Cloud Data Analytics for Security: Applications, Challenges, and Opportunities

Daphne Yao
Associate Professor of Computer Science
Turner Fellow and L-3 Fellow
Virginia Tech
Motivation: Security/Privacy as Enablers

My past work: Security Methodology Development
Near-0 false alarm enables analysts to focus on real attacks

Ongoing & future work: Intelligent secure systems and platforms that benefit large populations

Enable new infrastructures
Improve quality of life
Enable new discoveries

Enable new discoveries
A Scenario:
Cloud Data Analytics for Organizational Security

- Control-flow Hijacking
- Service Abuse
- Insider Threats
- APT
- Data-oriented programming
- Exploits
Another Scenario: Cloud Data Analytics for Smart Home Security

Origins of spam in a 2014 botnet study:

- Embedded Linux servers
- mini-httpd, apache
- ARM devices, MIPS, Realtek chipset
- Open telnet, an SMTP server

A vision: To lift host protection to the cloud

What have been done in cloud?

- Cloud anti-virus, e.g., Sophos and Symantec
- Protection of the cloud, e.g., VM sandboxing, [CloudDiag 2013]
- Software-as-a-service [Cloud Terminal 2012]

What have been done on host?

- Firewalls, host-based anti-virus
- Isolation, e.g., VMM
- Reference monitor, e.g., SELinux
- Trusted computing, e.g., TPM attestation
- Data-driven anomaly detection
Setup Type 1: the Cloud AV model

Sophos Cloud - Cloud-managed Security
Setup Type 2: Everything in the cloud

YOU'RE PRETTY NEW TO CLOUD STORAGE, AREN'T YOU?
Cloud terminal [Martignoni 2012]
Setup Type 3: Your refrigerator cannot be in the cloud
Drone Control Station Operating System

From NBC news (2013)
http://nbcnews.tumblr.com/post/47882129464#.UzGICChfd38
What does it take to lift program anomaly detection to the cloud?

In Setup Type 3: autonomous host with detection in the cloud
Acknowledgments

Collaborators

- **Drs. Kui Xu** (Google)
- **Xiaokui Shu** (IBM Research)
- **Hao Zhang** (Oracle)

Network causal analysis
- Zhang, Yao, Ramakrishnan. *AI Sec '16, ASIACCS '14, Computers & Security '16*

Global trace analysis
- Shu, Yao, Ramakrishnan. *ACM CCS '15*
- Shu, Yao, Ramakrishnan, Jaeger (journal version under review)

Program analysis in HMM
- Xu, Yao, Ryder, Tian. *IEEE CSF '15*
- Xu, Tian, Yao, Ryder. *IEEE DSN '16*

Unified framework for program AD
- Shu, Yao, Ryder. *RAID 2015*

- US Patent
- ACM CCS Tutorial 2016 on Program Anomaly Detection
- Work featured in Comm. of ACM
Anti-virus Scanning is the First Line of Defense

For files (apps and PDFs), URLs

Cuckoo Sandbox for dynamic analysis

Vtzilla plugin

Number of submissions in a week (March 19, 2017 – March 25, 2017)

File Types

[From VirusTotal]
Code or Behavior Classification is Undecidable

1. Program X
2. main()
3. {
4.   if ! isVirus(X)
5.     then infect;
6.   else goto next;
7. }
8. }
9. }

<table>
<thead>
<tr>
<th>Scanner Thinks</th>
<th>Actual Behavior of X</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsVirus returns True</td>
<td>X chooses not to infect</td>
</tr>
<tr>
<td></td>
<td>Contradicts</td>
</tr>
<tr>
<td>IsVirus returns False</td>
<td>X chooses to infect</td>
</tr>
<tr>
<td></td>
<td>Contradicts</td>
</tr>
</tbody>
</table>

From [Fred Cohen, J. of Virology 1987]
How to detect/prevent zero-day malware/exploits?

Formal verification, Control flow integrity

N-variant, Moving target defense

Anomaly-based detection [D. Denning ’87, Forrest et al. ’96]

(a) Classification

(b) Anomaly detection

[Wressnegger 2013]
Is Typical Insider Trading Detection Anomaly Detection?

<table>
<thead>
<tr>
<th>Purchase Patterns</th>
<th>Sell Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buy low performing stocks</td>
<td>Sell high performing stocks</td>
</tr>
<tr>
<td>Buy before stock prices go up</td>
<td>Sell before stock prices drop</td>
</tr>
<tr>
<td>Purchase followed by purchase</td>
<td>Sell followed by sell</td>
</tr>
</tbody>
</table>

My Work on Anomaly Detection Methodology Development

- Security logs, Network headers, Traffic payloads, System traces, Transaction logs ...

- Program Tracing (Library call, System call, Instruction sequences)

- Program Analysis (static)

- ML/DM (train and test)

- Post Classification Analysis

- Novelty detection

- Binary classification

- Prog analysis
Simplest Program Anomaly Detection: n-gram

A 2-gram example:

<table>
<thead>
<tr>
<th>ioctl()</th>
<th>open()</th>
</tr>
</thead>
<tbody>
<tr>
<td>open()</td>
<td>read()</td>
</tr>
<tr>
<td>read()</td>
<td>setpgid()</td>
</tr>
<tr>
<td>setpgid()</td>
<td>setsid()</td>
</tr>
<tr>
<td>setsid()</td>
<td>fork()</td>
</tr>
</tbody>
</table>

Runtime program trace:

1. From syscall traces of normal program executions (training data)

2. Test data

3. Classification

[Forrest 1996, Wressnegger 2013]
Who Uses Anomaly Detection on Programs/Systems?

- Average **$1.27 million/year** on false alerts by an enterprise.
- **4%** of alerts are investigated, due to high false positives.
- An organization receives an average of **17,000 alerts/week**.

From [Ponemon Institute]

Twitter Anomaly Detection. 
https://blog.twitter.com/2015/
Manual alert confirmation is costly

157 minutes

"We haven’t seen any false positives and nothing going on across our whole infrastructure. This minimize wasting resources on having to respond. Posture is even more valuable for us."

- SCOTT ADAMS
Big Data, Big Bucks

splunk

LOGGLY

NETFLIX

twitter

sumologic

NEX DEFENSE

ThetaRay

SCALYR

loglogic

ALERT LOGIC


LogRhythm

The Security Intelligence Company

greylog

elastic
Challenges: Diverse Normal Behaviors, High FP

False positive rate (1-class SVM on libpcre)

Too low!

Distribution of function calls in libpcre
False alarms & missed detection can be harmful

Voice-recognition based authentication [CITI Taiwan]

Child pornography detection (FP 1 out of 2 billions)

Spam detection

Pavement distress detection w/ sensors
You found some weird data. Are they meaningful?

rPCA [Candès 2009] works well for motion detection in videos

Images from [Wang 2016]
Semantics of Anomalies in Security

Actions of Attacks and Attack Preparations

- Control-flow hijacking
  - Return-oriented programming (ROP)
  - Backdoors

- Control-flag hijacking
  - Data-oriented programming (DOP) (not be detected by CFI)

- Service abuse attacks
  - Denial of Service (DoS)
  - Memory overread

- Workflow/state violation
  - E.g., bypass authentication

- Exploit preparation
  - Heap manipulation
  - Address space layout randomization (ASLR) probing
void do_authentication(...) {
    int authenticated = 0;
    while (!authenticated) {
        [...buffer overflow vulnerability...]
        if (auth_password(...)) {
            memset(...);
            xfree(...);
            log_msg(...);
            authenticated = 1;
            break;
        }
        memset(...);
        xfree(...);
        debug(...);
        break;
    }
    if (authenticated) {
        ...
    }
}

From [Chen ’05]
... sys_ioctl() sys_open() sys_read() sys_setpgid() sys_setsid() sys_fork() ...

n-gram [Forrest 1996]

[Forrest 2008]

[Chandola 2009]


[Forrest 2008]

[Chandola 2009]

FSA [Sekar 2001, Wagner 2001]

PDA [Feng 2003, Feng 2004, Giffin 2004]

[Chandola 2009]

Static Program Analysis

Dynamic Program Analysis


Data-flow analysis [Giffin 2006, Bhatkar 2006]

[Shu, Yao, Ryder. RAID 2015]
Old and New Challenges of Data-driven Anomaly Detection

Scale of Data
- Cloud support
- HPC
- Transparency

Subtlety
- Stealthy attacks, e.g., ROP, DOP

Definition of Anomalies
- Domain knowledge
- Inter-discipline
- Usability

Experimental Reproducibility
- Security guarantees
- Benchmarks, baselines, open source

Interpretation of Anomalies
- Semantic gap
- Meanings of anomalies
- Usability

Accuracy of Detection
Use 3 Host Protection Solutions as Examples

1: HMM-based local anomaly detection
2: Global trace analysis for frequency anomalies
3: Triggering relation discovery of system and network events
Issue 1: Incomplete Traces

<table>
<thead>
<tr>
<th>Program</th>
<th># of test cases</th>
<th>branch coverage</th>
<th>line cov.</th>
</tr>
</thead>
<tbody>
<tr>
<td>flex</td>
<td>525</td>
<td>81.34%</td>
<td>76.04%</td>
</tr>
<tr>
<td>grep</td>
<td>809</td>
<td>58.68%</td>
<td>63.34%</td>
</tr>
<tr>
<td>gzip</td>
<td>214</td>
<td>68.49%</td>
<td>66.85%</td>
</tr>
<tr>
<td>sed</td>
<td>370</td>
<td>72.31%</td>
<td>65.63%</td>
</tr>
<tr>
<td>bash</td>
<td>1061</td>
<td>66.26%</td>
<td>59.39%</td>
</tr>
<tr>
<td>vim</td>
<td>976</td>
<td>54.99%</td>
<td>51.93%</td>
</tr>
</tbody>
</table>

From SIR
How to do make HMM smarter in anomaly detection?

Better HMM initialization based on programs

Program analysis for HMM
• Xu, Yao, Ryder, Tian. *IEEE CSF ’15*
HMM with context
• Xu, Tian, Yao, Ryder. *IEEE DSN ’16*
Hidden Markov Model (HMM)

Markov process (memoryless) where some states are not observable
OBSERVABLE STATES

HIDDEN STATES

OBSERVABLE STATES

[Adapted from Udacity]
HMM-based Program Anomaly Detection
Probabilistic, Path sensitive, Local analysis, Semi-supervised training

Can we do better than random initialization?
STILO: STatically InitiaLized markOv

Transition probability of a call pair is its likelihood of occurrence during the execution of the function.

Example of call pair | Transition probability
---|---
read → write | 1-p
read → read | 0
execve → $\varepsilon_f'$ | pq

Function: $f$

<table>
<thead>
<tr>
<th>$\varepsilon_f'(exit)$</th>
<th>read</th>
<th>write</th>
<th>execve</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon_f$(entry)</td>
<td>p(1-q)</td>
<td>1-p</td>
<td>0</td>
</tr>
<tr>
<td>read</td>
<td>0</td>
<td>0</td>
<td>1-p</td>
</tr>
<tr>
<td>write</td>
<td>1-p</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>execve</td>
<td>pq</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$p, q$ are statically estimated.
Host Security Solution 1: Local Anomaly Detection with STILO

Static Program Analysis based HMM Initialization (New Contributions)
Why need context sensitive detection?
Improvement with Context Sensitivity

BEFORE: Context insensitive (STILO-basic)

AFTER: 1-level calling context sensitive (STILO-context)

Scalability: K-mean clustering reduces the # of hidden states

[Xu, Tian, Yao, Ryder. *IEEE DSN ’16*]
Reduction of Hidden States for Efficiency

Before clustering

One-to-one mapping -- a hidden state represents a single call

After clustering

Many-to-one mapping -- a hidden state may represent multiple similar calls

<table>
<thead>
<tr>
<th>Program Model</th>
<th># distinct calls</th>
<th># states after clustering</th>
<th>Estimated training time reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>bash</td>
<td>1366</td>
<td>455</td>
<td>88.91%</td>
</tr>
<tr>
<td>vim</td>
<td>829</td>
<td>415</td>
<td>74.94%</td>
</tr>
<tr>
<td>proftpd</td>
<td>1115</td>
<td>372</td>
<td>88.87%</td>
</tr>
</tbody>
</table>

- K-mean clustering, based on similarity between call-transition vectors
- Aim at 1/2 to 1/3 reduction of nodes
# STILO Evaluation

<table>
<thead>
<tr>
<th>Model</th>
<th>With Static Analysis</th>
<th>With Caller Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular-basic</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Regular-context</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>STILO-basic</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>STILO-context</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

2 Linux server programs: nginx, proftpd
6 Linux utility programs: flex, grep, gzip, sed, bash, vim

1. **Normal:** total 130,940,213 segments
2. **Abnormal-S:** 160,000 Abnormal-S segments (permute 1/3 calls)
3. **Abnormal-A:** attack call sequences obtained from exploits

Dyninst for static program analysis, Jahmm library for HMM, 1st-order Markov, strace/ltrace for collection, SIR for test cases, 10-fold cross validation, 15-grams from traces
For libcalls, false negative (missed detection) of context-sensitive models drops by 2-3 orders.
For syscalls, context improves false negative rate by 10 folds. Less dramatic improvement than libcalls.
Increasing hidden states in regular HMM does not guarantee classification accuracy.

- syscall:grep
  - 2x
  - 2.5x
  - 3x
  - 3.5x
  - Our-2.92x

- syscall:gzip
  - 1.5x
  - 2x
  - 2.5x
  - 3x
  - 3.5x
  - Our-2.35x
### Detection of Real-world Attacks

ROP attack segments against gzip (syscall)

<table>
<thead>
<tr>
<th>ID</th>
<th>Prob in STILO</th>
<th>Prob in Regular HMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>S₂</td>
<td>2.20 × 10^{-15}</td>
<td>0.29</td>
</tr>
<tr>
<td>S₃</td>
<td>1.54 × 10^{-5}</td>
<td>0.25</td>
</tr>
<tr>
<td>S₄</td>
<td>0</td>
<td>0.27</td>
</tr>
<tr>
<td>S₅</td>
<td>0.0005</td>
<td>0.33</td>
</tr>
<tr>
<td>S₆</td>
<td>0</td>
<td>0.23</td>
</tr>
<tr>
<td>S₇</td>
<td>0.0004</td>
<td>0.26</td>
</tr>
</tbody>
</table>

STILO gives much lower probabilities for attack sequences

<table>
<thead>
<tr>
<th>Exploit</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer Overflow (gzip)</td>
<td>ROP</td>
</tr>
<tr>
<td></td>
<td>ROP_syscall_chain</td>
</tr>
<tr>
<td>Backdoor (proftpd)</td>
<td>bind_perl</td>
</tr>
<tr>
<td></td>
<td>bind perl ipv6</td>
</tr>
<tr>
<td></td>
<td>generic cmd execution</td>
</tr>
<tr>
<td></td>
<td>double reverse TCP</td>
</tr>
<tr>
<td></td>
<td>reverse_perl</td>
</tr>
<tr>
<td></td>
<td>reverse_perl_ssl</td>
</tr>
<tr>
<td></td>
<td>reverse_ssl_double_telnet</td>
</tr>
<tr>
<td>Buffer Overflow (proftpd)</td>
<td>guess memory address</td>
</tr>
</tbody>
</table>
Ongoing Work: Hardware-assisted Program Tracing for Anomaly Detection

In collaboration with Trent Jaeger (PSU)
Performance and Ease of Deployment

Training Traces (host) → Probability forecast → HMM init & training → Test traces (host) → HMM classification

Could be Fast → Fast and slow → Painfully slow → Extremely fast

Moderate → Not easy → Not easy to set up → Moderate
What does it take to outsource STILO detection to the cloud?

Training Traces (host) → Probability forecast → HMM init & training → Test traces (host) → HMM classification
Issue 2: Local Analysis

Local analysis is inadequate

Anomalies consisting of normal execution fragments
Attack Model, Problem Statement

Cooccurrence Anomaly

Normal 1: a b d a c e a
Normal 2: c b e a c c e c f
Normal 3: f d c e c f e d
Anomaly: a b d a c c f e d

Attack examples:
• Non-control data attack
• Fragment-based mimicry attack
• Workflow violation attack

Problem Statement:
• Given an extremely long trace, should any set of events co-occur?
• With the expected frequency?

Frequency Anomaly

Attack examples:
• DoS attacks
• Directory harvest attacks

Can n-gram still work?
Host Security Solution 2: Global Anomaly Detection

An infinite long call trace:
... bar, main, foo, bar, bar, ...

\[ \text{chop} \rightarrow \text{into} \]

Long trace segments

\[ \text{convert} \rightarrow \text{into} \]

Behavior instance

1. Transition frequency matrix

<table>
<thead>
<tr>
<th></th>
<th>main</th>
<th>foo</th>
<th>bar</th>
<th>goo</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>foo</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>bar</td>
<td>2</td>
<td>6</td>
<td>89</td>
<td>1</td>
</tr>
<tr>
<td>goo</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Matrix representation is path insensitive

2. Event co-occurrence matrix

\[ \begin{array}{cccc}
F & T & F & F \\
F & F & T & F \\
T & T & F & T \\
F & F & F & F \\
\end{array} \]

[Shu, Yao, Ramakrishnan. ACM CCS ’15]
Our Solution: Grouping Similar Normal Behaviors

Training Phase

Detection Phase

A trace segment represented by matrices
Montage Anomalies Fall Between Clusters

Pass Auth. (expected)

... do_auth > xfree
do_auth > log_msg
do_auth > packet_start
... pwrite > buffer_len
do_auth > do_auth
...

Anomalous: attack

... do_auth > debug
do_auth > xfree
do_auth > packet_start
... pwrite > buffer_len
do_auth > do_auth
...

Fail Auth. (expected)

... do_auth > debug
do_auth > xfree
do_auth > packet_start
... pwrite > buffer_len
do_auth > pread
...

Function call trace
(collected through Pintool)
Our Operations

- Inter-cluster training
- Intra-cluster training
- Inter-cluster detection on co-occurrence matrices
- Intra-cluster detection on frequency matrices

[Diagrams showing clusters and anomalies]
Under 10-fold cross-validation with 10,000 normal test cases, 1,000 synthetic anomalies.
Exp 2: Detection of Real-world Attacks in Complex Programs

- **sshd**: Training w/ 4,800 normal behavior instances (34K events each)
- **libpcre**: Training w/ 11,027 normal behavior instances (44K events each)
- **sendmail**: Training w/ 6,579 normal behavior instances (1K events each)

- **Flag variable overwritten attacks w/ various lengths**
- **Regular Exp. DoS**: 3 malicious patterns 8-23 strings to match
- **Directory harvest attack w/ probing batch sizes**: 8 to 400 emails

100% Detection accuracy
0.01% Average false alarm rate
How to lift this host security solution to the cloud?

Privacy
- Trust the provider or not?
- What is leaked, if detection is outsourced to the cloud?
- Is it possible to relax the privacy model?

Transparency
- Does the client need to be involved?
- Client gives feedback on detection results, like spam detection?

Correctness
- How can client trust provider do a decent job?
Host Security Solution 3: Triggering Relation Discovery

Triggering Relation Graph (TRG)

NSF CAREER Award.

How to lift this analysis to the cloud?

Prototypes for
• Android traffic, Linux traffic
• Filesys events

[Zhang AISec ‘16] [Zhang C&S 2016]
[Zhang AS/ACCS ‘14] [Xu IEEE TDSC ’12]
Future Work: Anomaly Detection as a Cloud Service

Is it possible to
• be transparent to clients?
• for interdisciplinary data?
• with domain knowledge?
• in production systems?

Can domain experts understand these suggestions?

• Some algorithms are not good for global anomalies;
• The safe bet is to try first global detection algorithms;
• If willing to wait (not real-time detection), use nearest neighbor;
• If the dataset is small, definitely avoid clustering;
• Restart k-mean multiple times to obtain stable clusters;
• Avoid unsupervised anomaly detection for extremely high dimensions;

[Goldstein and Uchida 2016]
Privacy, is it a lost battle (at least in US)?

- US Internet service providers (ISP) to monitor customers’ behavior online without users’ permission,
- to use personal information to sell highly targeted ads

This includes:
- Internet history
- Mobile location data
- App usage
- Content of emails/messages
- Financial information
- Health data

[Washington Post, March 28, 2017]
Lifting data-driven host protection to the cloud

Thank you for your attention!

Questions?

More information:

- http://people.cs.vt.edu/danfeng/
- CCS program anomaly detection tutorial video and slides
- System traces, hands-on exercises