# Computational Decoys for Cloud Security

Angelos D. Keromytis
Columbia University

# Decoys

- Fake objects whose purpose is to deceive adversaries
  - Detection of adversary
  - Diversion (attraction) of effort
  - Harm avoidance
  - Continued misdirection
  - Reach-back

# Real Life Decoys

The New York Times

N.Y. / Region

WORLD | U.S. | N.Y. / REGION | BUSINESS | TECHNOLOGY | SCIENCE | HEALTH | SPORTS | OPINION

## Police to Use Fake Pill Bottles to Track Drugstore Thieves

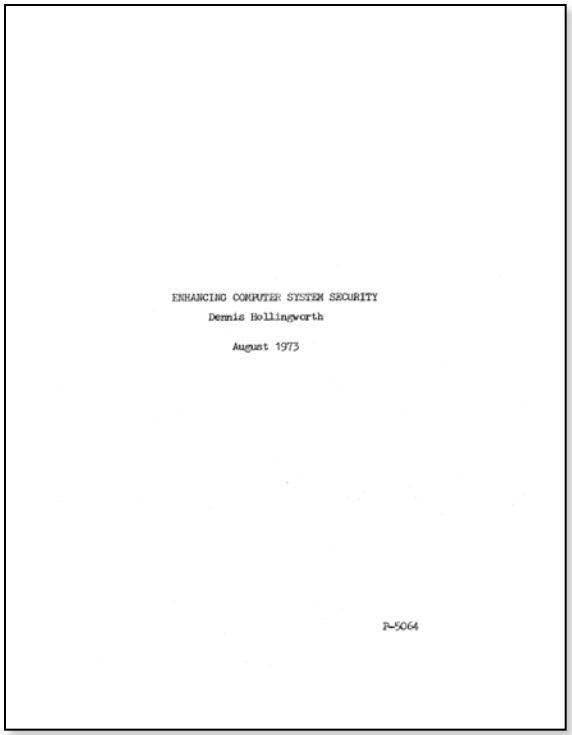
By JOSEPH GOLDSTEIN
Published: January 15, 2013

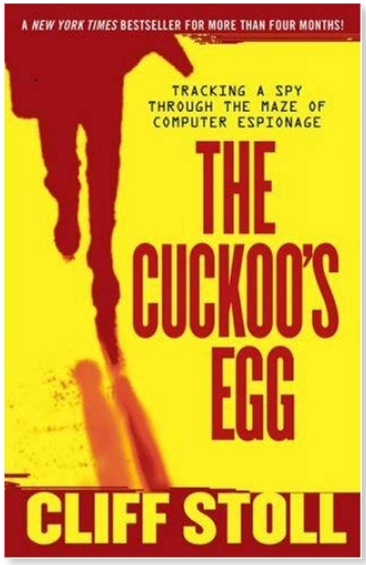
Seeking to catch and eventually deter addicts who steal painkillers, the New York Police Department will stock pharmacy shelves with decoy pill bottles that contain tracking devices.

FACEBOOK
TWITTER
GOOGLE+

# Traditional Cyber Decoys

- Well-known security strategy
  - Honeypots
  - Honey-monkeys (web)
  - Honey-accounts (email spam)


ENHANCING COMPUTER SYSTEM SECURITY
Dennis Hollingworth
August 1973
R-5064





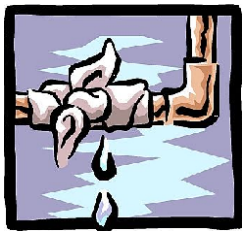
An Evening with Berferd
In Which a Cracker is Lured, Endured, and Studied
Bill Cheswick
AT&T Bell Laboratories
Abstract
1. Introduction

# Information Decoys

- Existing work on use of decoys (“honeytokens”)
  - Large scale
  - Automated generation and management
  - Ubiquitous/pervasive

# Why?

- Focus on attack target: information
  - It will leak



- Attempt to sidestep the technical arms race

# Decoys as a Primitive

- Decoys/deception should be considered a general purpose primitive in cybersecurity, akin to cryptography
- Properties
  - diverse
  - flexible
  - principled
  - measurable

# Decoy Properties

- Believability
- Enticingness
- Detectability
- Variability
- Conspicuousness
- Non-interference

# Believability Formalization

- Defined for object space M and decoy set D
- Decoy Believability Experiment
  - For any $d \in D$, choose two objects $m_0, m_1 \in M$ such that $m_0 = d$ or $m_1 = d$, and $m_0 \neq m_1$
  - Adversary A obtains $m_0$, $m_1$ and attempts to choose $m^* \in \{m_0, m_1\}$ such that m* != d, using only information intrinsic to $m_0$, $m_1$
  - The output of the experiment is 1 if m* != d and 0 otherwise.
- Perfect decoy when: $Pr[Exp_{believe} = 1] = 1/2$

# Broad Applications

- Network eavesdropping [WiSec 2009]
- Tor eavesdropping [RAID 2011]
- Keystroke loggers/rootkits [RAID 2010]
- Source code [ASIACCS 2012]
- Documents (unstructured data) [SecureComm 2009]

# Computational Decoys

- Move from data to computation
  - Not entirely distinct
- Create uncertainty and confusion to adversaries that gain access to the cloud infrastructure

# Computational Decoys: Goal

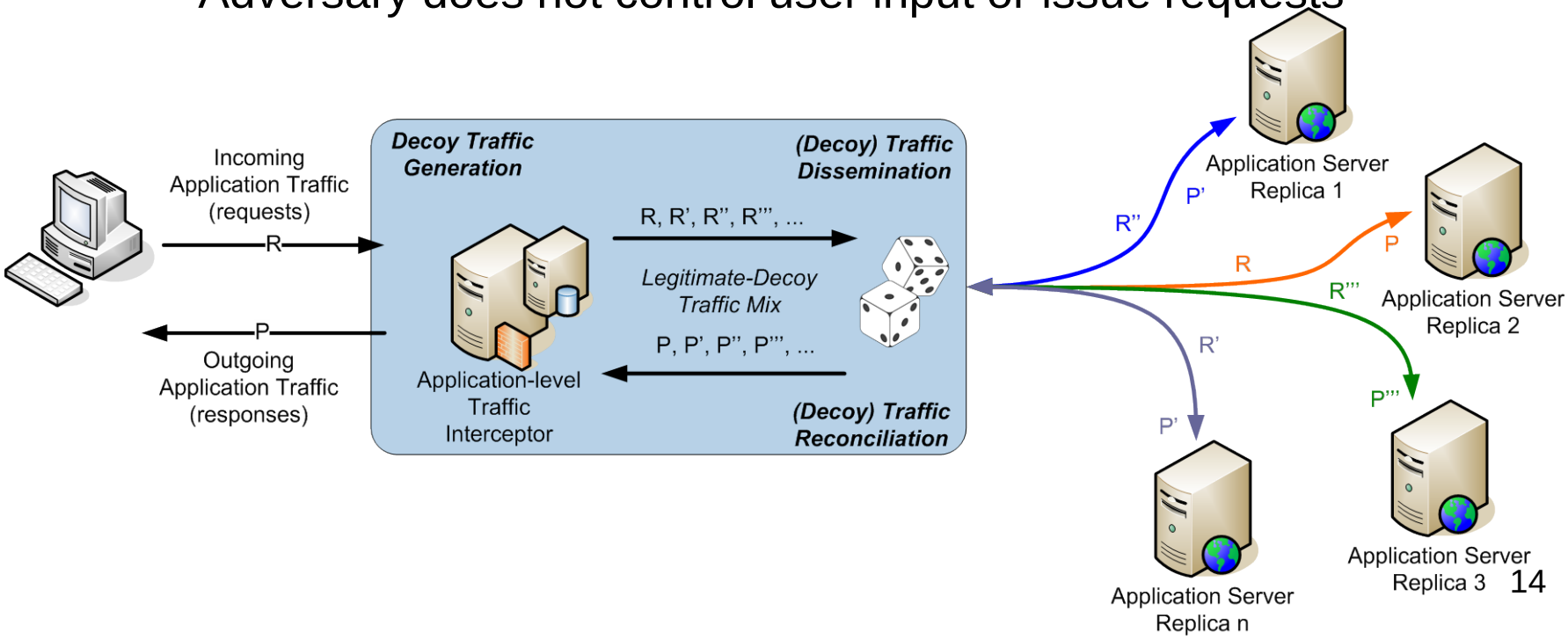
- Make it impossible to determine whether a captured system is handling real or decoy processing within N time units
  - Time units are envisioned to be in the order of hours or days
- Opportunity for detection, misdirection, and engagement with adversary

# Assumptions

- Partitioned, replicated applications (cloud!)
- Adversary can have access to full system
  - Possibly including root/kernel-level access
- Adversary cannot see all network traffic inside and into/out of the cloud
  - Adversary cannot determine whether a specific connection is from the outside or from a cloud-local proxy, except for adversary's own connections
- Adversary cannot readily determine ground truth in most cases

# PoC: DIGIT

- Goal: create uncertainly to the adversary as to what is real
- Threat model: unknown number of replicas is compromised
  - Compromised replicas receive requests with mission-sensitive info
  - Adversary does not control user input or issue requests

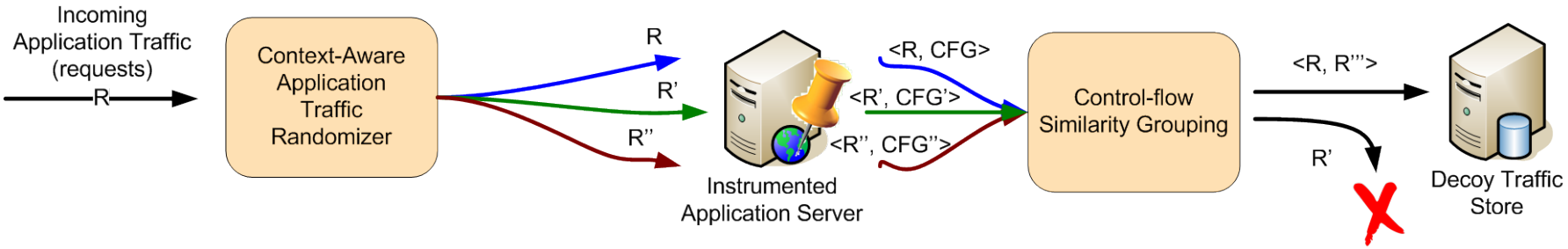


# Application-level Traffic Interceptor

- TLS terminator and incoming network traffic inspector
- Application-level protocol Identifier
  - Currently: port-based (e.g., 80 is HTTP) and single-message protocol-based (e.g., filters matching HTTP requests)
- Application-specific user input parsing (modules)
  - pairs incoming user-initiated requests with similar decoy requests to produce a legitimate-decoy traffic mix
  - decoys are generated offline and stored in the decoy store
  - randomly disseminates the legitimate-decoy traffic mix to the set of replicas, designated to handle original client request
  - reconciles replies from replicas and forwards the result to answer client's request

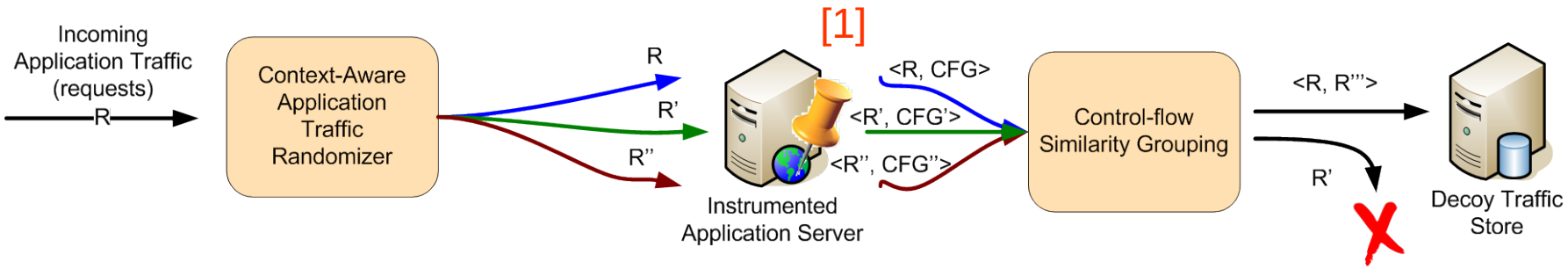
# Decoy Traffic Generation

- Generate input variations for client's request
  - Context-aware application input randomization
  - Example case: Web applications
    - Interaction endpoints with clients are well-defined and documented
    - Application-specific modules are written to identify the variable parts of incoming requests and invoke appropriate randomization routine (e.g., byte-range or dictionary based)
  - Are these variations plausible?



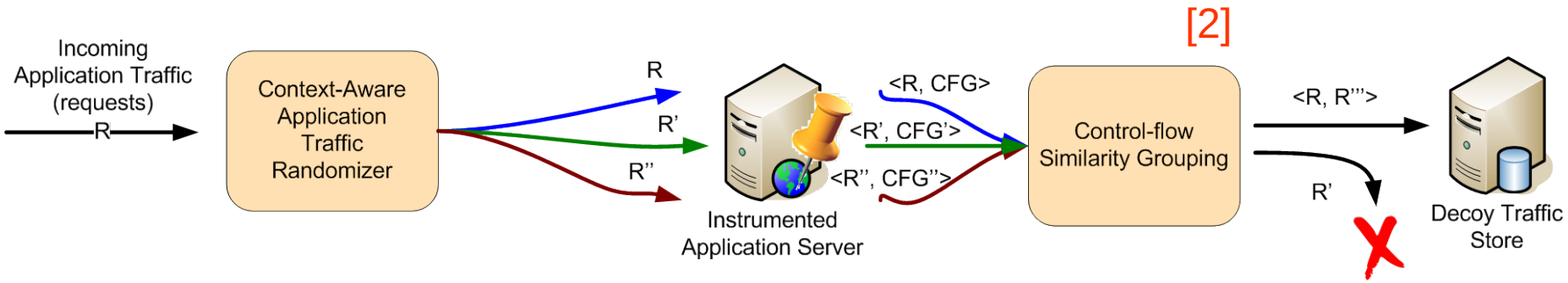
# Decoy Traffic Generation

- Identify variations exhibiting similar application behavior
  1. Evaluate application behavior for generated decoys
     - Dynamic binary instrumentation of (Web) application with a PIN tool
     - Output of application execution decisions (CFG, BB, SysCalls)



# Decoy Traffic Generation

- Identify variations exhibiting similar application behavior
  2. Similarity grouping based on application execution path
     - Execution trace evaluation tool determines alignment of decoy-input behavior with actual-input behavior.
       - Strict mode: absolute or near-absolute alignment
       - Relaxed mode: deviations are permitted as long as both traces align at the beginning and end and overlap more than T %.



# Decoy Traffic Generation

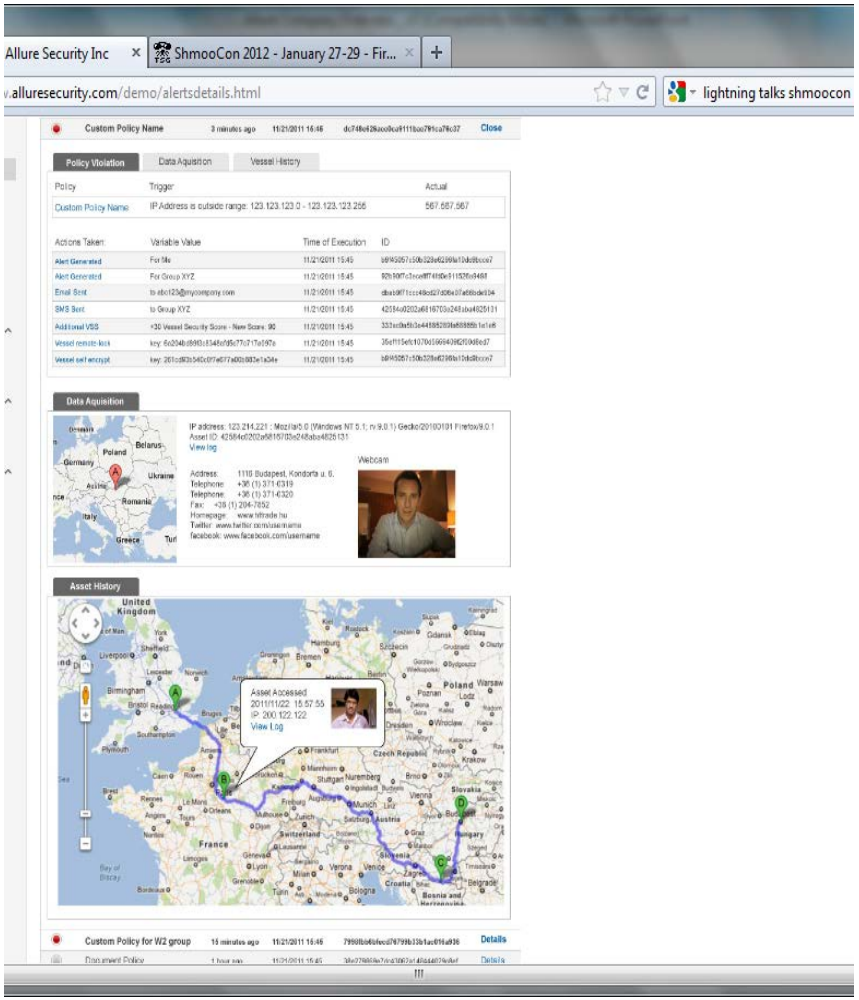
- Identify variations exhibiting similar application behavior

  2. Similarity grouping based on application execution path
     - Evaluated on simple custom-built Web applications.
     - Aligned successfully different HTTP or application-level responses (HTTP errors, valid HTTP response and in-app error message) with execution trace deviations. Input variations included URL fuzzing and HTTP header manipulation.

- Production of decoy-traffic groups, indexed by legitimate traffic template, stored in decoy traffic database
- Used in real-time by application-level traffic interceptor

# Reachback

- Decoy computation as cover traffic for payloads
  - Active information gathering
  - Forensic analysis (CI)
  - Other payloads…



# Challenges/Next Steps

- Scenario construction
- Characterize supported application (and data) complexity
  - e.g., use of crypto at the application layer
  - Can we design applications/systems with computation decoys in mind?
- Resource expenditure
  - Fine-grained I/O multiplexing across replicas
- Evaluation/validation