MEERKATS:

Maintaining EnterprisE Resiliency via Kaleidoscopic Adaptation & Transformation of Software Services

Roxana Geambasu and Sal Stolfo
Columbia University

MRC PI Meeting, April 2015

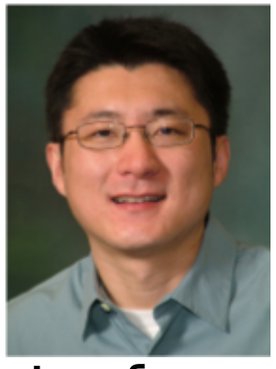
Columbia (current)

Roxana Geambasu

Michalis Polychronakis

Sal Stolfo

Simha Sethumadhavan



Junfeng Yang

GMU (current)

Angelos Stavrou

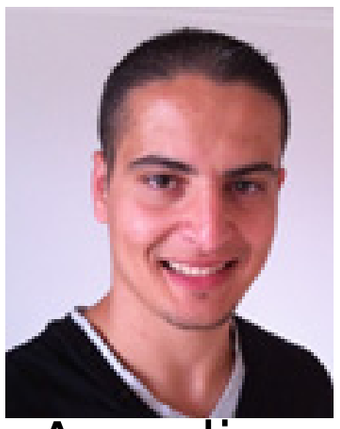
Allure (current)

xxx

Columbia (former)

Angelos Keromytis

Symantec (former)



Azzedine Benameur

Matt Elder

Nathan Evans

MEERKATS

Goal: New cloud architecture that increase resilience by incorporating unpredictability, confusion, and continuous change

- Diversify replica execution
  - CISR, CSSH, DREME, N-Schedule, Hardware support
- Migrate data and services continuously
  - Evade, MiSS
- Deceive attacker with decoy data and requests
  - DIGIT
- Monitor and adapt to suspicious activity
  - DMCC, CIFT

Bean Count

- Diversified Replica Execution
  - N-Schedule: Diversified replication [CACM’14, SOSP’13, PLDI’12,HOTPAR’12]
  - Hardware: Hardware support for performance and security [ISCA’13]
  - CSSH, CISR: Collaborative ISR & self-healing [CCS’13, CCS’12, WRIT’12]
  - DREME: Diversified replica management [ISRCS’13, CSET’13]
- Continuous Migration
  - MiSS: Moving-target DDoS defense [DSN'14, ICCCN’13]
  - Evade: Data cleanup and migration [OSDI'14, OSDI’12]
- Deception
  - DIGIT: Decoy data and computations [CCS’13, WRIT’12, ARO’13]
  - DIGIT Transition: Turning decoys into product [Allure Technologies]
- Monitoring and Detection
  - CIFT: Cloud data tracking [SEC'14, HCI’13, RAID’13, CCSW’12, SEC’12]
  - DMCC: Distributed alert correlation [AISec'14, RAID'14, RAID'14]

CS @CU

Bean Count

- Diversified Replica Execution
  - N-Schedule: Diversified replication [CACM'14, SOSP'13, PLDI'12,HOTPAR'12]
  - Hardware: Hardware support for performance and security [ISCA'13]
  - CSSH, CISR: Collaborative ISR & self-healing [CCS'13, CCS'12, WRIT'12]
  - DREM
- Continu
  - MiSS:
  - Evade
- Deception
  - DIGIT: Decoy data and computations [CCS'13, WRIT'12, ARO'13]
  - DIGIT Transition: Turning decoys into product [Allure Technologies]
- Monitoring and Detection
  - CIFT: Cloud data tracking [SEC'14, HCI'13, RAID'13, CCSW'12, SEC'12]
  - DMCC: Distributed alert correlation [AISec'14, RAID'14, RAID'14]

Total: > 26 pubs to date, DIGIT tech commercialization, NYT article

Renewed Focus

- Push particularly promising technologies toward impact
  - Academic and open-source impact
  - Technology transfer in industry

- Goal: Understand, develop, and commercialize technologies for unpredictability, confusion, and continuous change across a broad use cases

- Today we are focusing on 7 out of the original 11 technologies and pushing them each
  - Our focus is to develop them independently and investigate their powers in many use cases

CS @CU

Projects in Current Focus

- Diversified Replica Execution
  - Stable and diversified multithreading [CACM'14, SOSP’13, PLDI’12, HOTPAR’12]
  - Hardware: Hardware support for performance and security [ISCA’13]
- Continuous Migration
  - MiSS: Moving-target DDoS defense [DSN'14, ICCCN’13]
  - Evade: Data cleanup and migration [OSDI'14, OSDI’12]
- Deception
  - DIGIT: Decoy data and computations [CCS’13, WRIT’12, ARO’13]
  - DIGIT Transition: Making decoys a product [Allure Technologies]
- Monitoring and Detection
  - CIFT: Cloud data tracking [SEC'14, HCI’13, RAID’13, CCSW’12, SEC’12]
  - DMCC: Distributed alert correlation [AISec'14, RAID'14, RAID'14]

CS @CU

Projects in Current Focus

- Diversified Replica Execution
  - Stable and diversified multithreading [CACM'14, SOSP’13, PLDI’12, HOTPAR’12]
  - Hardware: Hardware support for performance and security [ISCA’13]
- Continuous Migration
  - MiSS: Moving-target DDoS defense [DSN'14, ICCCN’13]
  - Evade: Data cleanup and migration [OSDI'14, OSDI’12]
- Deception
  - DIGIT: Decoy data and computations [CCS’13, WRIT’12, ARO’13]
  - DIGIT Transition: Making decoys a product [Allure Technologies]
- Monitoring and Detection
  - CIFT: Cloud data tracking [SEC'14, HCI’13, RAID’13, CCSW’12, SEC’12]
  - DMCC: Distributed alert correlation [AISec'14, RAID'14, RAID'14]

pitched previously

CS @CU

Projects in Current Focus

- Diversified Replica Execution
  - Stable and diversified multithreading [CACM'14, SOSP’13, PLDI’12, HOTPAR’12]
  - Hardware: Hardware support for performance and security [ISCA’13]
- Continuous Migration
  - MiSS: Moving-target DDoS defense [DSN'14, ICCCN’13]
  - Evade: Data cleanup and migration [OSDI'14, OSDI’12]
- Deception

pitching today

  - DIGIT: Decoy data and computations [CCS’13, WRIT’12, ARO’13]
  - DIGIT Transition: Making decoys a product [Allure Technologies]
- Monitoring and Detection
  - CIFT: Cloud data tracking [SEC'14, HCI’13, RAID’13, CCSW’12, SEC’12]
  - DMCC: Distributed alert correlation [AISec'14, RAID'14, RAID'14]

CS @CU

Agenda

- Reminder of previously pitched major results
  - Major result 1: Stable multithreading (N-Schedule)
  - Major result 2: Multi-sensor architectures (DMCC)
  - Major result 2: Moving target DDoS defense (MiSS)
- Major result 4: Decoy technical transition (DIGIT)
  - Sal Stolfo (Columbia, Allure) will present and demo
- Major result 5: Design and implementation of a ``clean'' operating system (Evade)
  - Roxana Geambasu (Columbia) will present
- Include slides for other major results to be presented at next PI meeting

CS@CU

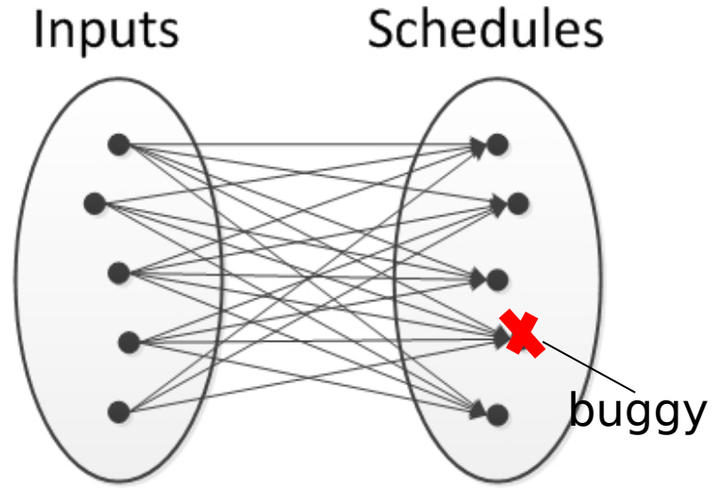
Agenda

- Reminder of previously pitched major results
  - Major result 1: Stable multithreading (N-Schedule)
  - Major result 2: Multi-sensor architectures (DMCC)
  - Major result 2: Moving target DDoS defense (MiSS)
- Major result 4: Decoy technical transition (DIGIT)
  - Sal Stolfo (Columbia, Allure) will present and demo
- Major result 5: Design and implementation of a ``clean'' operating system (Evade)
  - Roxana Geambasu (Columbia) will present
- Include slides for other major results to be presented at next PI meeting

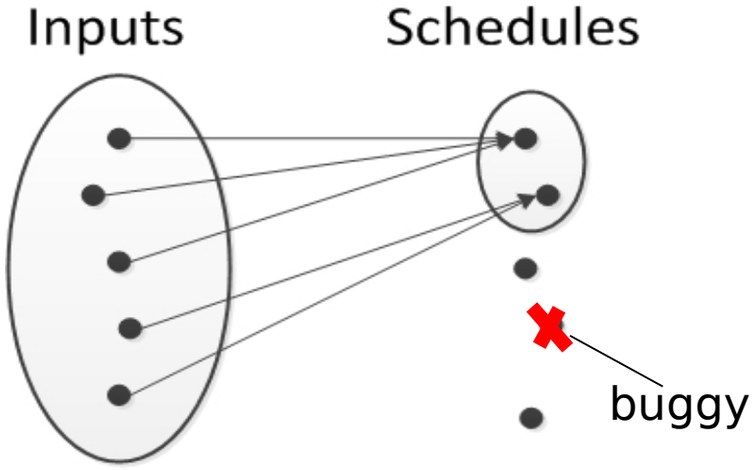
CS@CU

Major Result 1: Stable Multithreading

- Problem: multithreading is hard
  - Too many interleavings (schedules) to reason about, test, or model check
  - Concurrency bugs are exploitable [FSE'13]



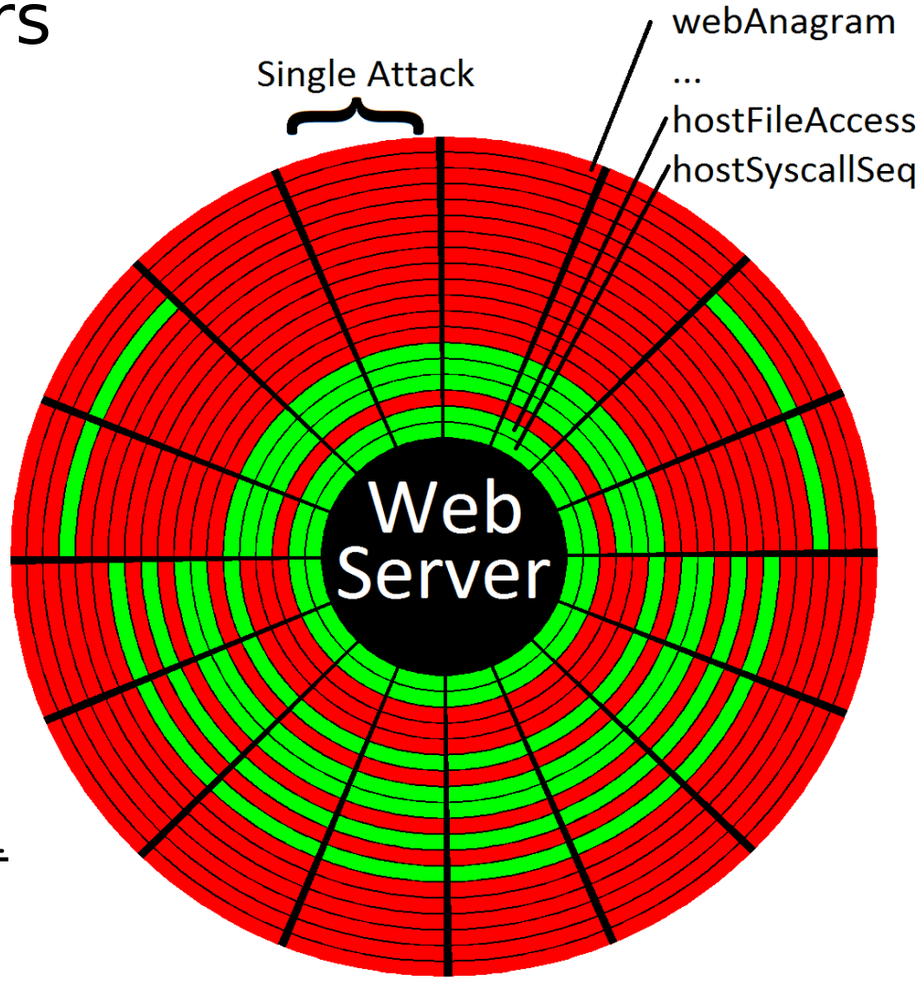
- Stable multithreading
  - Runtime that constrains schedules
  - What you check is what you get!
  - [SOSP'13, PLDI'12, SOSP'11]
  - Src: http://github.com/columbia/smt-mc



- Technology has many use cases
  - Enables higher coverage for model checking and testing tools
  - Enables diversification that can confuse adversaries
  - **Enables black-box replication** ← awesome result (next time)!

Major Result 2: Multi-Sensor Monitoring and Correlation

- Problem: no single sensor (AD, AV) is perfect
  - Often companies deploy multiple sensors
  - But what is the cost and how much "more secure" are they?



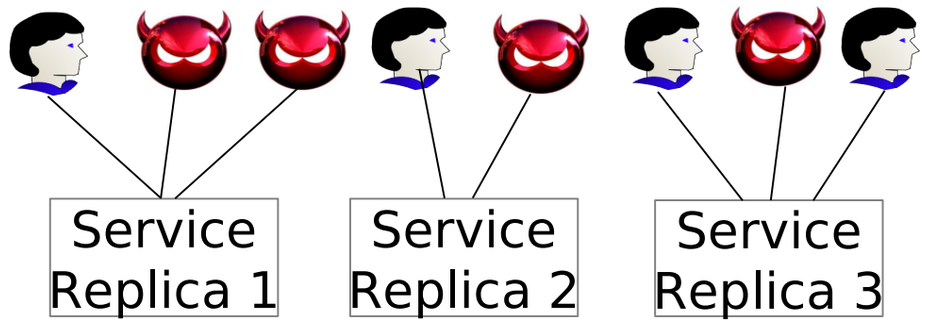
- WindTunnel
  - Framework for assessing the security of multi-sensor architectures
  - Modular, visual, has 500GB+ datasets!
  - [AISec'14, RAID'14]
  - Src: ids.cs.columbia.edu/content/windtunnel

- Technology has many use cases
  - Assess cost/security tradeoffs of defense-in-depth architectures
  - (MRC) Lets us evaluate **cross-layer sensor correlations**

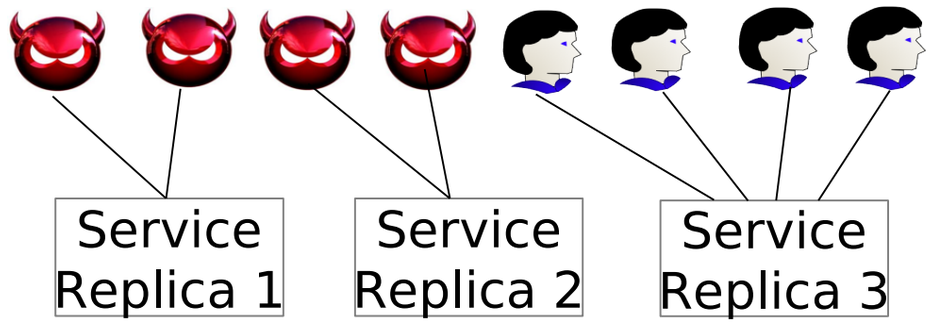
CS @CU

Major Result 3: Moving Target DDoS Defense

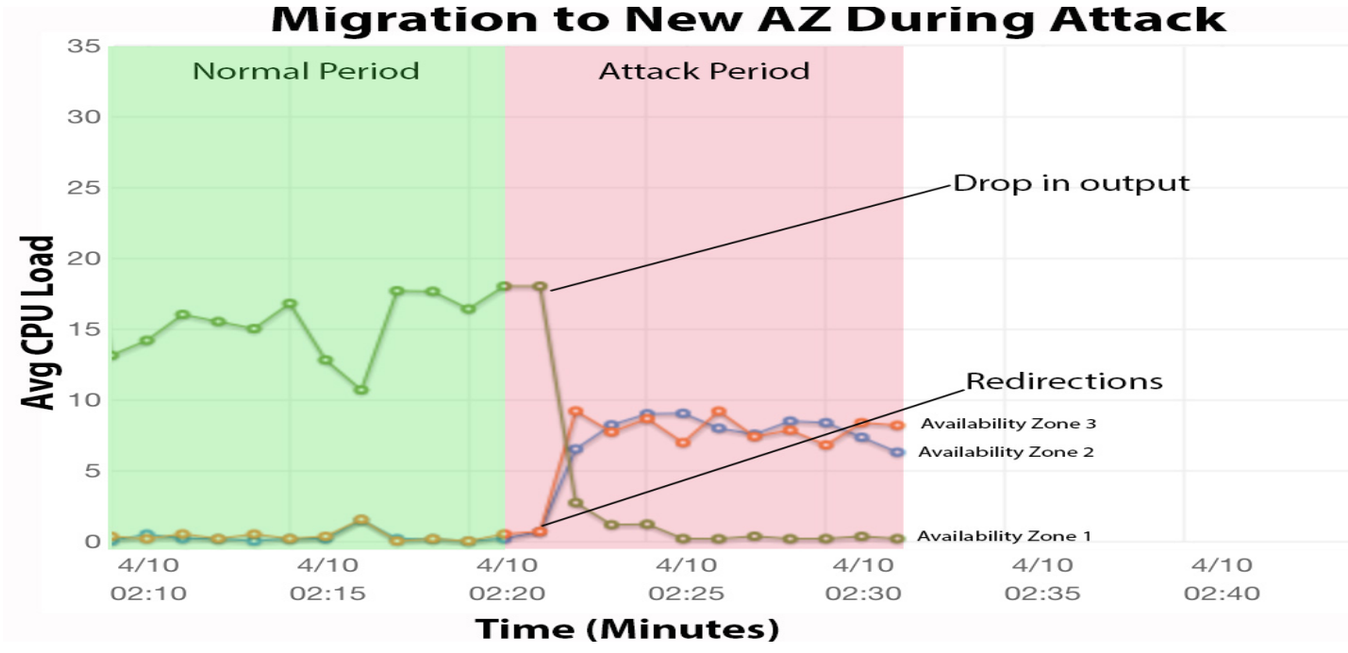
- Problem: mitigating clever DDoS attacks is hard



- Moving target defense
  - Idea: shuffle users so as to “save” benign clients from attacker bots
  - [DSN'14, ICCCN’13]



- (Recent) Implementation on AWS is effective



Agenda

- Reminder of previously pitched major results
  - Major result 1: Stable multithreading (N-Schedule)
  - Major result 2: Multi-sensor architectures (DMCC)
  - Major result 2: Moving target DDoS defense (MiSS)
- Major result 4: Decoy technical transition (DIGIT)
  - Sal Stolfo (Columbia, Allure) will present and demo
- Major result 5: Design and implementation of a ``clean'' operating system (Evade)
  - Roxana Geambasu (Columbia) will present
- Include slides for other major results to be presented at next PI meeting

CS@CU

Major Result 4: Decoy Data and Computations

- Problem: protecting data and services is hard
- Decoys in DIGIT:
  - Hide data/services in plain sight by generating fake but believable data/services
  - We've built decoy factories for multiple types of data
    - HTTP requests
    - Passwords
    - Mobile apps and data

Sample request:
GET /search?q=sushi HTTP/1.1

Generated decoys:
GET /search?q=books HTTP/1.1
GET /search?q=burgers HTTP/1.1
GET /search?q=windows HTTP/1.1

- Concept is valuable in surprisingly many use cases
  - Cloud attacks, mobile theft and loss mitigation, identity theft,...
  - **We are commercializing decoys for mobile devices now**

Sal's Slides
(15 minutes)

Agenda

- Reminder of previously pitched major results
  - Major result 1: Stable multithreading (N-Schedule)
  - Major result 2: Multi-sensor architectures (DMCC)
  - Major result 2: Moving target DDoS defense (MiSS)
- Major result 4: Decoy technical transition (DIGIT)
  - Sal Stolfo (Columbia, Allure) will present and demo
- Major result 5: Design and implementation of a ``clean'' operating system (Evade)
  - Roxana Geambasu (Columbia) will present
- Include slides for other major results to be presented at next PI meeting

CS @CU

Major Result 5: Clean Operating System Design

- Problem: OSes accumulate a lot of sensitive data
  - OS doesn’t securely erase deallocated data
  - FS doesn’t securely erase deleted files
  - Applications hoard sensitive data for performance or convenience

- CleanOS: OS designed to continuously cleanse itself of sensitive data in anticipation of attack
  - Monitors the use of data objects and “evicts” them when unused
  - Enables access auditing at meaningful object level
  - [OSDI'14, OSDI'12]

- Concept is broadly applicable to many use cases
  - Clouds, mobile devices
  - Our implementation is for the Android OS, hence I'll focus on that

CS @CU

Major Result 5:

The Design and Implementation of a "Clean" Operating System

(Evade Component of MEERKATs)

[OSDI'14, OSDI'12]

Example Motivating Scenario: Mobile Theft or Loss

Example Motivating Scenario: Mobile Theft or Loss

- Today's answers: Anything and Maybe
  - Anything could have been on the device
  - The thief could have accessed anything or nothing, no way to tell

Example Motivating Scenario: Mobile Theft or Loss

- Today's answers: Anything and Maybe
  - Anything could have been on the device
  - The thief could have accessed anything or nothing, no way to tell
- CleanOS provides more meaningful answers:
  - Minimizes amount of data on the device at any time
  - Lets a user audit which "objects" were on the device at theft time and which have been accessed since

The Problem

- OSes accumulate enormous sensitive data
  - OS doesn’t securely erase deallocated data
  - FS doesn’t securely erase deleted files
  - Applications hoard sensitive data for performance or convenience

- This data is placed at risk if the device is stolen or lost
  - Attacker can dump RAM, flash memory contents
  - Attacker can break passwords
  - Attacker can fabricate user's fingerprints
  - …

Examples of Exposed Data

App	Password	Contents
Email	✔	✔
Y! Mail	✔	✔
Google Docs		✔
OI Notepad		✔
Dropbox		✔
KeePass	✔	✔
Keeper	✔	✔
Pageonce	✔	✔
Mint		✔
Google+		✔
Facebook		✔
LinkedIn		✔
TOTAL	**5/14**	**13/14**

- We dumped memory and SQLite DB for 14 popular Android apps

Default Email App:

- Password and email snippets are kept in cleartext RAM at all times
- Everything is present in cleartext SQLite DB

Securing Data Is Darn Hard!

- Example protection systems:
  - Encrypted file systems
  - Encrypted RAM
  - Remote wipe-out systems

- Challenges / limitations:
  - Users don’t lock their devices (56% based on 2013 Lookout survey)
  - Fingerprints can be manufactured
  - Physical attacks are notoriously difficult to protect against
    - E.g., memory dumps, cold boot attacks, breaking trusted-hardware seals

- In general, these are imperfect stop-gaps on top OSes that were never designed with physical insecurity in mind

Need New OS Design

- OSes should manage sensitive data rigorously, so as to maintain the device “clean” at any point in time in anticipation of device theft/loss
- If device is stolen, lost, or otherwise compromised:
  1. The minimal amount of sensitive data is exposed
  2. User can tell exactly what was exposed

CleanOS

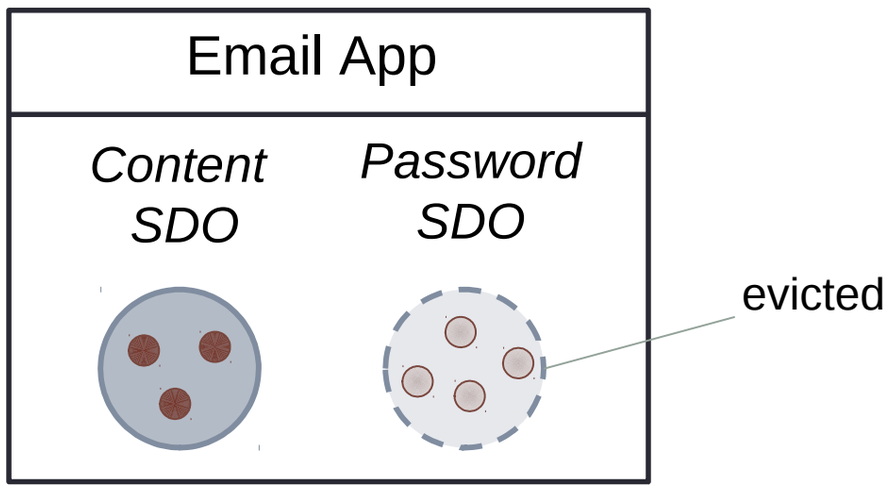
- New OS that minimizes sensitive data exposure by evicting it to a trusted cloud whenever not under active use

- Implements sensitive data objects (SDOs)
  - Identifies locations of sensitive data in RAM and stable storage
  - Monitors its use by applications
  - “Evicts” sensitive data to the cloud whenever it is not under active use

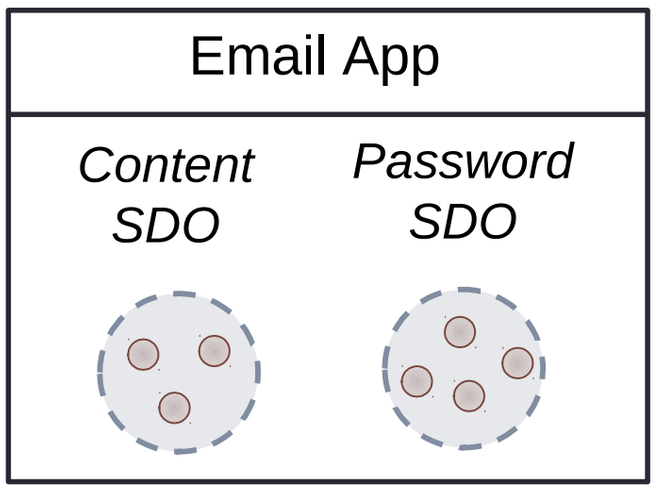
- The cloud intermediates all accesses to unused SDOs and can offer useful post-loss functions:
  - Disable SDO access after theft
  - Audit SDO exposure and access

CleanOS Basic Functioning

- Applications create SDOs and add sensitive data to them
- CleanOS implements three functions for SDOs:
  1. Tracks data in SDOs using taint tracking
  2. Evicts SDOs to a trusted cloud whenever idle
  3. Decrypts SDO data when it is accessed again



when user reads an email



when the app goes to the background

The SDO Abstraction

- An SDO is a logical collection of Java objects that contain sensitive data and that are related somehow
  - Email-password SDO: password string + all objects computed from it
  - Email-contents SDO: all emails in a thread
  - Bank-account SDO: all transactions in an account

human-readable and used for auditing

```
class SDO {
   SDO(String description, ...);    // create SDO
   void add(Object o);              // adds object to SDO
   void remove(Object o);           // removes object from SDO
   ...
   private int sdoID;               // unique ID for SDO
}
```

used for identifying SDOs locally and remotely

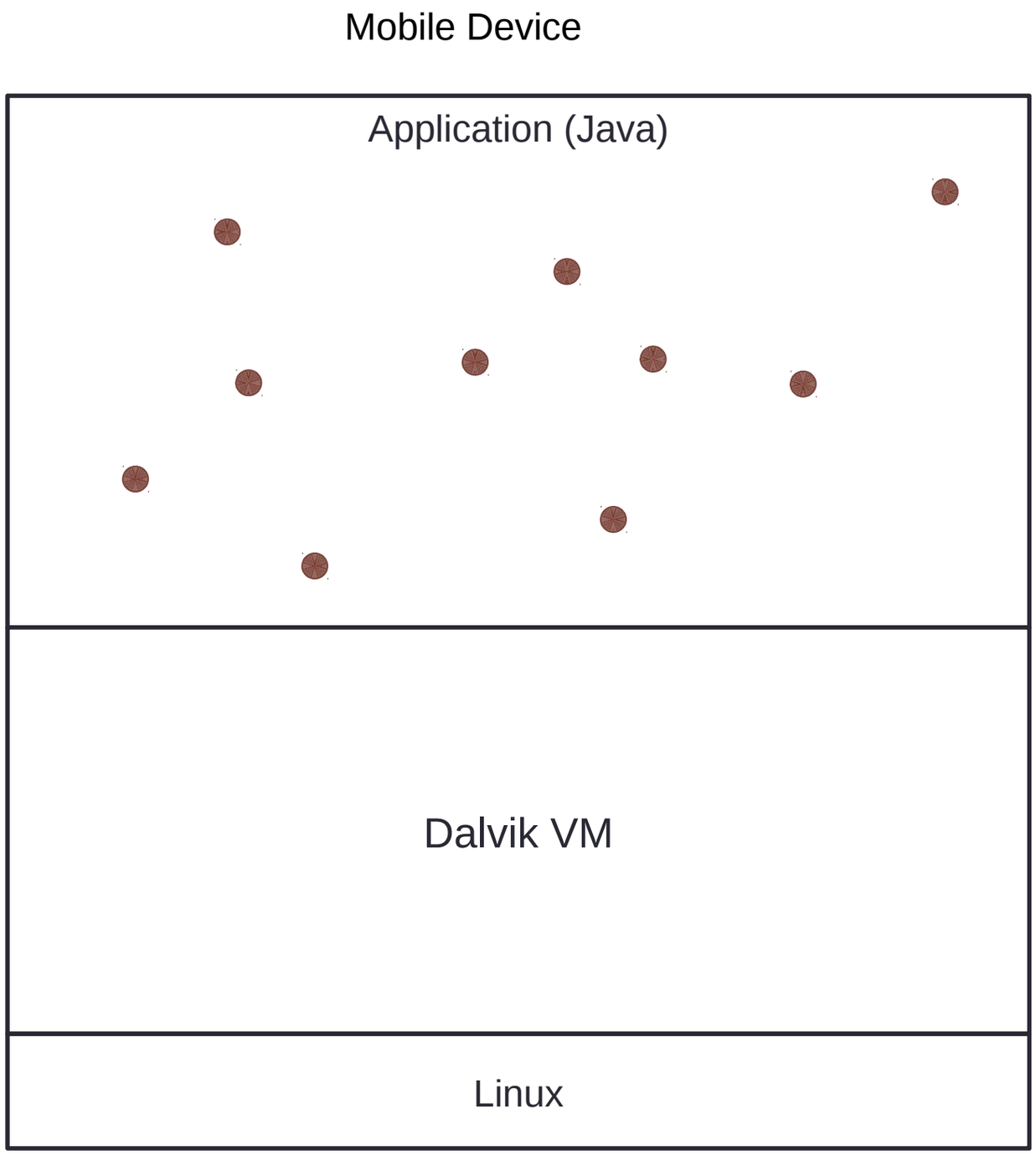
Example: A Clean Email App

- Default Email app hoards passwords and subjects
- Hard for apps to manage sensitive on their own
- With CleanOS, developers simply create SDOs and place their sensitive data in them

```
SDO pwSDO = new SDO(“Password of ” + user);
pwSDO.add(password);
...
SDO emailSDO = new SDO(“Email: ” +
mSubject);
emailSDO.add(mSubject);
emailSDO.add(mTextContent);
emailSDO.add(mHtmlContent);
emailSDO.add(mTextReply);
emailSDO.add(mHtmlReply);
```

(code adapted for clarity)

The CleanOS Architecture --- To be completed later XXX



Prototype and Evaluation

- Built CleanOS on modified Android
  - We plugged into the Java garbage collector to encrypt sensitive data
- Modified several apps to use API to declare sensitive data
- Built post-theft auditing service

Component	New or Changed Features
Dalvik (JVM)	Evict-idle Garbage Collector (eiGC) Eviction-aware bytecode interpretation Secure deallocation of interpreted stacks
Android SDK	SDO API Default SDO heuristics
TaintDroid	Support for millions of taints SQLite vulnerability fix
SQLite	Taint tracking in database
Webkit	Screen-buffer purging
Bionic (libc)	Secure user-space deallocation
Linux Kernel	Secure page deallocation with grsecurity

- Evaluation highlight: **CleanOS lowers exposure of sensitive data by up to 97% for reasonable overheads**

Does CleanOS Limit Data Exposure?

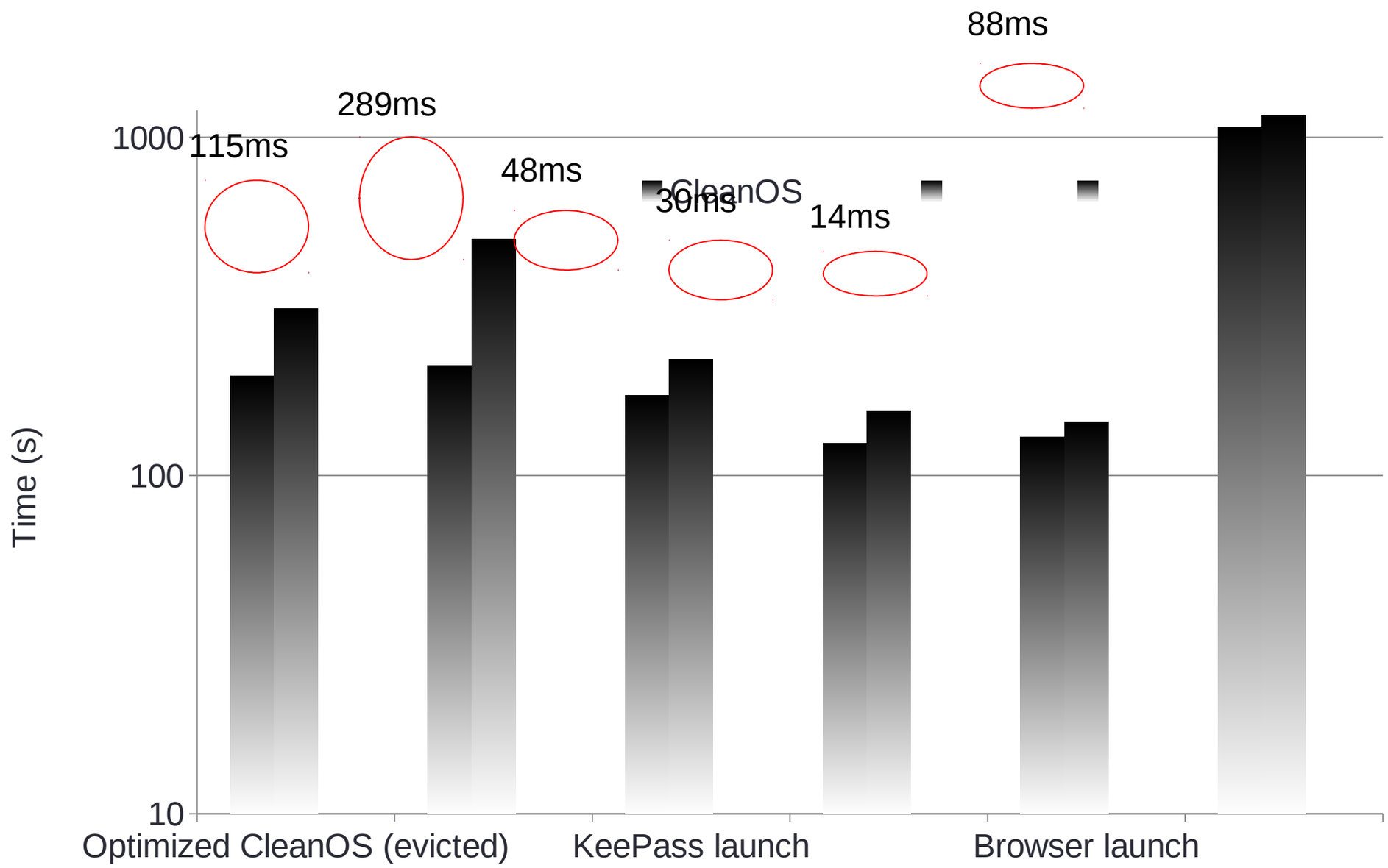
Email App Version	Password SDO	Contents SDO
Email without CleanOS	100%	95.5%
Email with **default SDOs**	6.5%	5.9%
Cleaned Email with **app SDOs**	6.5%	0.3%

Fraction of time in which sensitive data was exposed

- Default SDOs reduce exposure by 89.6-93.5%
- App SDOs further reduce content exposure to 0.3%
- Similar results in Facebook and Mint apps

Thus, default SDOs drastically curb sensitive-data exposure

Is CleanOS Practical (Wi-Fi)?



Thus, overheads are largely unnoticeable over Wi-Fi

CleanOS Summary

- Modern OSes and apps accumulate a lot of sensitive data, exposing it to attacks if device is stolen or lost

- CleanOS is a new Android-based OS design that minimizes amount of sensitive data exposed on device at any time
  - Sensitive data objects: new OS abstraction for data management
  - Idle eviction: garbage-collector that evicts unused SDO data

- CleanOS brings new view on data security: minimize and audit exposure of sensitive data to attack

Agenda

- Reminder of previously pitched major results
  - Major result 1: Stable multithreading (N-Schedule)
  - Major result 2: Multi-sensor architectures (DMCC)
  - Major result 2: Moving target DDoS defense (MiSS)
- Major result 4: Decoy technical transition (DIGIT)
  - Sal Stolfo (Columbia, Allure) will present and demo
- Major result 5: Design and implementation of a ``clean'' operating system (Evade)
  - Roxana Geambasu (Columbia) will present
- Include slides with updates of other major results
  - Will not cover today, will be described at next PI meeting

MiSS Objectives & Challenges
PI: Angelos Stavrou

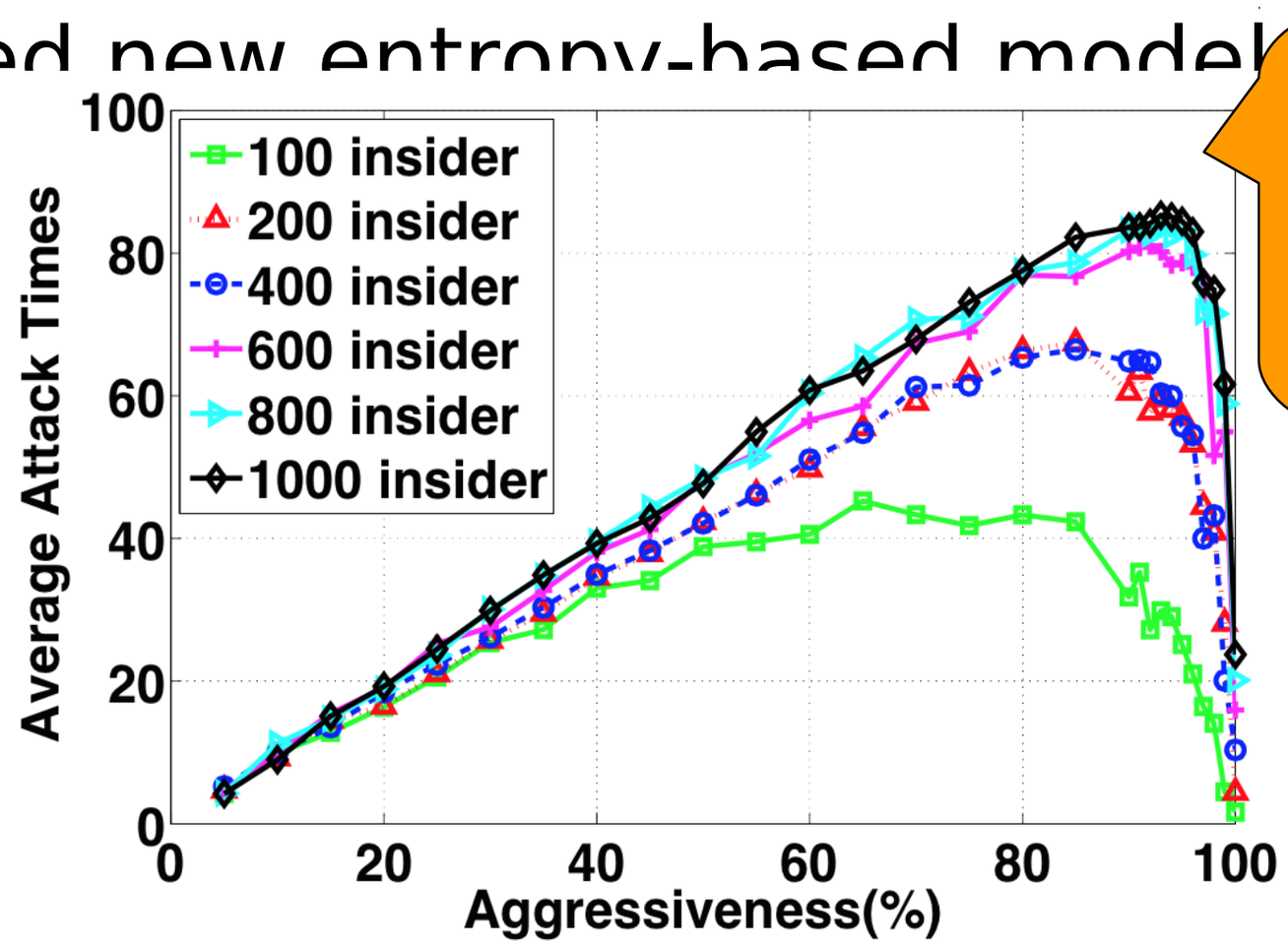
- Enable Seamless Service mobility
  - Mitigation of targeted attacks against application instances
  - Application instances become a “Moving Target” for the adversary
  - Maintain service robustness for legitimate clients
- Support Fast migration of multi-tier and multi-threaded application
  - Eliminate disruptions of service
  - Minimize service downtime

Attacks

Updated theoretical models by removing simplifying assumption that attackers are immediately aggressive

Discovered malicious insider can do more damage using a non-aggressive approach

Developed new entropy-based model
handle n
reputatic
attackers



Maximum potency at 93% aggressive

Build-Out

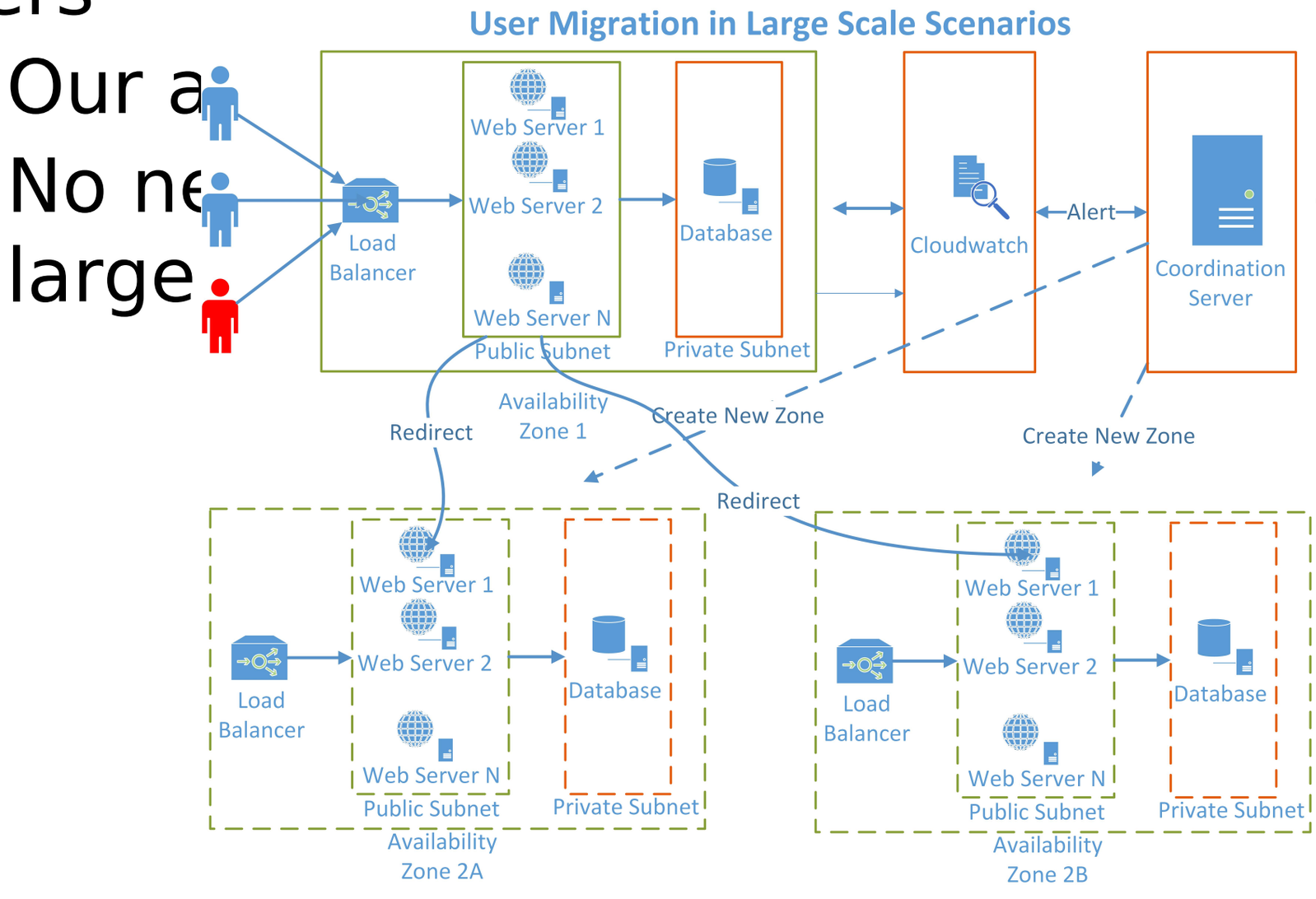
Implemented the full working system on Amazon Cloud

Used Availability Zones (AZ) of multiple web servers

- Our a
- No ne

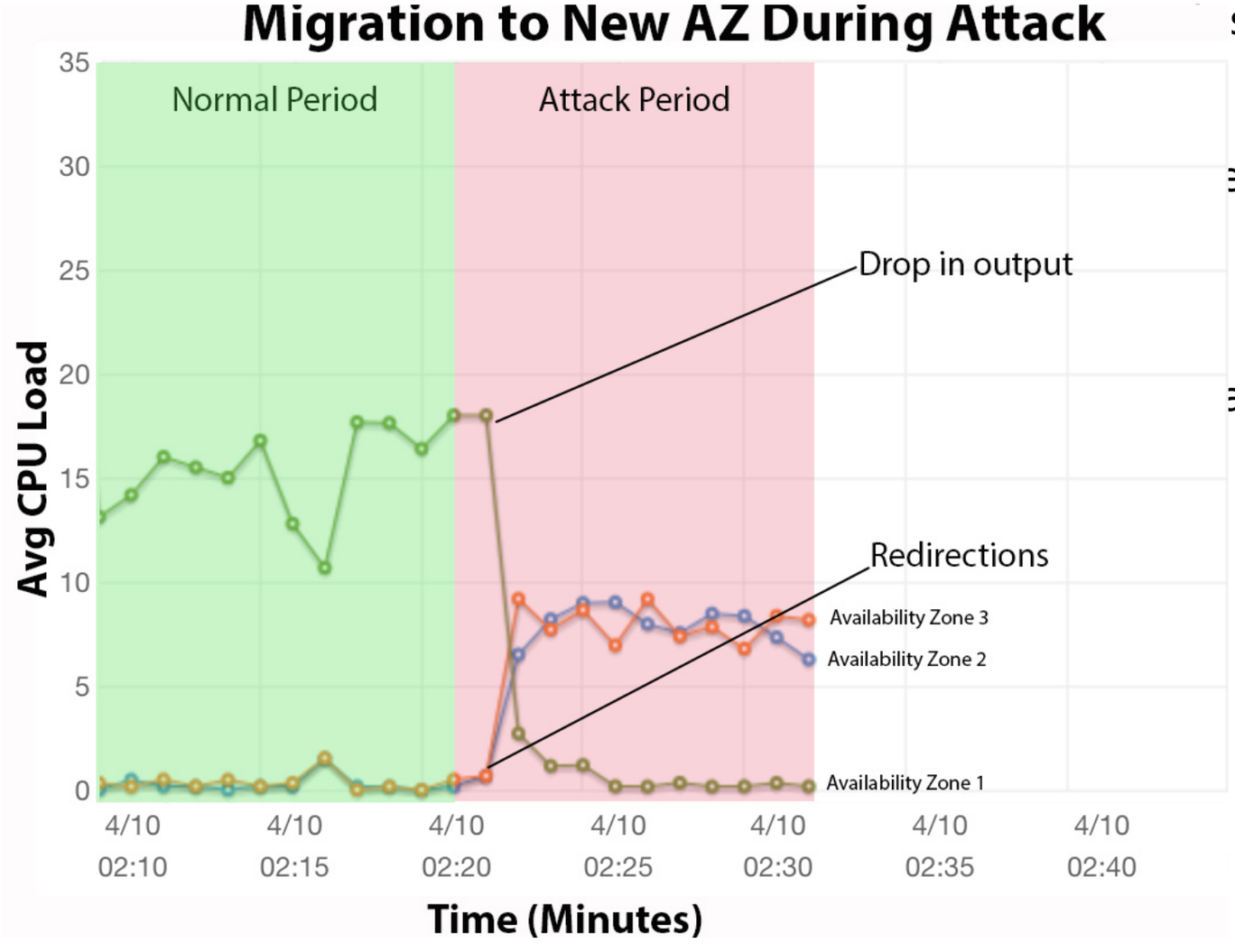
large

in



Future Work

Defending Against SYN Flood



Future Work

stant DDoS detection

- Start response before the servers get saturated

ession maintenance

- Rely on migration
- Requirement for seamless user experience
- Beneficial for labeling users

apid migration

- Use incremental snapshots

Automatic Memoization

Motivation:

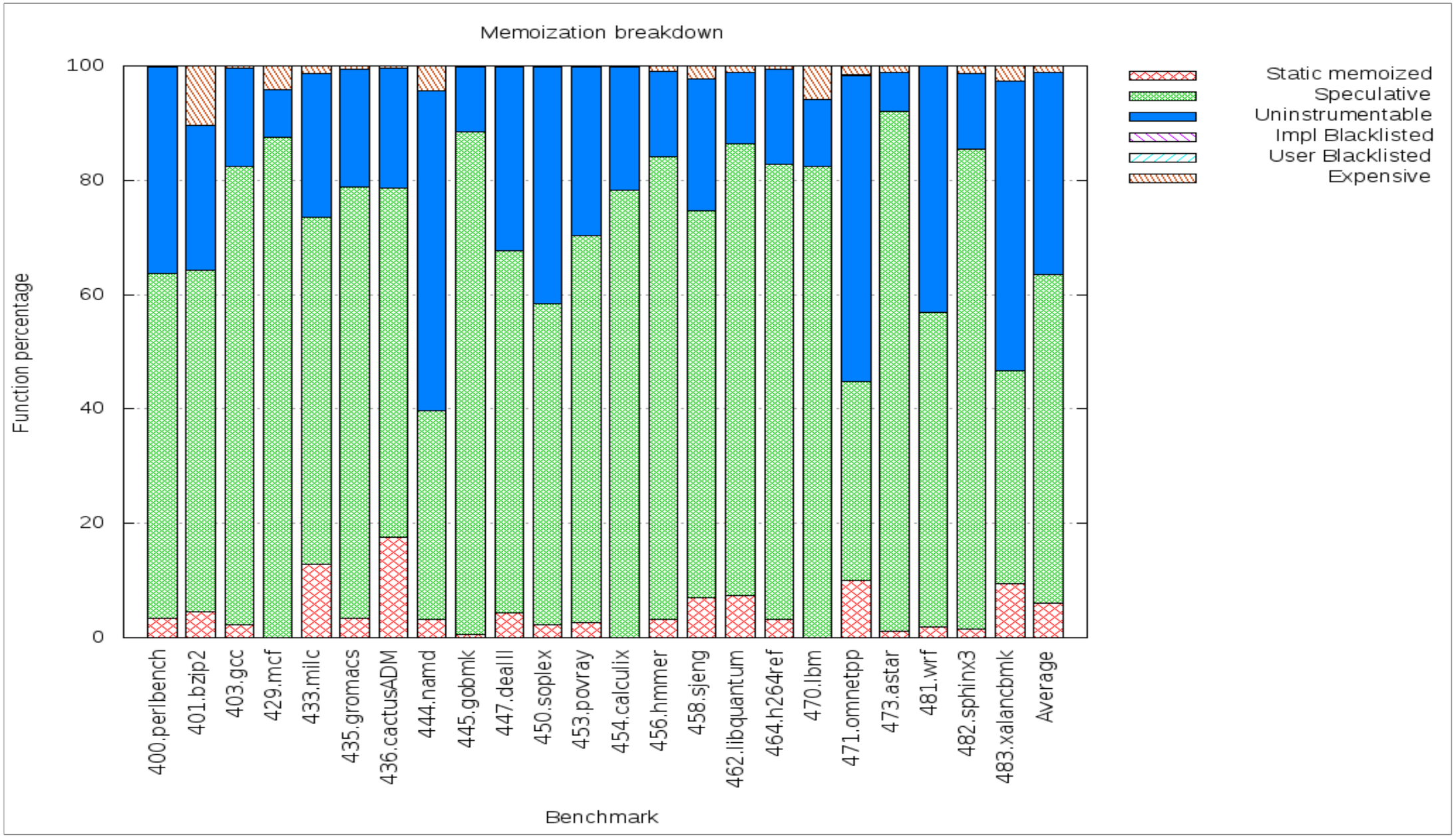
- Provide inexpensive process duplication to obfuscate cloud attack surface

Status:

- Software implementation complete
- Prototyping simple hardware accelerator

Feasibility

SPEC CPU2006 contains large percentages of memoizable code (average 64%)



Hardware acceleration

- Observed limitations in software implementation:
  - Compiler constrained to produce pessimistic, correct code
  - Instrumenting across function boundaries is complex
  - Large dynamic instruction overhead
  - Poor code locality unless instrumentation inlined
  - Static reordering of code to decrease memoization “hazards” does not scale; large code duplication, increased register pressure

Hardware Acceleration, cont

Simple hardware acceleration solves these problems:

- Function boudaries vanish into instruction stream
- Dynamic instruction count reduced; macro operations
- No code locality penalty; instrumentation in pipeline
- Can use special load prefetches to retrieve function inputs aggressively, allowing for more effective memoization coverage.

Challenges:

- Sensitivity to memoization cache capacity and entry size
- Reproducing precise program state
- Choosing performant hash algorithm with low area overhead

RepBox: Enabling Transparent State-machine Replication for General Server Applications

Heming Cui, Rui Gu, Cheng Liu, Junfeng Yang

Background and Motivation

- State machine replication (SMR): attractive to servers
  - Run the same application on a set of machines (or replicas)
  - Enforce the same sequence of inputs among replicas with Paxos
  - Guarantees: same execution state transitions among replicas, highly available despite minor replica failures

- But, existing SMR systems are **untransparent** to servers
  - Problem: **nondeterminism**. Thread interleavings (or *schedules*) can easily cause applications' execution states to diverge
  - Existing SMR systems require server developers to

RepBox: **Rep**lication as a **B**lack B**ox**

- Leverage Parrot [SOSP ‘13], a recent Deterministic Multithreading runtime system, because it:
  - Work with x86 executables, enforce a well-defined round-robin schedule for Pthreads synchronizations
  - Use ***performance hints*** when default schedules are slow
  - Incur merely 12.7% mean overhead on a wide range of 108 popular multithreaded programs on 24-core

- Create a new socket-API-based Paxos consensus interface, easy to plug in server applications

- Provide a scheme for transparently checkpoint and recover server applications by leveraging CRIU

Evaluation Summary

- Evaluation setup
  - Number of replicas: three, each has 24 CPU cores
  - Applications: Apache, Mongoose, ClamAV, and MediaTomb
  - Workloads: use popular benchmarks or representative workloads
  - Concurrency: choose # threads at each server's peak performance

- Evaluation results
  - Easy to use: all four servers can just plug and play without modification; Apache and Mongoose required adding two lines of performance hints each (for performance optimization only)

RepBox has Broad Applications

- Build new defense techniques
  - E.g., pro-actively diversifying schedules to handle concurrency bugs that may lead to exploits [HotPar '12]
- Benefit existing techniques that also suffer from nondeterminism
  - E.g., byzantine fault-tolerance and record/replay
- Tolerate the slowdown of reliability and security tools by running the tool on a backup node
  - E.g., data race detection and data flow tracking

CIFT: Cloud-wide Data Flow Tracking

Goal: Track data propagation across a cloud

Outcomes:

CloudFence [RAID'13]

Per-user data flow tracking for cloud-resident user data auditing

Cross-process and cross-host data tracking

Cloudopsy [HCI'13]

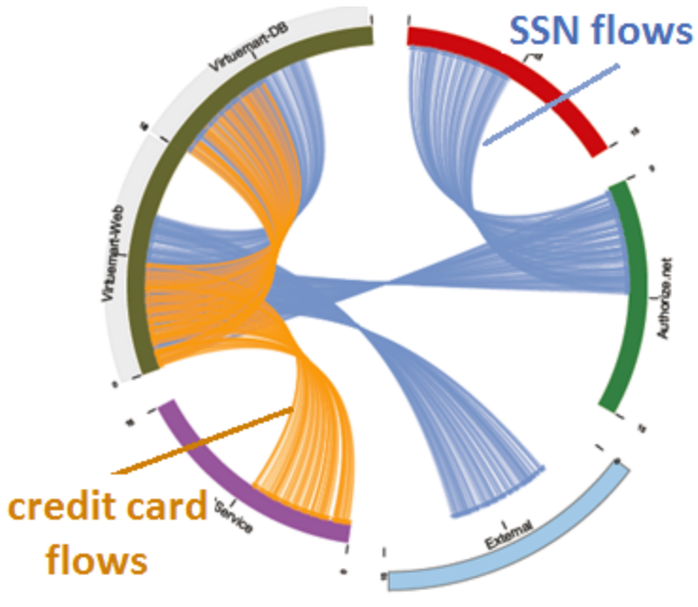
Visualization of audit log information

User GUI for CloudFence

ShadowReplica [CCS '13]

Parallelized dynamic binary instrumentati

4x speed up for data flow tracking compar
to Libdft (used in the CloudFence prototyp



CIFT: Recent Progress

CloudFence++ [to be submitted]

Moved from binary-level to interpreter-level

Prototype: Python interpreter with DFT enabled in AppScale (open-source AppEngine equivalent)

PaaS-wide tracking

Contrary to the first Cloudfence prototype:

- Developers can use it with minimal application modifications
- Enforcing DFT for all sensitive incoming data automatically, without any cooperation from the site of the developers. Hence, defending against malicious developers who might want to manipulate user data behind the back of the users
- Improved runtime DFT performance

Evaluation with representative applications

To be integrated with Cloudopsy

Demo available at the next meeting!

DIGIT: Decoy Data and Computation

Goal: create uncertainly to the adversary as to what is real

Outcomes:

Application-level decoys [ARO'13]

Decoy service interactions, mixed with real ones

Cloud back-end does not know which actions/data are real

Focus on application *behavior*, in addition to data context

Sample request:
GET /search?q=sushi HTTP/1.1

Generated decoys:
GET /search?q=books HTTP/1.1
GET /search?q=burgers HTTP/1.1
GET /search?q=windows HTTP/1.1

Password decoys [CCS'13]

Indistinguishable form actual ones

Password leak detection: use of a decoy password triggers an alert

Mitigate password reuse habits: an attacker has to try all decoys