Inference and Analysis of Formal Models of Protocols

Domagoj Babić

Chia Yuan Cho Richard Shin Dawn Song



University of California, Berkeley
Computer Science Division

Outline

1. Motivation
   - Problem Definition
   - Applications
   - Previous Work
2. Inference of Complete State Machines
   - Basic Ideas
   - Improvements to the $L*$ Inference Algorithm
   - MegaD Botnet Analysis Results

Automatic Inference of Formal Protocol Models

Protocol is a set of rules defining:

- Data representation (message format)
- Protocol state-machine

Automatic Inference of Formal Protocol Models

Protocol is a set of rules defining:

- Data representation (message format)
- Protocol state-machine

Automatic Inference of Formal Protocol Models

Protocol is a set of rules defining:

- Data representation (message format)
- Protocol state-machine

Protocol formal models:

- Context-sensitive
- Context-free
- Regular (finite-state machines)

Automatic Inference of Formal Protocol Models

Protocol is a set of rules defining:

- Data representation (message format)
- Protocol state-machine

Protocol formal models:

- Context-sensitive
- Context-free
- Regular (finite-state machines)

Automatic Inference of Formal Protocol Models

Protocol is a set of rules defining:

- Data representation (message format)
- Protocol state-machine

Protocol formal models:

- Context-sensitive
- Context-free
- Regular (finite-state machines)

Types of inference:

- Offline (passive) — Only from observed behavior
- Online (active) — From interactive queries

Automatic Inference of Formal Protocol Models

Protocol is a set of rules defining:

- Data representation (message format)
- Protocol state-machine

Protocol formal models:

- Context-sensitive
- Context-free
- Regular (finite-state machines)

Types of inference:

- Offline (passive) — Only from observed behavior
- Online (active) — From interactive queries

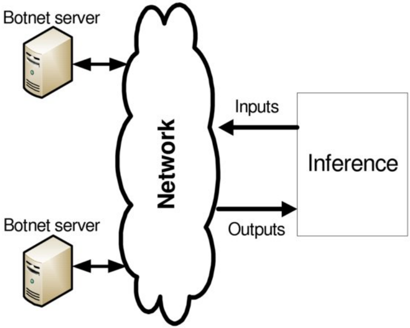
Automatic Inference of Formal Protocol Models

Our goal

Automatic black-box online inference of finite-state protocol models.

Our assumptions

- Resettability of the protocol state-machine
- Known message format
- Known encryption



Applications of Protocol Inference

1. Reverse-engineering of proprietary/classified protocols (e.g., botnet protocols)
   - Identification of critical links
   - Identification of background communication channels

Applications of Protocol Inference

1. Reverse-engineering of proprietary/classified protocols (e.g., botnet protocols)
   - Identification of critical links
   - Identification of background communication channels
2. Verification/testing of protocol implementations
   - Identification of design flaws (model checking)
   - Identification of implementation differences (equivalence checking)
   - Test generation for fuzzing the implementation

Applications of Protocol Inference

1. Reverse-engineering of proprietary/classified protocols (e.g., botnet protocols)
   - Identification of critical links
   - Identification of background communication channels
2. Verification/testing of protocol implementations
   - Identification of design flaws (model checking)
   - Identification of implementation differences (equivalence checking)
   - Test generation for fuzzing the implementation
3. Fingerprinting implementations

Applications of Protocol Inference

1. Reverse-engineering of proprietary/classified protocols (e.g., botnet protocols)
   - Identification of critical links
   - Identification of background communication channels
2. Verification/testing of protocol implementations
   - Identification of design flaws (model checking)
   - Identification of implementation differences (equivalence checking)
   - Test generation for fuzzing the implementation
3. Fingerprinting implementations

Comparison with Previous Protocol Inference Work

Some basic notions...

- **Mealy machines**: a type of a finite state-machine, each transition produces an output, more suitable for modeling reactive systems like protocols
- **Moore machines**: a type of a finite state-machine, output is defined by states
- **Complete models**: no state-machine transitions are missing
- **Minimal models**: the smallest possible number of states

Reference	Formalism	Online	Complete	Minimal
(Hsu, Shu, and Lee, 2008)	Mealy	-	-	-
(Comparetti et al., 2009)	Moore	-	-	+
Our work	Mealy	+	+	+

Comparison with Previous Protocol Inference Work

Some basic notions...

Mealy machines a type of a finite state-machine, each transition produces an output, more suitable for modeling reactive systems like protocols

Moore machines a type of a finite state-machine, output is defined by states

Complete models no state-machine transitions are missing

Minimal models the smallest possible number of states

Reference	Formalism	Online	Complete	Minimal
(Hsu, Shu, and Lee, 2008)	Mealy	-	-	-
(Comparetti et al., 2009)	Moore	-	-	+
Our work	Mealy	+	+	+

- Completeness and minimality important for model analysis!

Automatic Inference Flow: The First Attempt

1. Reverse-engineer message format (automatic with Polyglot)
2. Abstract messages with a finite alphabet (manual)
3. Protocol inference (automatic with Shahbaz and Groz's $L*$ algorithm)
4. Sampling to check conjectured model

Automatic Inference Flow: The First Attempt

1. Reverse-engineer message format (automatic with Polyglot)
2. Abstract messages with a finite alphabet (manual)
3. Protocol inference (automatic with Shahbaz and Groz's $L*$ algorithm)
4. Sampling to check conjectured model

$L*$ inference challenges:

- Experiment anonymization for inference of botnet protocols (Tor)
- Tor network & server delay (6.8 sec/message)
- Inference of 17-state machine required 56,716 messages (4.46 days of computation)

A Bird's Eye View of $L*$

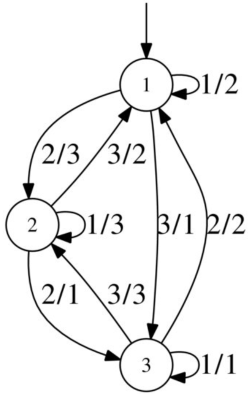
- $L*$ builds an observation table
- State-machine can be read from the table

Observation Table

Σ_I , input alphabet
Σ_O , output alphabet
(S, E, T) , observation table
S , prefix-closed subset of Σ_I^*
E , suffix-closed subset of Σ_I^+
$T : (S \cup S \cdot \Sigma_I) \times E \longrightarrow \Sigma_O^+$, map

		E		
		1	2	3
S	ϵ	2	3	1
	2	3	1	2
	3	1	2	3
$S \cdot \Sigma_I$	1	2	3	1
	$2 \cdot 1$	3	1	2
	$2 \cdot 2$	1	2	3
	$2 \cdot 3$	2	3	1
	$3 \cdot 1$	1	2	3
	$3 \cdot 2$	2	3	1
	$3 \cdot 3$	3	1	2

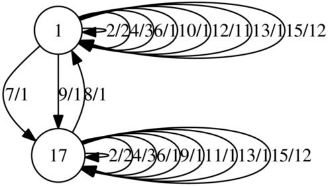
A Bird's Eye View of $L*$



		E: 1	2	3
S	ϵ	2	3	1
	2	3	1	2
	3	1	2	3
$S \cdot \Sigma_I$	1	2	3	1
	$2 \cdot 1$	3	1	2
	$2 \cdot 2$	1	2	3
	$2 \cdot 3$	2	3	1
	$3 \cdot 1$	1	2	3
	$3 \cdot 2$	2	3	1
	$3 \cdot 3$	3	1	2

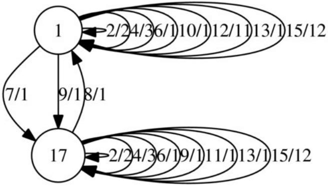
Predicting Responses to Sequences of Input Msg.

- Many messages serve only one purpose ⇒
- Many self-loops ⇒
- Use loop-free responses to predict those with loops



Predicting Responses to Sequences of Input Msg.

- Many messages serve only one purpose ⇒
- Many self-loops ⇒
- Use loop-free responses to predict those with loops

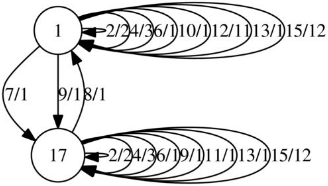


Theorem

Strings in the S part of the table are loop free.

Predicting Responses to Sequences of Input Msg.

- Many messages serve only one purpose ⇒
- Many self-loops ⇒
- Use loop-free responses to predict those with loops



Theorem

Strings in the S part of the table are loop free.

1. Compute the set of symbols D used in S
2. Compute a projection of any sequence of input messages onto D
3. Use the projection as a prediction
4. When the algorithm converges, test with random sequences (error: $\epsilon = 10^{-2}$, confidence: $\gamma = 10^{-6}$)
5. If prediction error, backtrack

Response Prediction Results

	MegaD C&C		MegaD SMTP		Postfix SMTP	
	Queries	Msgs	Queries	Msgs	Queries	Msgs
Basic $L*$	10,978	56,716	1,190	4,522	1,386	5,894
RESTR	-8,024	-42,872	-294	-980	-476	-1764
STAT	-1,456	-7,514	-22	-88	-0	-0
Backtrack	+24	+76	+24	+90	+56	+252
Total	1,522	6,406	898	3,544	966	4,382
Reduction	-86.1%	-88.7%	-24.5%	-21.6%	-30.3%	-25.7%
Accuracy	99.7%	99.9%	92.4%	97.8%	88.2%	96.8%

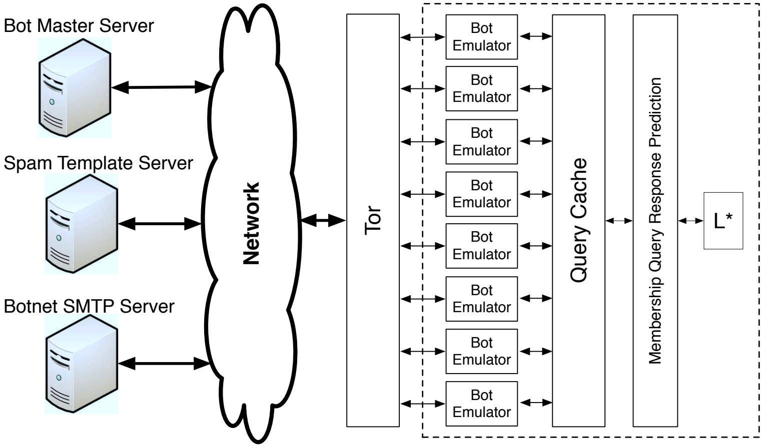
Query — sequence of messages
RESTR — prediction (restriction based)
STAT — prediction (statistical, not described in the talk)
Backtrack — cost of incorrect predictions (backtracking)
Accuracy — prediction accuracy

Response Prediction Results

	MegaD C&C		MegaD SMTP		Postfix SMTP	
	Queries	Msgs	Queries	Msgs	Queries	Msgs
Basic $L*$	10,978	56,716	1,190	4,522	1,386	5,894
RESTR	-8,024	-42,872	-294	-980	-476	-1764
STAT	-1,456	-7,514	-22	-88	-0	-0
Backtrack	+24	+76	+24	+90	+56	+252
Total	1,522	6,406	898	3,544	966	4,382
Reduction	-86.1%	-88.7%	-24.5%	-21.6%	-30.3%	-25.7%
Accuracy	99.7%	99.9%	92.4%	97.8%	88.2%	96.8%

Query — sequence of messages
RESTR — prediction (restriction based)
STAT — prediction (statistical, not described in the talk)
Backtrack — cost of incorrect predictions (backtracking)
Accuracy — prediction accuracy

Architecture of the Inference System



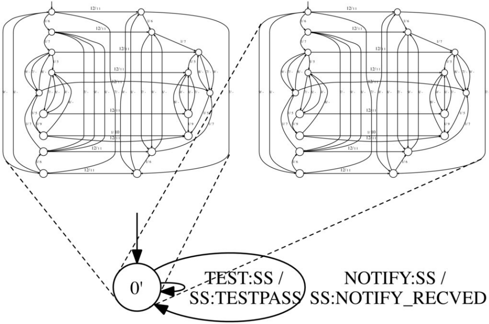
Identification of Critical Links of the MegaD Botnet

- Min-cutset algorithms usually used
- Taking down any botnet server prevents bots from spamming

Identification of Critical Links of the MegaD Botnet

- Min-cutset algorithms usually used
- Taking down any botnet server prevents bots from spamming

1. Infer models for two different pools of bots
2. Compute intersection of models

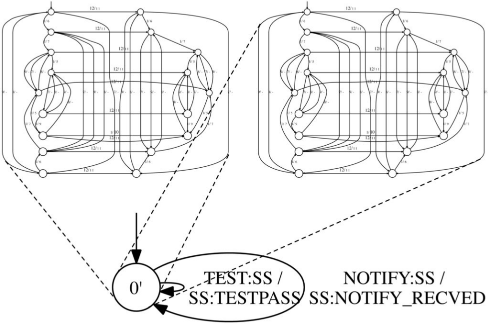


Identification of Critical Links of the MegaD Botnet

- Min-cutset algorithms usually used
- Taking down any botnet server prevents bots from spamming

1. Infer models for two different pools of bots
2. Compute intersection of models

- Critical resource is the shared SMTP server!
- Previous attempts to defeat MegaD focused on the master server

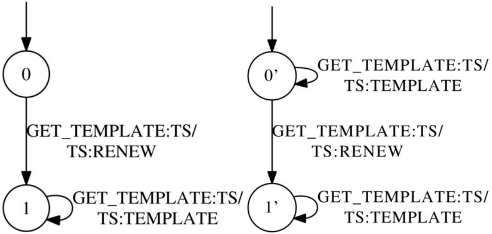


Identification of Background Channels

1. Infer a protocol model M
2. For each server
   - Restrict alphabet to server's alphabet A
   - Infer a model M_A over A
   - Project M onto A, obtaining M_A^p
3. Compare M_A and M_A^p
4. Difference proves master and template servers talk to each other

Identification of Background Channels

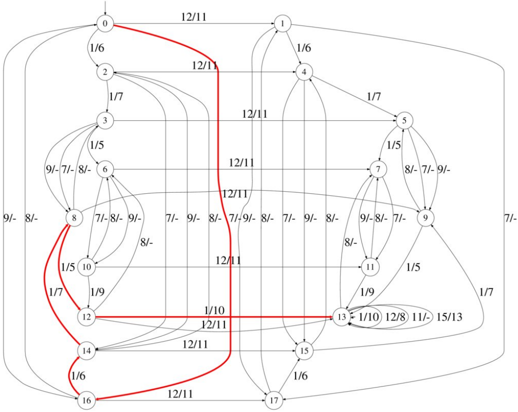
1. Infer a protocol model M
2. For each server
   - Restrict alphabet to server's alphabet A
   - Infer a model M_A over A
   - Project M onto A, obtaining M_A^p
3. Compare M_A and M_A^p
4. Difference proves master and template servers talk to each other



The right figure (with an extra edge) is M_A^p (projection of the protocol model on the template server alphabet).

Design Flaws in the MegaD Protocol

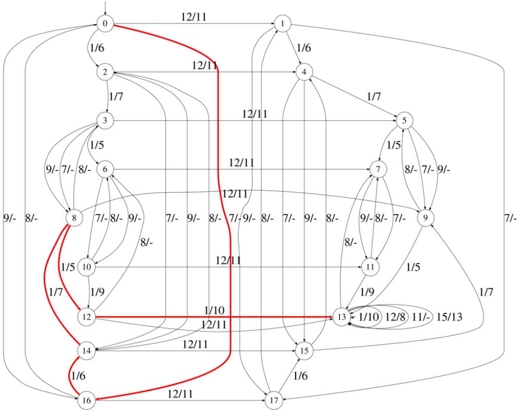
- Normal bot execution
  $0 \rightarrow 16 \rightarrow 14 \rightarrow 8 \rightarrow 12 \rightarrow 13$
- In state 13, bot sends
  - GET_COMMAND (to master s.), gets identifier
  - GET_TEMPLATE (to template s.), gets spam templates



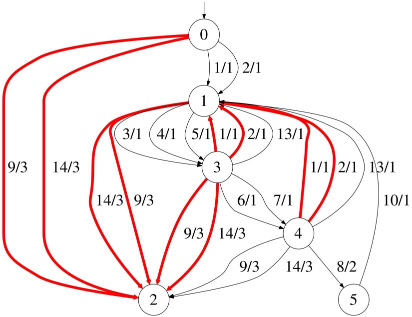
Design Flaws in the MegaD Protocol

- Normal bot execution
  $0 \rightarrow 16 \rightarrow 14 \rightarrow 8 \rightarrow 12 \rightarrow 13$
- In state 13, bot sends
  - GET_COMMAND (to master s.), gets identifier
  - GET_TEMPLATE (to template s.), gets spam templates

1. Contact the template server directly ($0 \rightarrow 1$)
2. Use random identifier
3. Get fresh templates!
4. Update spam filtering rules before spam hits the web!



Identification of Implementation Differences



Postfix SMTP 2.5.5
Red edges not found by Prospex (Comparetti et al., 2009).

- Postfix deviates from the standard
- MegaD SMTP deviates from both the standard and Postfix
- $\Rightarrow$ Precise fingerprinting

Summary and Future Work

Summary

- First complete finite-state machine protocol inference
- Works in the real-world setting
- Highly effective response prediction (88% reduction of # of messages)
- Parallelization and caching (4.85X speedup)
- New knowledge gained about MegaD

Future Work

- More expressive formal models
- Automatic abstraction
- Stateful fingerprinting
- More applications?

For Further Reading

- P. M. Comparetti, G. Wondracek, C. Kruegel, and E. Kirda.
  Prospex: Protocol specification extraction.
  In *SP'09: Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, pages 110–125, Washington, DC, USA, 2009. IEEE Computer Society.
- T. Hsu, G. Shu, and D. Lee.
  A model-based approach to security flaw detection of network protocol implementation.
  In *ICNP'08: Proceedings of the 15th IEEE International Conference on Network Protocols*, pages 114–123, Oct 2008.
- A. Gupta, K. L. McMillan, and Z. Fu.
  Automated assumption generation for compositional verification.
  Form. Methods Syst. Des., 32(3):285–301, 2008.
- M. Shahbaz and R. Groz.
  Inferring Mealy machines.
  In *FM'09: Proceedings of the 2nd World Congress on Formal Methods*, pages 207–222, Berlin, Heidelberg, 2009. Springer.