AIQL: Enabling Efficient Attack Investigation from System Monitoring Data

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

1 PRINCETON UNIVERSITY 2 CASE WESTERN RESERVE UNIVERSITY 3 NEC Laboratories America Relentless passion for innovation 4 NANJING UNIVERSITY

USENIX ATC '18

Advanced Persistent Threat (APT) Attack

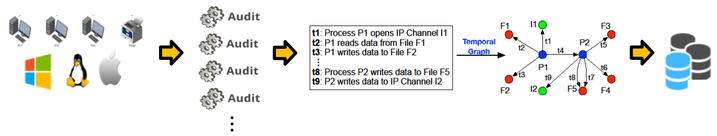
APT attacks have plagued many well-protected businesses

- Advanced: sophisticated techniques exploiting multiple vulnerabilities
- Persistent: continuously monitoring and stealing data from target
- Threat: strong economical or political motives

Ubiquitous System Monitoring

System monitoring records system events from kernels in a unified structure of logs (not bound to applications)

- System activities (system events): <subject, operation, object>



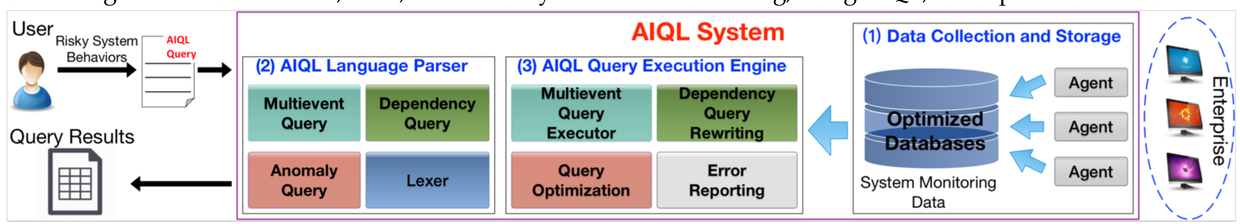
Challenges in enabling efficient attack investigation

- Attack behavior specification
- 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 for attack investigation => query execution optimization

AIQL System

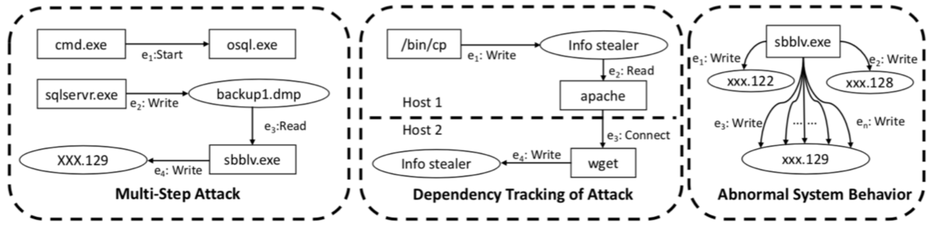
Novel query system for attack investigation (~50K lines of Java code)

- Built on top of existing mature tools: auditd, ETW, DTrace for system-level monitoring; PostgreSQL, Greenplum for relational databases



AIQL (Attack Investigation Query Language) Design

AIQL supports the specification of three types of attack behaviors. Typical constructs: event patterns (subject-operation-object), global constraints, attribute/temporal relationships, constraint chaining, sliding windows and history state access, syntax shortcuts (e.g., attribute inference)




```
(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
```

Multievent AIQL query

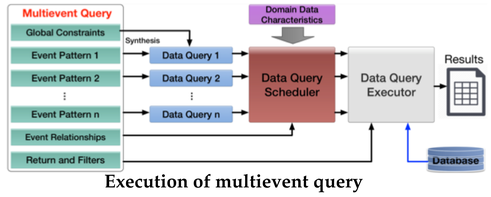
```
(at "01/01/2017")
forward: proc p1["%/bin/cp%", agentid = 2] ->[write]
     file f1["/var/www/%info_stealer%"]
<-[read] proc p2["%apache%"]
->[connect] proc p3[agentid=3] // tracking across
     host
->[write] file f2["%info_stealer%"]
return f1, p1, p2, p3, f2
```

Dependency AIQL query

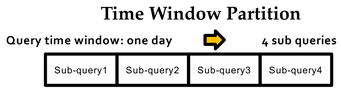
```
(at "mm/dd/2017") // date (obfuscated)
agentid = xxx // SQL database server (obfuscated)
window = 1 min, step = 10 sec
proc p write ip i[dstip="XXX.129"] as evt
return p, avg(evt.amount) as amt
group by p
having (amt > 2 * (amt + amt[1] + amt[2]) / 3)
```

Anomaly AIQL query

Query Execution Engine

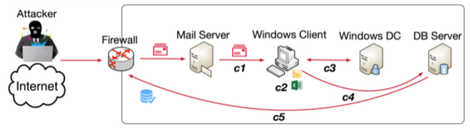


Execution of multievent query



- Synthesize a SQL data query for every event pattern
- Schedule the data queries using domain-specific optimizations
  - Leverage event relationships for optimizing search strategies
    - Prioritize event search based on estimated pruning power
    - Prune search space of related events
  - Leverage domain-specific characteristics of data for parallel search
    - Time window partition

Case Study: APT Attack Investigation



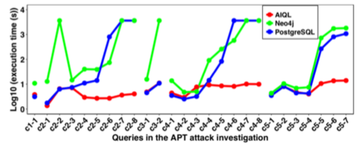
- c1 Initial Compromise
- c2 Malware Infection
- c3 Privilege Escalation
- c4 Penetration into Database Server
- c5 Data Exfiltration

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

- As the attack behaviors become more complex, SQL and Cypher queries become verbose with many joins and constraints
- AIQL takes less than 3 minutes for the entire investigation process , 127x faster than PostgreSQL, 157x faster than Neo4j

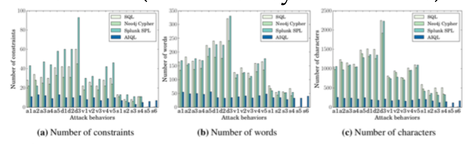
Table 3: Aggregate statistics for case study

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



Conciseness and Scheduling Efficiency

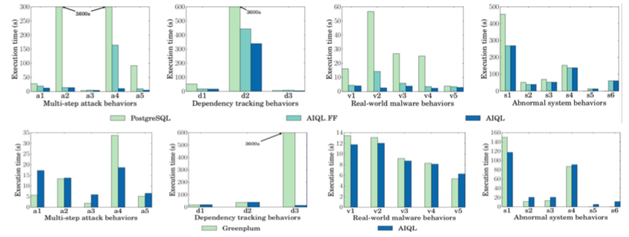
19 queries on four major types of attack behaviors, touching 738 GB of data (2.1 billion system events)



- a1-a5: Multi-step attack behaviors
- d1-d3:Dependency tracking behaviors
- v1-v5: Real-world malware behaviors
- s1-s6: Abnormal system behaviors

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

- Both PostgreSQL and Greenplum employ our optimized data storage



40x faster than PostgreSQL (optimized)

16x faster than Greenplum (optimized)