AIQL: Enabling Efficient Attack Investigation from System Monitoring Data

Peng Gao[1], Xusheng Xiao[2], Zhichun Li[3], Kangkook Jee[3], Fengyuan Xu[4], Sanjeev R. Kulkarni[1], Prateek Mittal[1]

[1]Princeton University

[2]Case Western Reserve University

[3]NEC Laboratories America, Inc.

[4]Nanjing University

Introduction

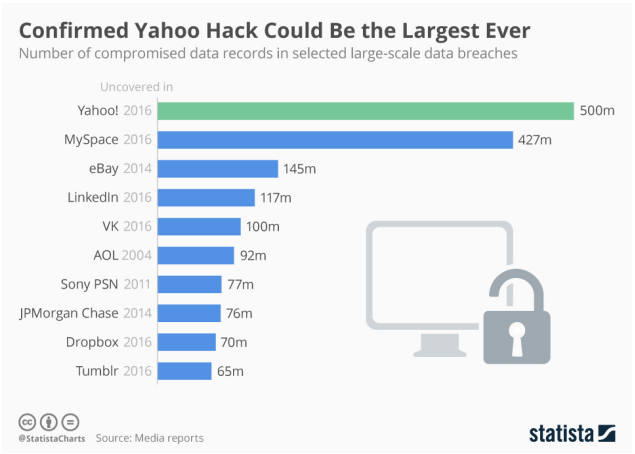
Case Western Reserve University
Princeton University
NEC Laboratories America
Relentless passion for innovation

The Equifax Data Breach



Impact of Advanced Persistent Threat (APT) Attack





- Advanced: sophisticated techniques, e.g., exploiting multiple vulnerabilities
- Persistent: adversaries are continuously monitoring and stealing data from the target
- Threat: strong economical or political motives

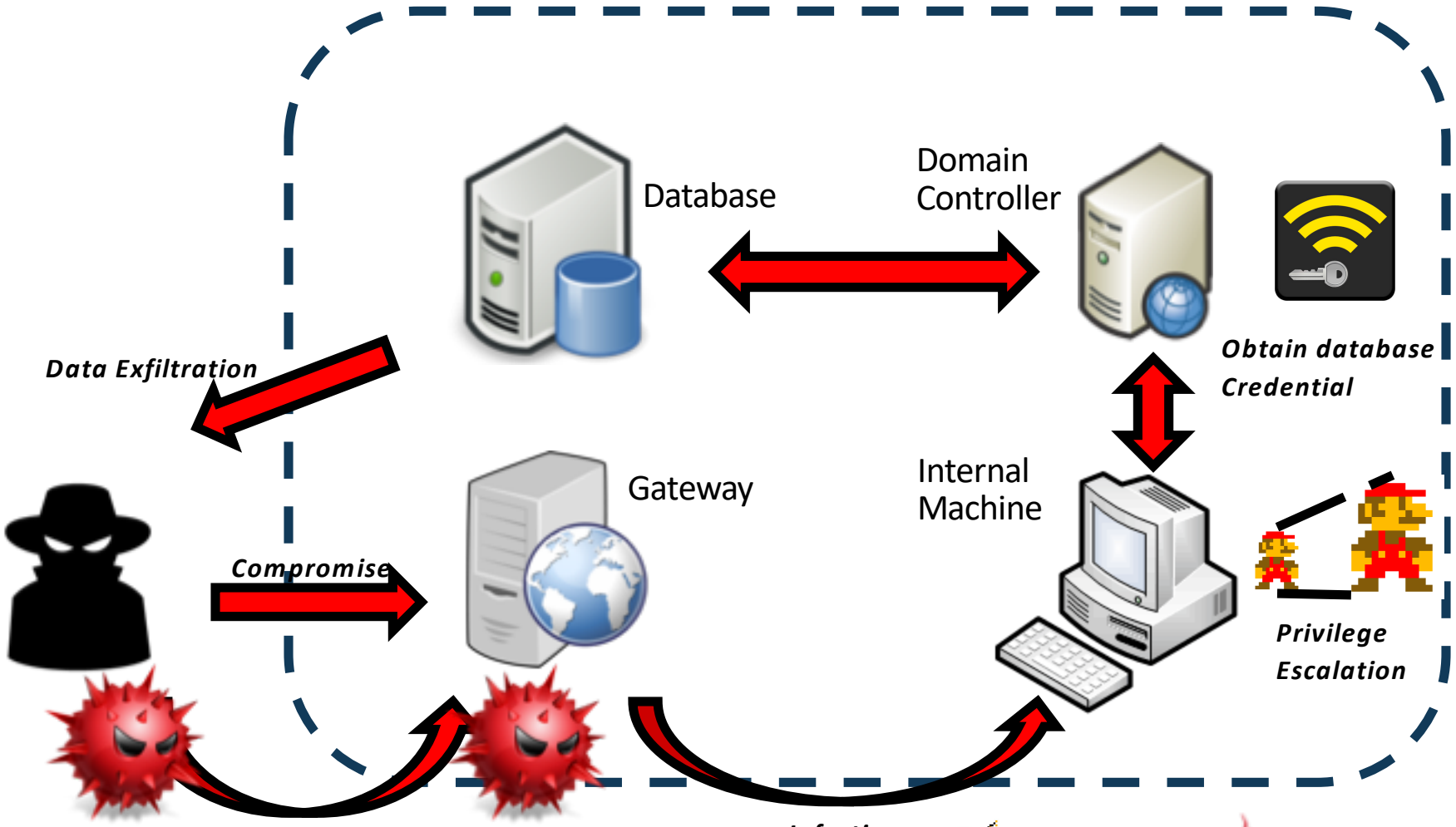


Case Western Reserve University

Princeton University

NEC Laboratories America
Relentless passion for innovation

Advanced Persistent Threat (APT)



Case Western Reserve University

Princeton University

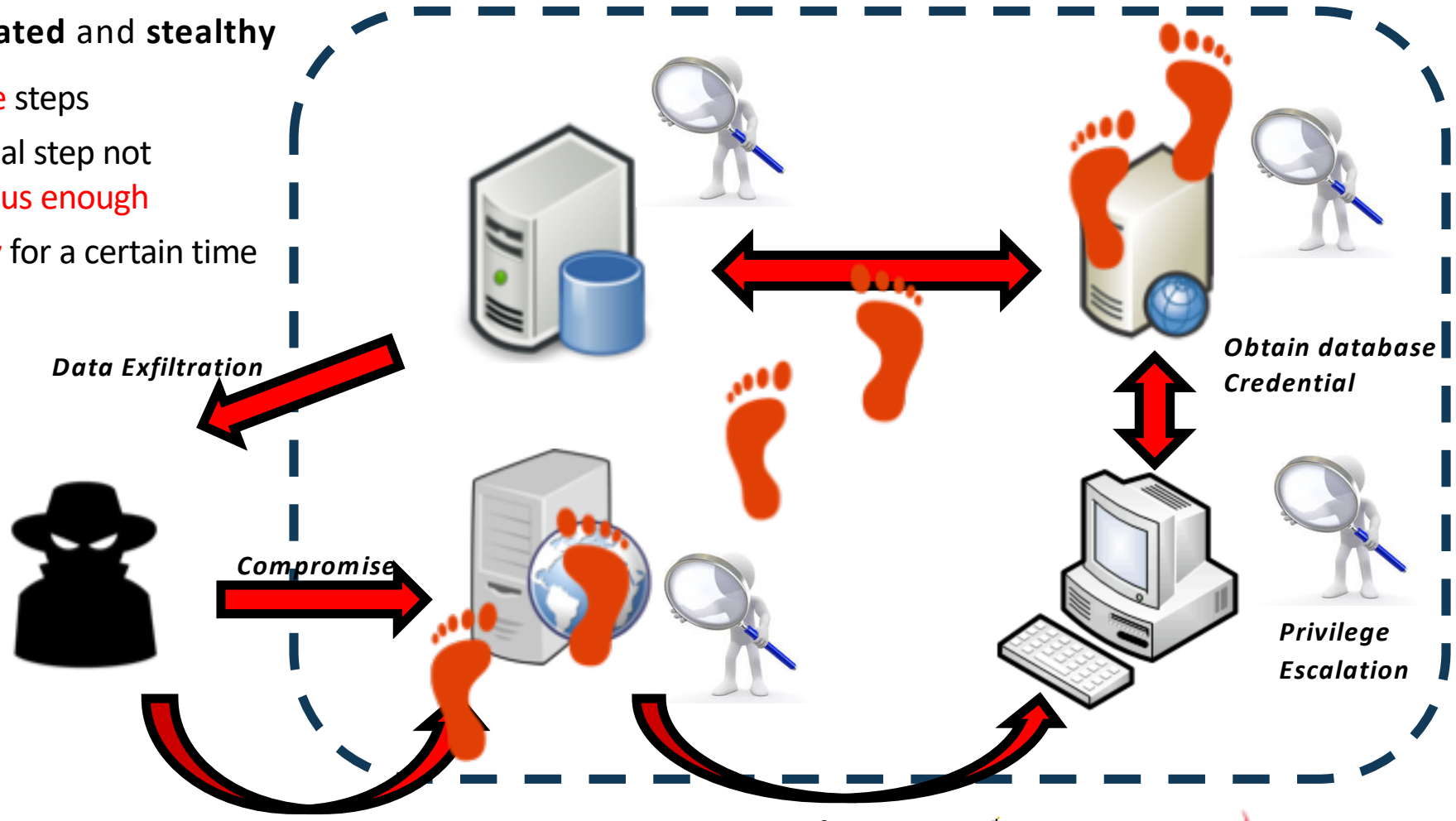
Symantec FireEye

NEC Laboratories America
Relentless passion for innovation

Challenges and Opportunities

Sophisticated and **stealthy**

- Multiple steps
- Individual step not suspicious enough
- Stealthy for a certain time



Case Western Reserve University

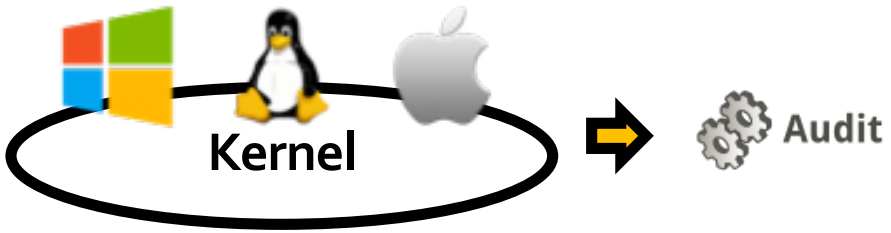
Princeton University

Symantec FireEye

NEC Laboratories America
Relentless passion for innovation

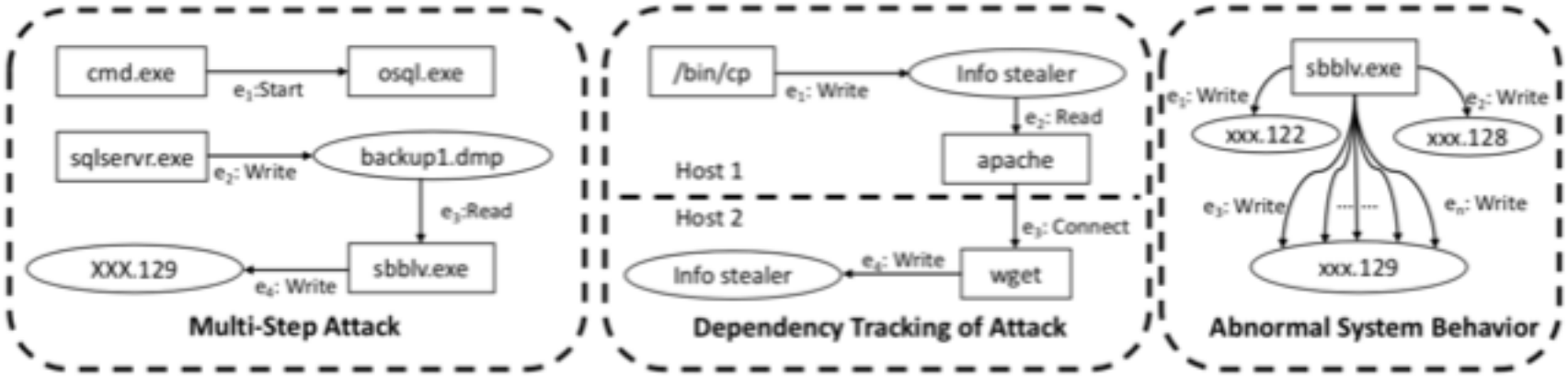
Ubiquitous System Monitoring

- Recording system behaviors from kernel
  - ➢Unified structure of logs: not bound to applications



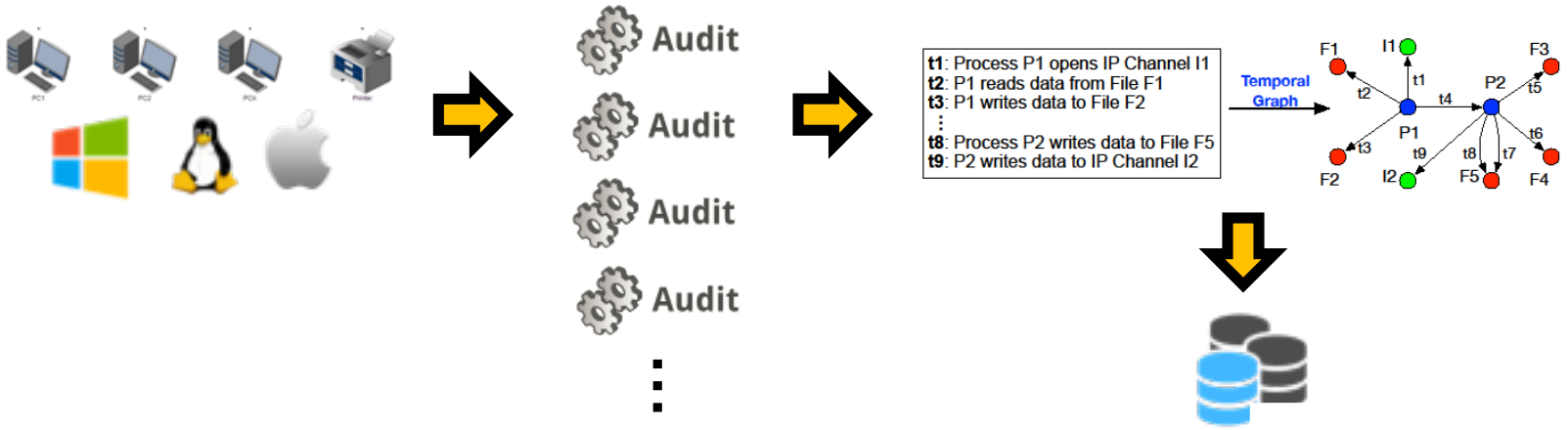
- System activities w.r.t. system resources
  - ➢System resources: processes, files, network connections
  - ➢System activities (system events): <subject, operation, object>,
    - e.g., proc p1 read file f1
- Enabling attack investigation via querying the collected data
  - ➢Interactive query: recovering attack steps in multiple hosts

Challenge 1: Attack Behavior Specification



- **Multi-step attack**: multiple system activities and their relationships
- **Dependency tracking**: chaining constraints among system activities
- **Abnormal system behavior**: sliding windows and statistical aggregation methods

Challenge 2: Timely "Big Data" Security Analysis



- Collect and store system monitoring data for hosts in an organization (~50 GB for 100 hosts per day)
  - Data storage optimization
- Query data in a central server for attack investigation
  - Query execution optimization

AIQL System



- Novel query system for attack investigation
  - Built on top of existing mature tools
    - System-level monitoring tools: auditd, ETW, DTrace
    - Relational databases: PostgreSQL, Greenplum
  - Implementation: ~50,000 lines of Java code
- **System Components <- System Optimizations**
  - Data Collection and Storage <- Data Model and Storage
  - AIQL Language Parser <- Domain-Specific Query Language
  - Query Execution Engine <- Relationship-based Scheduling

Data Collection and Storage

CASE WESTERN RESERVE UNIVERSITY EST. 1826
PRINCETON UNIVERSITY
NEC Laboratories America
Relentless passion for innovation

Data Collection

- Data collection agent: system calls as a sequence of system events
  - Windows: Event Tracing for Windows (ETW)
  - Linux: Audit Framework (auditd)
  - Mac: DTrace
- Collect critical attributes for security analysis

Table 1: Representative attributes of system entities

Entity	Attributes
File	Name, Owner/Group, VolID, DataID, etc.
Process	PID, Name, User, Cmd, Binary Signature, etc.
Network Connection	IP, Port, Protocol

Table 2: Representative attributes of system events

Operation	Read/Write, Execute, Start/End, Rename/Delete
Time/Sequence	Start Time/End Time, Event Sequence
Misc.	Subject ID, Object ID, Failure Code

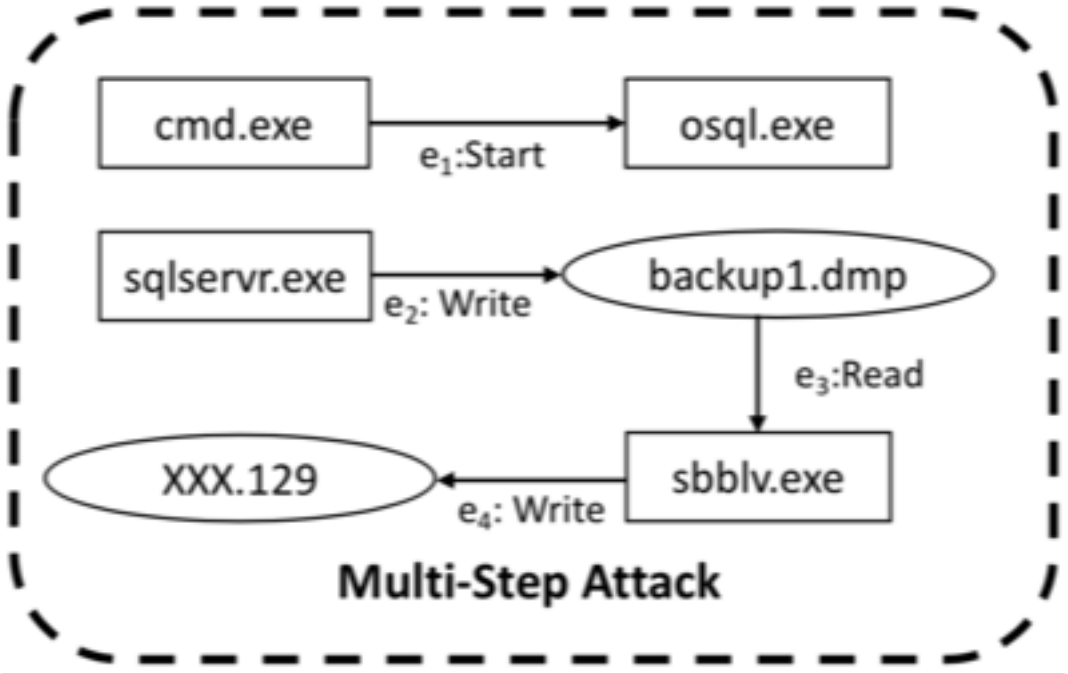
Data Storage

- Store data in relational databases powered by PostgreSQL
  - Challenges in handling high ingestion rate
- Data storage optimizations
  - Data deduplication and in-memory indexes
    - Entity tables: file, process, network connection
    - Event tables: file event, process event, network event
  - Batch commit
  - Time and space partitioning
  - Hypertable

AIQL (Attack Investigation Query Language)

Case Western Reserve University
Princeton University
NEC Laboratories America
Relentless passion for innovation

Multievent AIQL Query



- Multiple system events involved in the attack
- Temporal order of events: e1->e2->e3->e4

Multievent AIQL Query

```
1 (at "mm/dd/2017")
2 agentid = xxx // SQL database server (obfuscated)
3 proc p1["%cmd.exe"] start proc p2["%osql.exe"] as
      evt1
4 proc p3["%sqlservr.exe"] write file f1["%backup1.dmp"
      ] as evt2
5 proc p4["%sbblv.exe"] read file f1 as evt3
6 proc p4 read || write ip i1[dstip="XXX.129"] as evt4
7 with evt1 before evt2, evt2 before evt3, evt3 before
      evt4
8 return distinct p1, p2, p3, f1, p4, i1
```

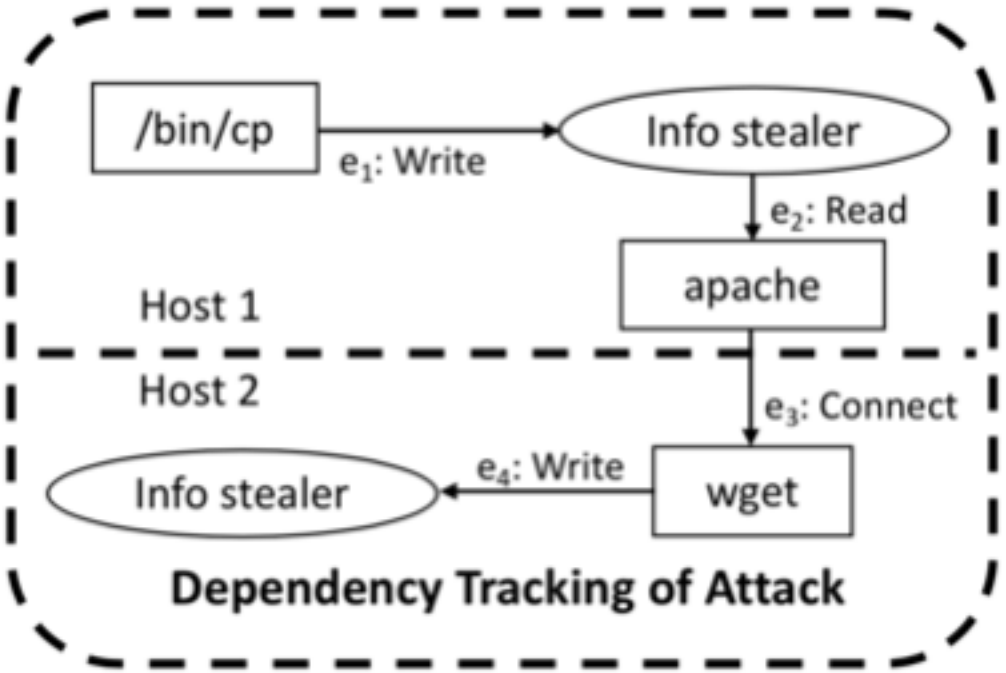
exe_name = “%cmd.exe”

name = “%backup1.dmp”

p1.exe_name, p2.exe_name, p3.exe_name, f1.name, p4.exe_name, i1.dst_ip

- Global constraints: e.g., agent ID, time window
- Event pattern: <subject, operation, object>
- Temporal relationship: e.g., enforce the event order
- Attribute relationship: e.g., two events linked by the same entity
- Syntax shortcuts: e.g., context-aware attribute inference

Dependency AIQL Query



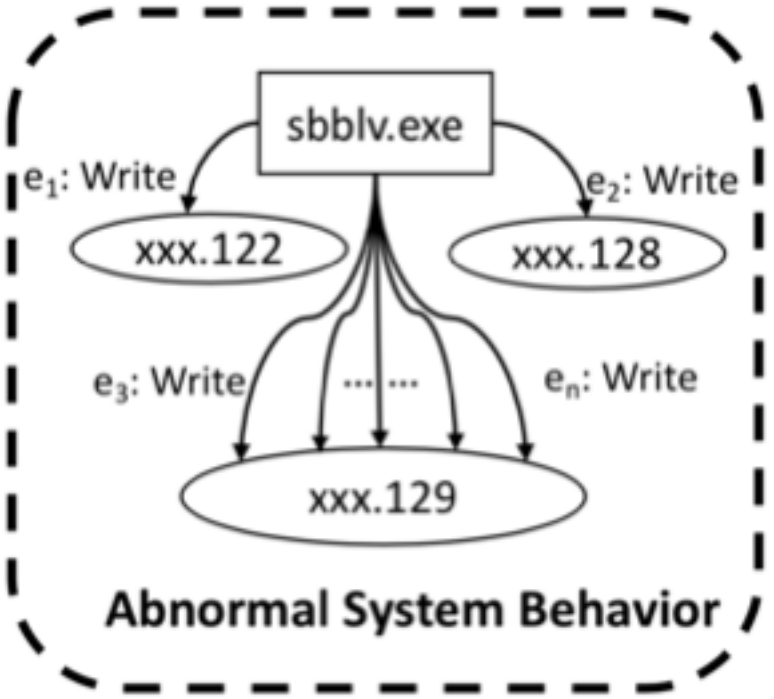
- Chain constraints among events
- Dependency tracking for attack causality analysis

Dependency AIQL Query

```
1 (at "01/01/2017")
2 forward: proc p1["%/bin/cp%", agentid = 2] ->[write]
      file f1["/var/www/%info_stealer%"]
3 <-[read] proc p2["%apache%"]
4 ->[connect] proc p3[agentid=3] // tracking across
      host
5 ->[write] file f2["%info_stealer%"]
6 return f1, p1, p2, p3, f2
```

- Dependency tracking direction: forward/backward, ->/<-
- Cross host dependency tracking

Anomaly AIQL Query



- Frequency-based anomaly models

Anomaly AIQL Query

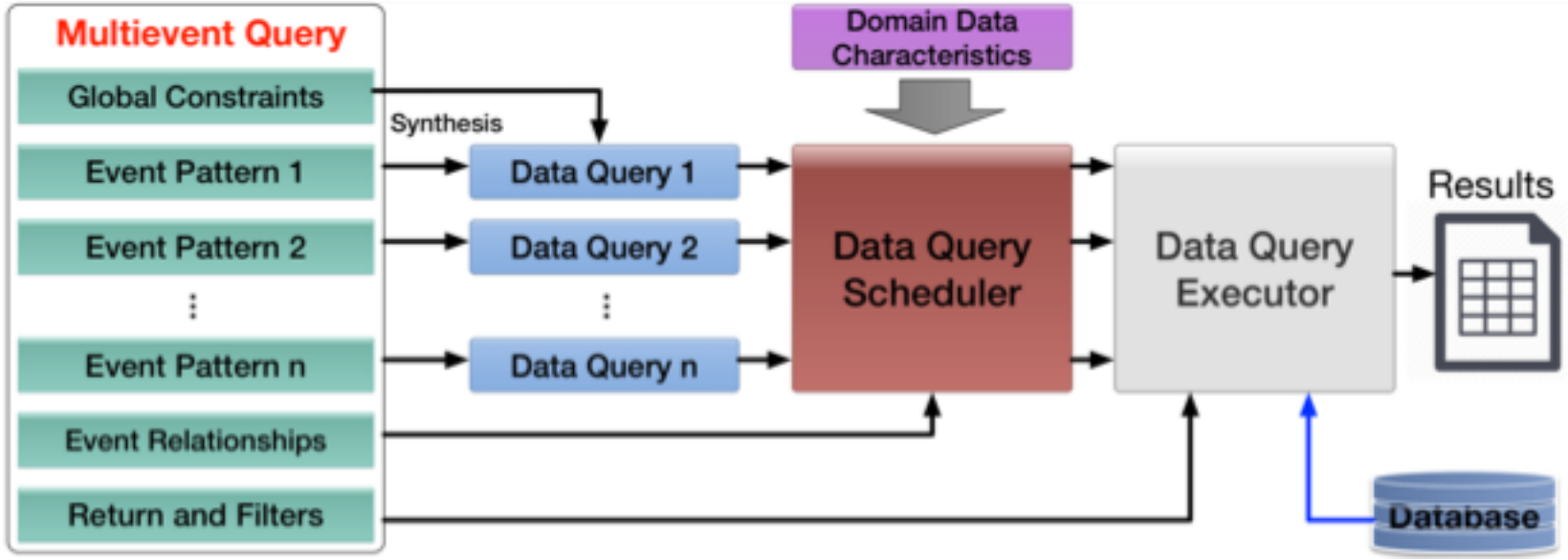
```
1 (at "mm/dd/2017") // date (obfuscated)
2 agentid = xxx // SQL database server (obfuscated)
3 window = 1 min, step = 10 sec
4 proc p write ip i[dstip="XXX.129"] as evt
5 return p, avg(evt.amount) as amt
6 group by p
7 having (amt > 2 * (amt + amt[1] + amt[2]) / 3)
```

- Sliding windows
- Access to history states
- Frequency-based anomaly models: e.g., SMA

AIQL Query Execution Engine

Case Western Reserve University
Princeton University
NEC Laboratories America
Relentless passion for innovation

Execution of Multievent Query



- Synthesize a SQL data query for every event pattern
- Schedule the data queries using domain-specific optimizations

Data Query Scheduler

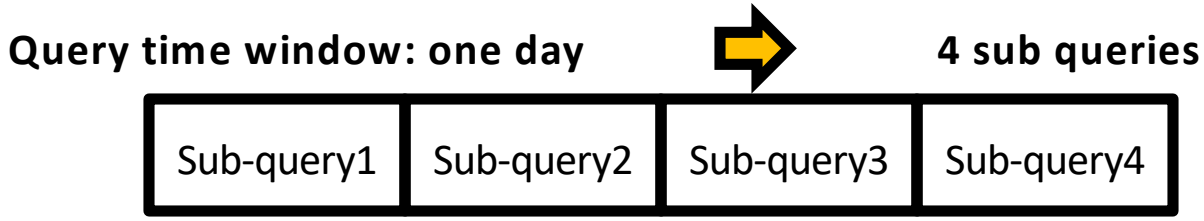
- Leverage event relationships for optimizing search strategies
  - ➢Prioritize event search based on estimated pruning power
  - ➢Prune search space of related events

```
proc p1 write file f1['%pwdump7.exe%'||'%libeay32.dll%'] as evt2
proc p2 execute file f1 as evt3
```

- Leverage domain-specific characteristics of system monitoring data for parallel search
  - ➢Time window partition

Time Window Partition

```
1 (at "mm/dd/2017")
2 agentid = xxx // SQL database server (obfuscated)
3 proc p1["%cmd.exe"] start proc p2["%osql.exe"] as
      evt1
4 proc p3["%sqlservr.exe"] write file f1["%backup1.dmp"
      ] as evt2
5 proc p4["%sbblv.exe"] read file f1 as evt3
6 proc p4 read || write ip i1[dstip="XXX.129"] as evt4
7 with evt1 before evt2, evt2 before evt3, evt3 before
      evt4
8 return distinct p1, p2, p3, f1, p4, i1
```

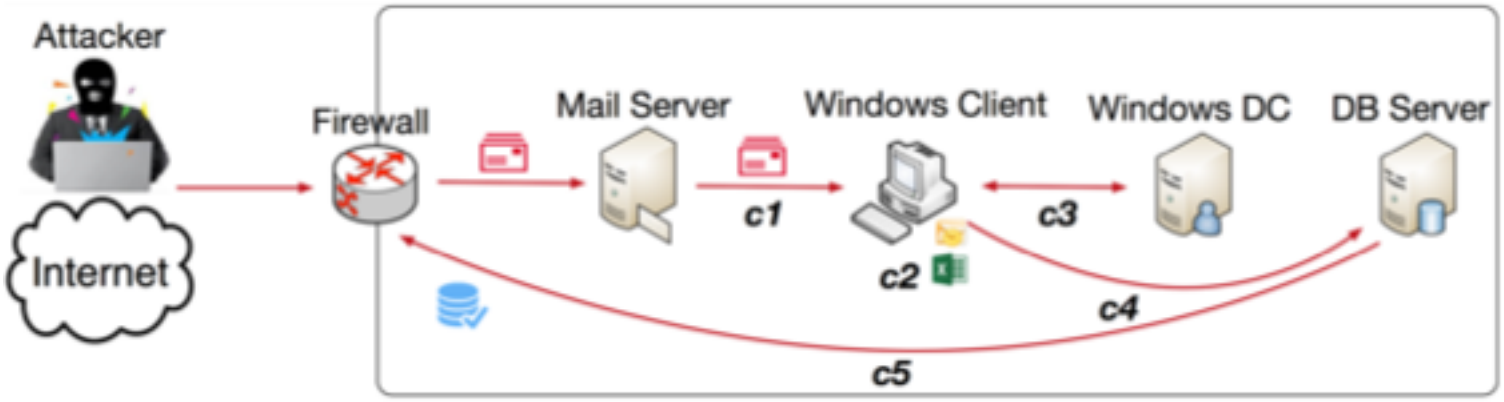


- Uniform time window partition
  - ➢ # of sub queries depending on I/O capacity (concurrent read)
- Other potential partition: uniform workload …

Case Study and Evaluation

Case Western Reserve University
Princeton University
NEC Laboratories America
Relentless passion for innovation

Case Study: APT Investigation



- **c1 Initial Compromise**: Attacker sends a crafted e-mail to the victim, which contains an Excel file with a malicious macro embedded
- **c2 Malware Infection**: Victim opens the file and runs the macro, which downloads and executes a malware to open a backdoor
- **c3 Privilege Escalation**: Attacker runs the database cracking tool to obtain database credentials
- **c4 Penetration into Database Server**: Attacker drops another malware to open another backdoor
- **c5 Data Exfiltration**: Attacker dumps the database content and sends the dump back to his host

Investigation of Step c5

- Deploy anomaly detectors on the database server to detect large amount of data transfer
  - ➢ Identify a suspicious IP “XXX.129”
- Construct an anomaly AIQL query to find the processes that access “XXX.129”
  - ➢ “sbblv.exe” (p) writes to “XXX.129”

```
1 (at "mm/dd/2017") // date (obfuscated)
2 agentid = xxx // SQL database server (obfuscated)
3 window = 1 min, step = 10 sec
4 proc p write ip i[dstip="XXX.129"] as evt
5 return p, avg(evt.amount) as amt
6 group by p
7 having (amt > 2 * (amt + amt[1] + amt[2]) / 3)
```

Investigation of Step c5

- Find the files accessed by “sbblv.exe”
  - “sbblv.exe” reads from “backup1.dmp” (f1)

```
(at "mm/dd/2017")
agentid = xx // SQL database server (obfuscated)
proc p1["%sbblv.exe"] read || write file f1 as evt1
proc p1 read || write ip i1[dstip="XXX.129"] as evt2
with evt1 before evt2
return distinct p1, f1, i1, evt1.optype, evt1.access
```

- Creation processes of “backup1.dmp”
  - “sqlservr.exe” (p1) writes to “backup1.dmp”

```
(at "mm/dd/2017")
agentid = xxx // SQL database server (obfuscated)
proc p1 write file f1["%backup1.dmp"] as evt1
proc p2["%sbblv.exe"] read file f1 as evt2
proc p2 read || write ip i1[dstip="XXX.129"] as evt3
with evt1 before evt2, evt2 before evt3
return distinct p1, f1, p2, i1
```

Investigation of Step c5

- Further verification: “cmd.exe” starts “osql.exe” (OSQL utility)
- Complete AIQL query for c5 (data exfiltration)

```
(at "mm/dd/2017")
agentid = xxx // SQL database server (obfuscated)
proc p1["%cmd.exe"] start proc p2["%osql.exe"] as
     evt1
proc p3["%sqlservr.exe"] write file f1["%backup1.dmp"
     ] as evt2
proc p4["%sbblv.exe"] read file f1 as evt3
proc p4 read || write ip i1[dstip="XXX.129"] as evt4
with evt1 before evt2, evt2 before evt3, evt3 before
     evt4
return distinct p1, p2, p3, f1, p4, i1
```

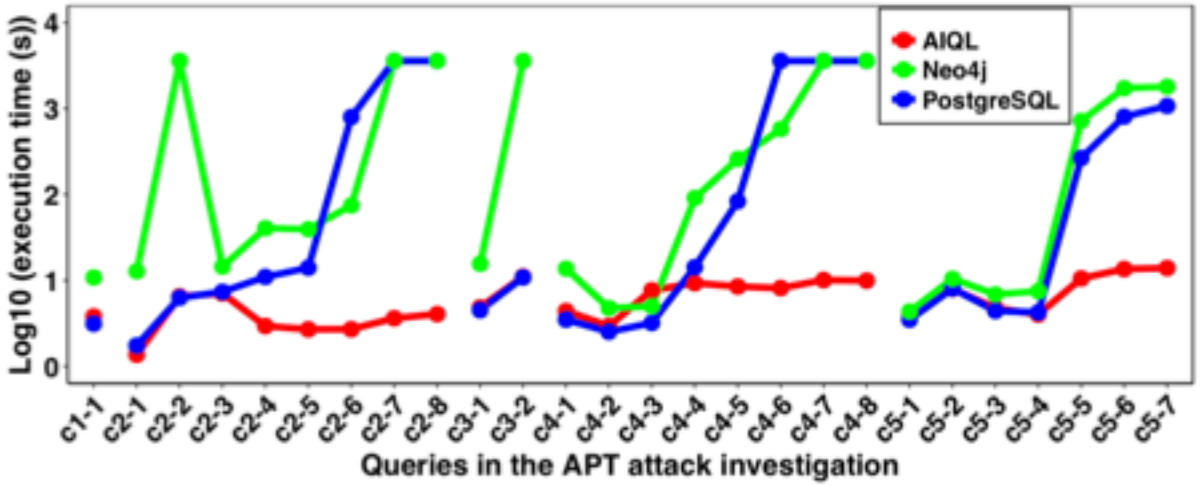
End-to-End Efficiency for Case Study

- 27 queries, touching 119 GB of data (422 million system events)

Table 3: Aggregate statistics for case study

Attack Step	# of Queries	# of Evt Patterns	AIQL (s)	PostgreSQL (s)	Neo4j (s)
c1	1	3	3.8	3.1	10.8
c2	8	27	31.0	8038.7	10981.7
c3	2	4	15.9	15.3	3615.6
c4	8	35	61.0	10906.7	8150.6
c5	7	18	58.8	2166.5	4285.4
All	26	87	170.5	21130.3	27044.1

~3 min 124x slower 157x slower



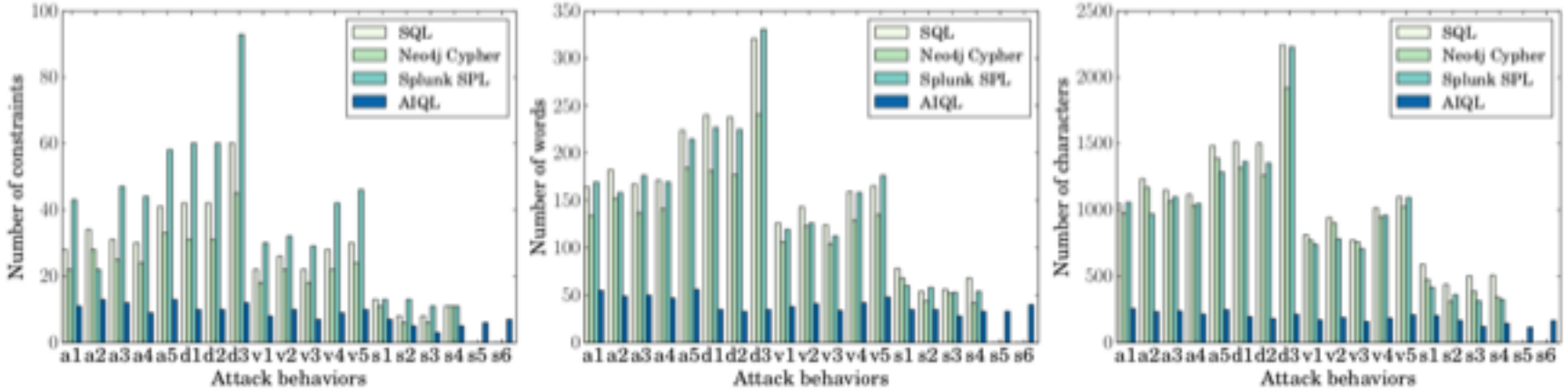
Case Western Reserve University

Princeton University

NEC Laboratories America
Relentless passion for innovation

Conciseness Evaluation

- 19 queries on four major types of attack behaviors, touching 738 GB of data (2.1 billion events)
  - **a1-a5**: Multi-step attack behaviors
  - **d1-d3**: Dependency tracking behaviors
  - **v1-v5**: Real-world malware behaviors (from Virussign)
  - **s1-s6**: Abnormal system behaviors



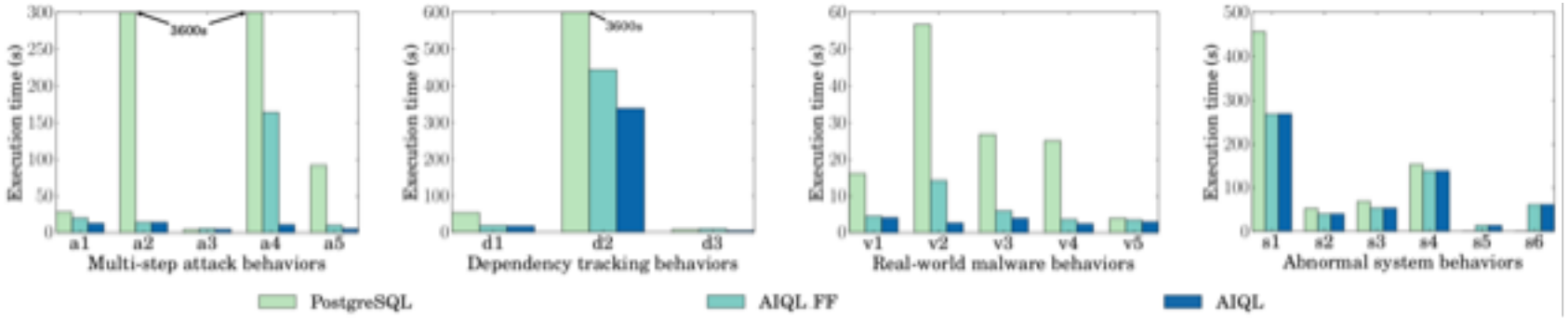
(a) Number of constraints **(b)** Number of words **(c)** Number of characters

Other languages vs. AIQL: 2.4x more constraints, 3.1x more words, and 4.7x more characters

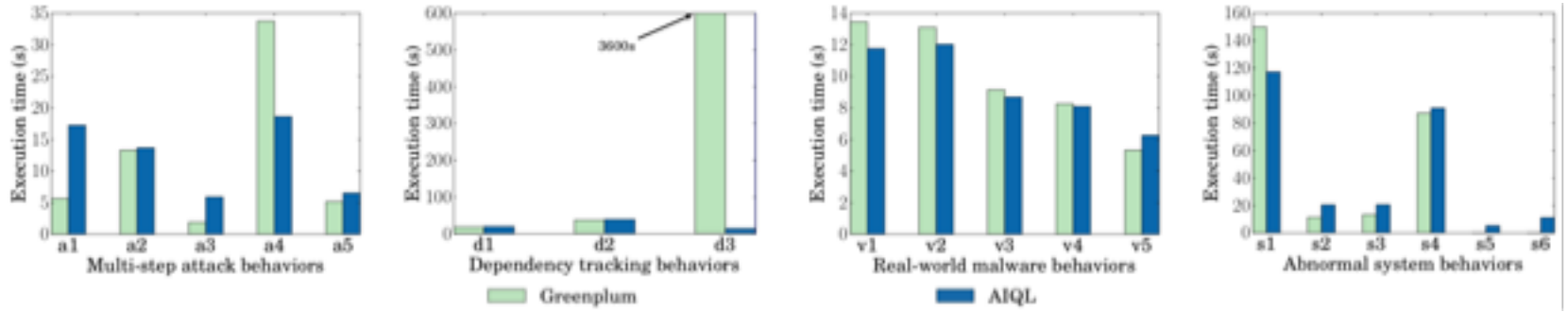
Case Western Reserve University · Princeton University · NEC Laboratories America — Relentless passion for innovation

Scheduling Efficiency in Our Optimized Storage

- Efficiency in PostgreSQL
  - ➢ 40x speedup



- Efficiency in Greenplum
  - ➢ 16x speedup



Case Western Reserve University

Princeton University

NEC Laboratories America
Relentless passion for innovation

Conclusion

CASE WESTERN RESERVE UNIVERSITY EST. 1826
PRINCETON UNIVERSITY
NEC Laboratories America
Relentless passion for innovation

Conclusion

- **AIQL**, a system for efficient attack investigation from system monitoring data
  - Expressive and concise domain-specific query language
  - Efficient query execution based on domain-specific data characteristics

- All queries, the investigation details, and a system demo video are available in project website
  - https://sites.google.com/site/aiqlsystem/
  - https://youtu.be/5v3Tk6Y6YkQ

Q & A

Thank you!