	NA	NA
Rlogin-sig	0/17,306	0/175	NA
Rlogin-sig-filter	4/17,306	0/175	1.6%

Performance Evaluation (con't)

Algorithm	FP	FN	% bytes captured
telnet-sig	0/12,708	18*/1,526	NA
telnet-sig-filter	0/12,708	18/1,526	0.15%
ftp-sig	0/20,135	29**/5,629	NA
ftp-sig-filter	0/20,135	29/5,629	0.12%
General Algo.	12/12,000+	22/1,450	NA

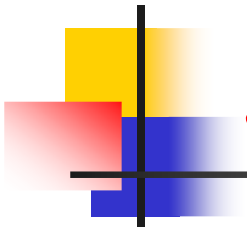
* 17 involve the same passwordless catalog server w/o any option negotiation; the 18th is HTTP/1.1 on port 23 → not FN

** Most are partial connections w/o the initial dialog



Operational Experience

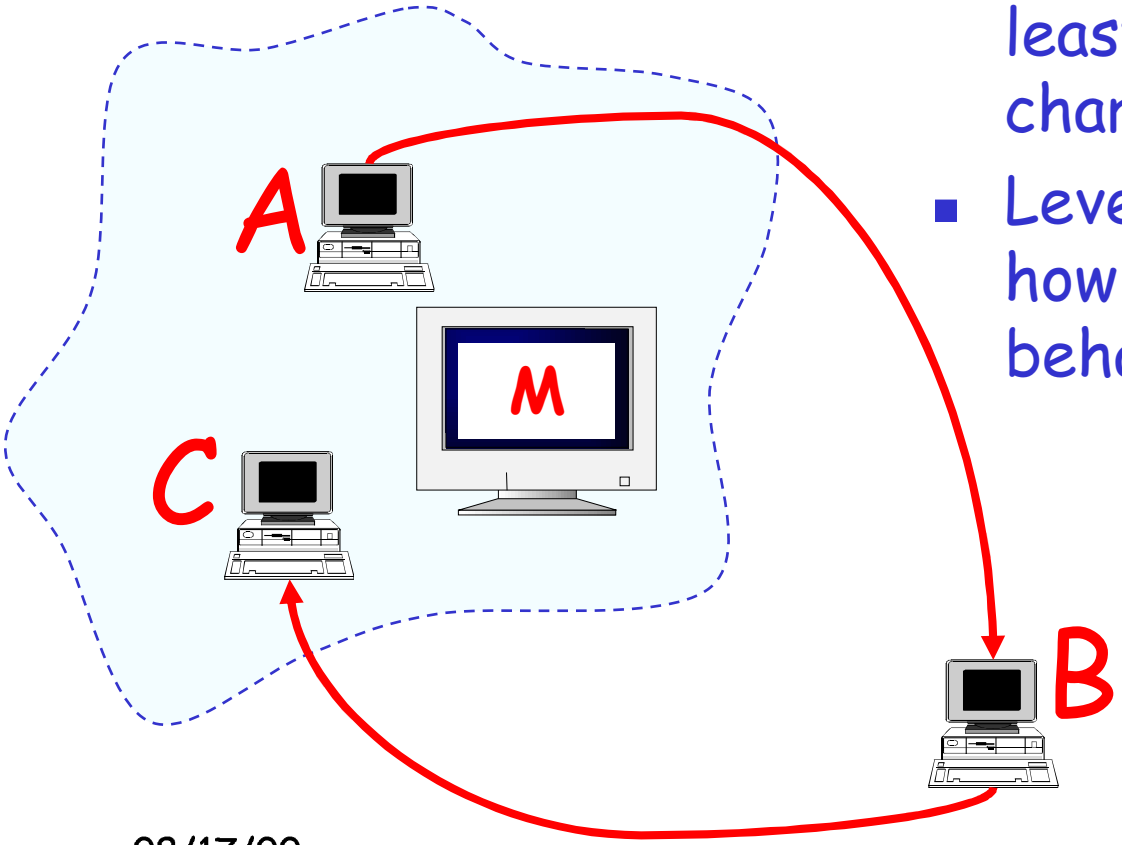
- Root-sig-filter: dirt cheap, but strikingly powerful
  - Finds su's
  - Finds 437 root backdoors at 291 sites in 24 hours from Berkeley
- SSH detectors find SSH servers on various ports
  - 80 (HTTP); 110 (POP); 32; 44320-44327; variants of 22 (222, 922, 2222, …)
- Napster detectors find Napster server on port 21 (FTP), and plenty of others!
- Large number of legitimate backdoors require refined policy scripts



Stepping Stone Methodology

- Design space
  - A lot in common
- A timing-based algorithm
  - Doesn't require content
- Calibration algorithms
  - Mainly used as baseline algorithms
  - Efficient ones are also used for production use
- Performance Evaluation

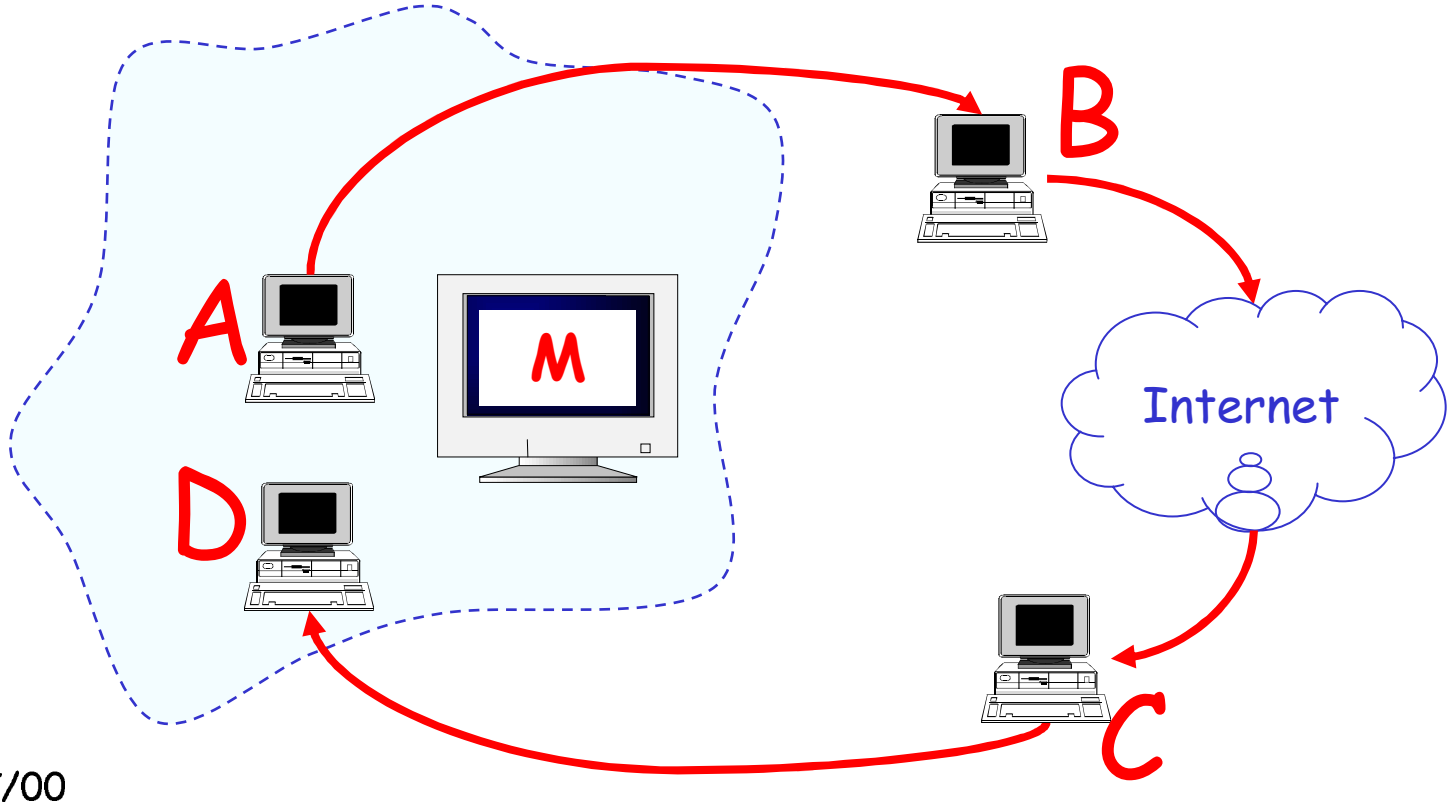
General Principles

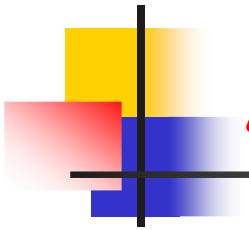


- Find invariant or at least highly correlated characteristics
- Leverage particulars of how interactive traffic behaves

Additional Design Space

- Direct vs. indirect stepping stones, i.e. "A-B-C" vs. "A-B … C-D"

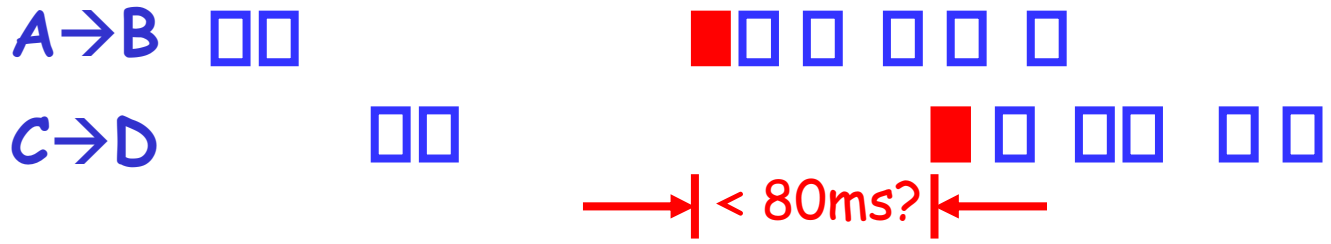




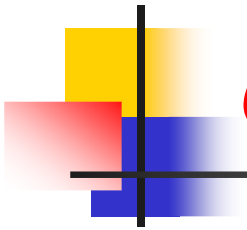
Additional Design Space (con't)

- Whether to analyze content ?
  - Content-based fingerprinting [SH95]
  - Pro: natural; Con: cost, opportunity.
- Minimize state for connection pairs
  - N^2 memory explosion

Timing Correlation When OFF Periods End

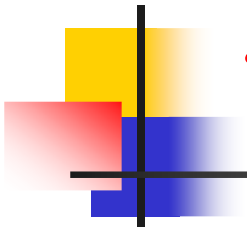


- Only consider the end of OFF periods
  - OFF period: no activity for ≥ 0.5 sec
  - Immensely reduces analysis possibilities!
- Two OFF periods considered correlated, if their ending times differ by < 80ms.
- Detection criteria
  - #coincidences / #OFF_periods
  - #consecutive_coincidences
  - #consecutive_coincidences / #OFF_periods



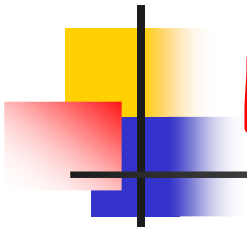
Calibration Algorithms

- Brute-force one-time calibration
  - Extract the aggregate Telnet/Rlogin output
  - Find connections with similar content by looking at lines in common using standard Unix utilities
  - Identify stepping stones with additional manual inspection
- Two Unix-centric hacks: Looking for
  - propagated **`$DISPLAY`**
  - propagated status line in the login dialog.
    - **`Last login: Fri Jun 18 12:56:58 from host.x.y.z.com`**



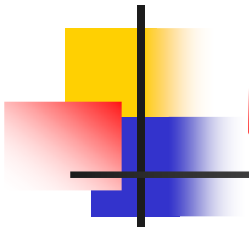
Trace Descriptions

- Lbnl-telnet.trace
  - 1 day's worth of telnet/rlogin traffic at LBNL
  - 120 MB, 1.5M pkts, 3,831 conns
  - 21 stepping stones
- Ucb-telnet.trace
  - 5.5 hours' worth of telnet/rlogin traffic at UCB
  - 390 MB, 5M pkts, 7,319 conns
  - ~79 stepping stones



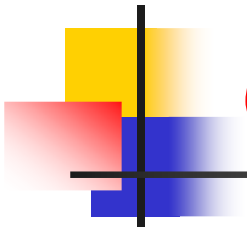
Performance Evaluation

- Accuracy: Very low false positive/negative ratios
  - Lbnl-telnet.trace: FP = 0, FN = 2/21
  - Ucb-telnet.trace: FP = 0, FN = 5/79
    - Brute-force scheme missed 32
- Efficiency: capable of real-time detection
  - 1.1 real-time minutes for lbnl-telnet.trace
  - 24 real-time minutes for ucb-telnet.trace
- Impact of different control parameters
  - Current parameter settings are fairly optimal
  - Considerable room exists for varying the parameters in response to certain evasion threats



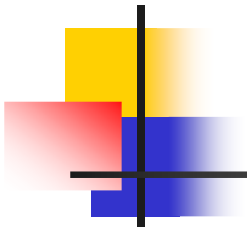
Failures

- Excessively small stepping stones
  - Limits attackers to a few keystrokes
- Message broadcast applications lead to non-stepping-stone correlation
  - Can filter out
- Phase-drift in periodic traffic leads to false coincidences
  - Can filter out



Operational Experience

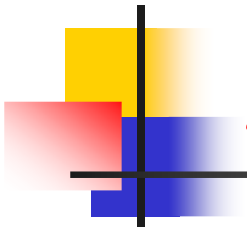
- Nifty algorithm, clearly useful in some circumstances
- Large number of legitimate stepping stones require refined policy scripts
- An unanticipated security bonus
  - Exposed passphrase due to clear-text protocol upstream and encrypted protocol downstream
  - Unfortunately, this happens all too often ☹



Future Directions

- Backdoor detection
  - Combining general algorithm with protocol-specific algorithms
  - Other protocols, e.g., BackOrifice
- Stepping stone detection
  - Detecting non-interactive stepping stones, e.g. "relays", and "slaves".
- All sorts of evasion possible --

  "let the arms race begin"



Acknowledgements

- Ken Lindahl, Cliff Frost
- Stuart Staniford-Chen, Felix Wu
- Mark Handley, Tara Whalen, and anonymous reviewers