Tunnel Hunter: Detecting Application-Layer Tunnels with Statistical Fingerprinting

Presented by ZhenlongYuan

Motivation

- Several techniques tunnels one application-layer protocol into another one.
  - encapsulate all outgoing traffic into semantically valid HTTP requests
  - tunnel TCP traffic through an SSH connection by means of cryptography.
  - tunnel non-legitimate traffic through the network boundary.

Overview of Tunneling Techniques

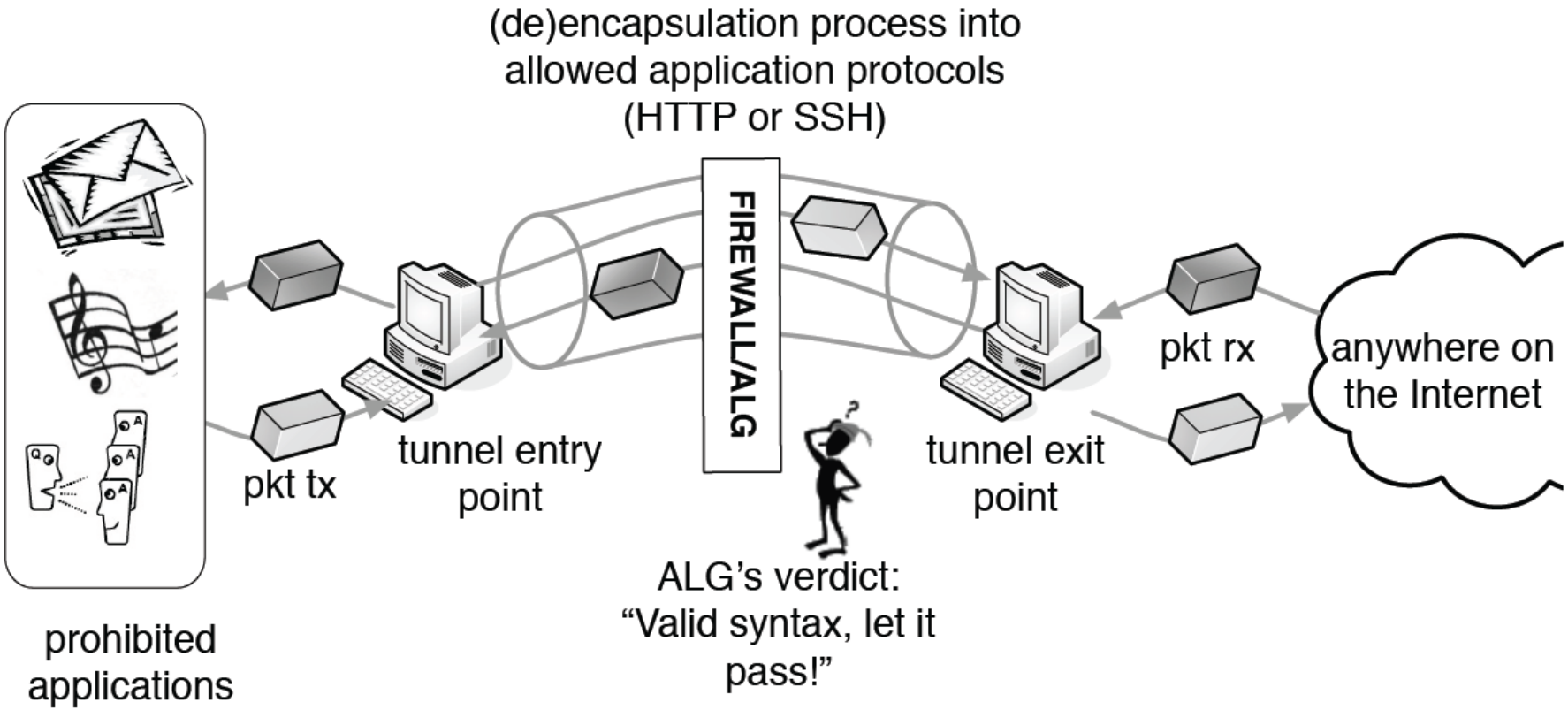


Fig. 1. How tunnels work: high-level scheme.

Aporia

- Hard to distinguish protocols as the traffic is encrypted or encapsulated
- Completely blocking all connections might not be possible.
- The legitimate use of tunnel techniques is essential to the job performed by the users.
- Unable to confirm all types of traffic which would be tunneled on the top of HTTP or SSH.

Key Ideas

- Characterizing legitimate uses of a given application-layer protocol.
- The information carried by packets at the network layer, such as packet-size and inter-arrival time between consecutive packets, are enough to infer the nature of the application protocol.

Contributions

- The formalization of a statistical model, based on established pattern recognition approaches, that can be used to characterize application-layer protocols.
- The definition of a simple classification algorithm that can discriminate when an application protocol, characterized with the technique mentioned above, is being used to tunnel another protocol on top of it.

Contributions

- The report on a series of experiments carried out on a real network that prove the effectiveness of the technique we propose here.
- The analysis of how adding knowledge to the classifier, even when not directly related to the classification target, can significantly improve the accuracy of the tunnel detector.

Tunneling Techniques

- DNS tunnels
  - rarely achieve throughputs higher than a few kb/s.
- HTTP tunnels
  - aim tests at tunnels generated by *httptunnel*.
- SSH tunnels
  - runs on top of TCP and it is designed to provide data confidentiality and integrity between two hosts over an insecure network.
  - typically used for command execution through a secure shell and secure file copy between peers.

SSH Protocol

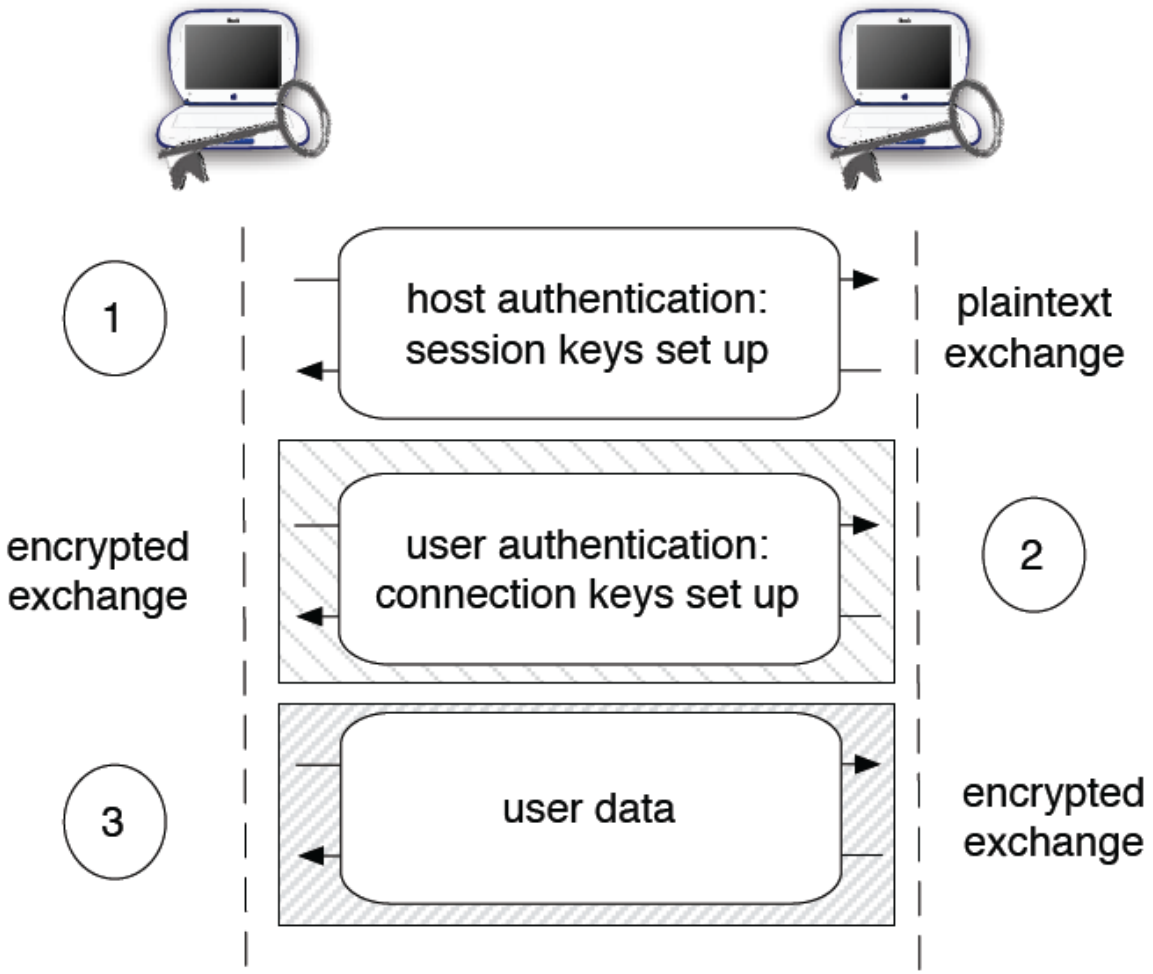


Fig. 2. Authentication stage in SSH.

SSH Authentication Process

- Host authentication
  - This authentication phase is transmitted un-encrypted and ends with an SSH_ MSG_ NEWKEYS message.
  - All messages sent after this must use the negotiated keys and algorithms, and are privacy and integrity protected.
- User authentication
  - It is more difficult to detect when user authentication ends, and actual data starts being exchanged, because the second authentication stage is encrypted.
  - This paper solves this issue assuming that network administrators are required to choose and allow only one kind of SSH user authentication method to implement Tunnel Hunter on their networks.

Pattern Recognition

- A histogram method can be adopted to provide a non-parametric density estimation of the class.

$$\hat{p}(\vec{x}|\omega_i) = \frac{n_j}{\sum_{j \in N} n_j dV}$$

$$\hat{p}(\vec{x}|\omega_i) = \prod_{k=1}^{r} \hat{p}(x_k|\omega_i)$$

$$\hat{\omega}(\vec{x}) = f(\vec{x}) = \arg\max_{\omega_j} \hat{p}(\vec{x}|\omega_j)$$

Stages in Pattern Recognition

- Data collection.
- Feature selection or feature extraction.
- Definition of patterns and classes.
- Definition and application of the discrimination procedure.
- Assessment and interpretation of results.

Tunnel Hunter

- A simple principle
  - since characterizing precisely all possible non-legitimate activities is practically impossible, Tunnel Hunter focuses on building an accurate description of legitimate traffic, trying then to detect which flows to block by comparing their behavior to the target class.

Measurement of Features

- The packet size s and the inter-arrival time Δt between two consecutive packets.
  - from the beginning stage of each TCP connection.
  - not to consider packets that do not carry TCP payload.

$$\vec{x} = \begin{pmatrix} s_1 & \dots & s_r \\ \log_{10}[\Delta t_1] & \dots & \log_{10}[\Delta t_r] \end{pmatrix}$$

Class Model

- One class classifier
  - requires to train the classifier only with flows from a single legitimate class.
- Multi-class classifier
  - new classes composed of outlier flows are added to the analysis.
- Characterize separately the behavior of each of the r column vectors inside pattern X.

$$\{N(s) \cdot N(\Delta t)\}^r \longrightarrow \{N(s) \cdot N(\Delta t)\}$$

Class Model

- Parzen method
  - filter the density estimations matrices using either a Gaussian or a hyperbolic secant kernel.
  - each one defined in a N (s) ·N (Δ t) domain.

$$G(\vec{u}) = \frac{1}{2\pi h^2} e^{-\frac{||\vec{u}||^2}{2h^2}} \qquad \mathrm{Sech}(\vec{u}) = \mathrm{sech}\frac{||\vec{u}||}{h}$$

One-class tunnel detection algorithm

- Fingerprint M models the behavior of each of the first L packets in the flow without taking into account any possible influence between consecutive packets.
- Every packet will hence supply a small contribution to the metric: pieces will be then combined independently.
- Anomaly score

$$S(\vec{x}|\omega_t) = \left| \frac{\log_{10} \prod_{i=1}^{\min(r,L)} p(x_i|\omega_t)}{\min(r, L)} \right|$$

$$p\left(x_i|\omega_t\right) \simeq M_i\left(s_i, \Delta t_i\right)$$

One-class tunnel detection algorithm

$$\hat{\omega}\left(\vec{x}\left(F\right)\right) = \begin{cases} \omega_t \text{ if } S(\vec{x}|\omega_t) < T_{acc} \\ \omega_r \text{ otherwise,} \end{cases}$$

- Chose for *Tacc* the lowest value that allowed exactly 99% of the trained flows.

One-class tunnel detection algorithm

- In the case of SSH tunnels
  - n_0 should point to the first packet that carries actual user traffic in each SSH session.

$$S(\vec{x}|\omega_t) = \left| \frac{\log_{10} \prod_{i=n_0}^{n} p(x_i|\omega_t)}{(n - n_0 + 1)} \right|$$

Experimental Assumption

- The packet size s assumes values in the range [40, 1500].
  - variable s is quantized with step 1 bytes.
  - N (s) = 1461.
- The time interval values in the range $[10^{-7},10^{3}]$
  - inter arrival time with step 10^{-2}.
  - N (Δ t) = 1001.
- Each matrix takes around 1.5 million cells.

Experimental Results

- The HTTP case (Tacc = 7.84)

Protocols	Hit ratio	# sessions
HTTP	98.71%	15000
POP3 over HTTP	100%	6400
SMTP over HTTP	100%	3500
CHAT over HTTP	100%	5900
P2P over HTTP	88.09%	1000

Experimental Results

- The SSH case (Tacc = 6.86)

Protocols	Hit ratio	# sessions
SSH	99.00%	600
SCP	99.10%	1700
POP3 over SSH	88.55%	2360
SMTP over SSH	99.92%	4300
CHAT over SSH	90.69%	2100
P2P over SSH	82.45%	1600

Multi-class Classification

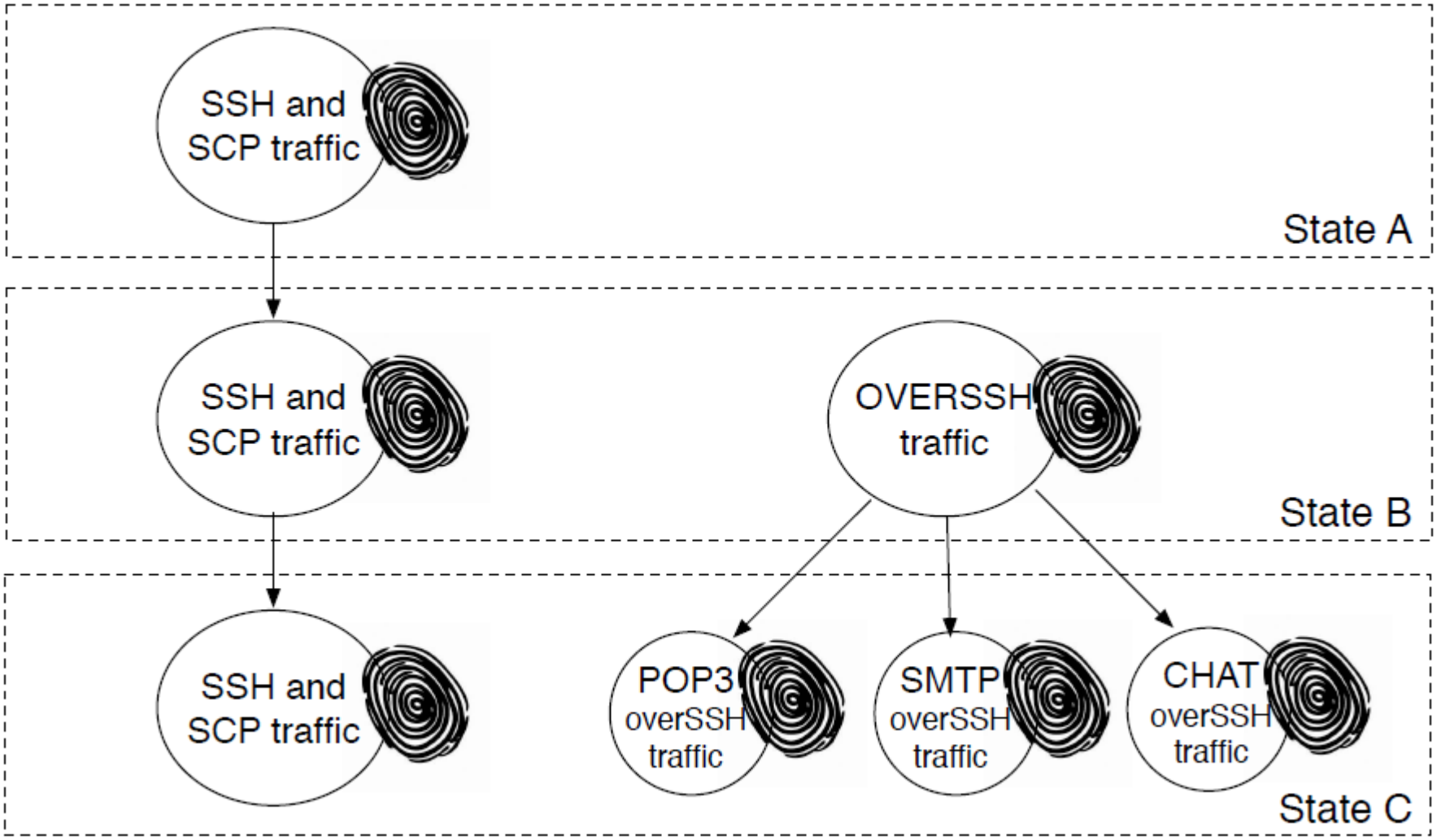
- Tunnel Hunter can perform better if it is provided with more knowledge about the nature of the traffic that needs to be blocked.

$$S(\vec{x}|\omega_i) = \left| \frac{\log_{10} \prod_{j=n_0}^{n} p(x_j|\omega_i)}{(n - n_0 + 1)} \right|$$

$$S(\vec{x}|\omega_m) = \min_{\omega_i \in C} \{S(\vec{x}|\omega_i)\}$$

- $\omega_m \equiv \omega_t$ (i.e., the candidate class is ω_t)
- $S(\vec{x}|\omega_m) \leq T_{acc}$,

Adding Knowledge to Tunnel Hunter



Experimental Results

- The SSH case (Tacc = 6.86)

Protocol	SSH and SCP	Unknown
SSH	99.00%	1.00%
SCP	99.10%	0.90%
POP3 over SSH	0.66%	99.34%
SMTP over SSH	0.08%	99.92%
CHAT over SSH	0.08%	99.92%
P2P over SSH	0.47%	99.53%

State B

Experimental Results

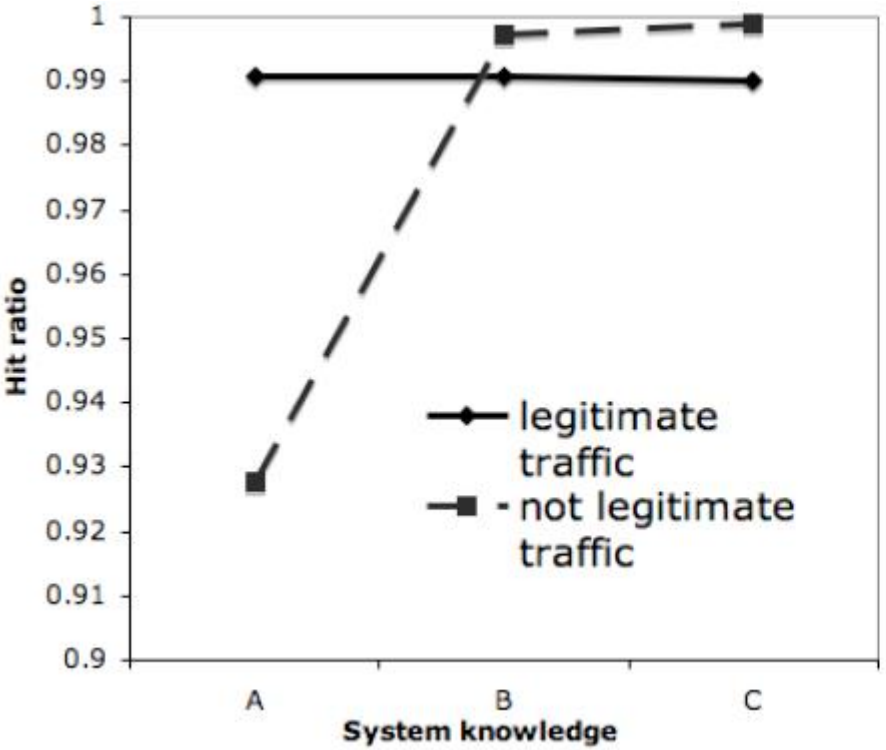
- The SSH case (Tacc = 6.86)

Protocol	SSH and SCP	Unknown
SSH	99.33%	0.67%
SCP	98.91%	1.09%
POP3 over SSH	0.28%	99.72%
SMTP over SSH	0.00%	100.00%
CHAT over SSH	0.00%	100.00%
P2P over SSH	0.21%	99.79%

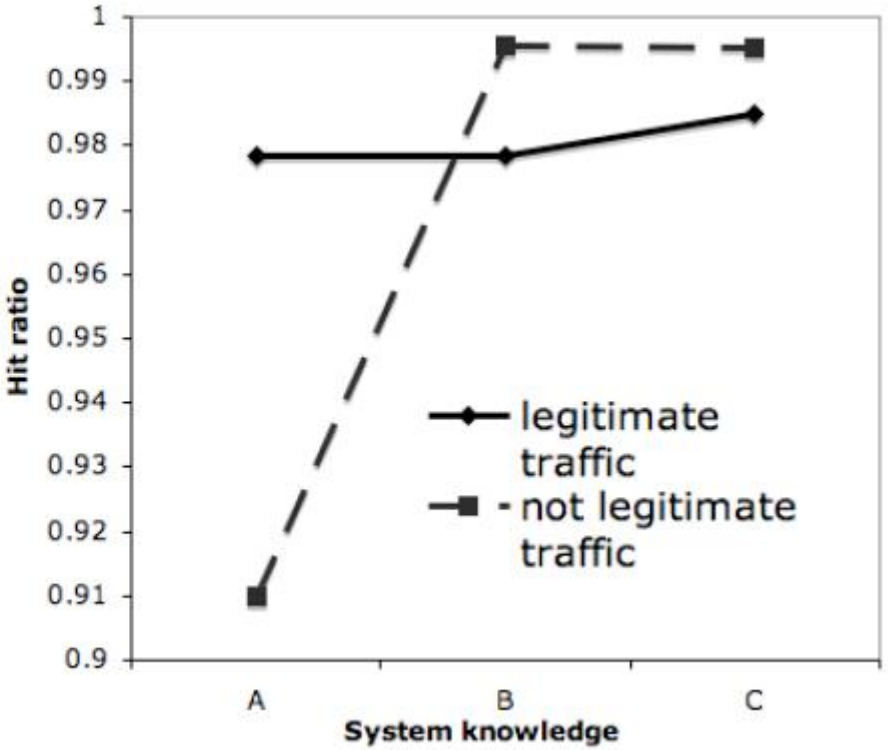
State C

Hit-Ratio Trend

hyperbolic secant kernel



Gaussian kernel



Potential Attacks

- Opens a tunnel over SSH and initially uses it for remote administration. After this allowed stage, he starts exploiting the session to tunnel other kinds of application protocol.
- It is sensitive to packet-size and timing value manipulation.

Thank you!

Any Question?