Project in Computer Security
Project in Computer Security
Algorithm

Feature selection

- correlated features
  The simplest feature vector is
  [country, os, browser, event]

- uncorrelated features
  We selected the following features:
  [country, OS x browser, OS x event]

For example:

browser: [Mobile Safari, Chrome Mobile, ...]

Increasing the number of features typically increases classification accuracy to a point, but when it continues to increase, classification accuracy will eventually decrease because we are then undersampled relative to the large number of features.

Results - 1

We worked with approx. 10,000 user's activities in office365.

We trained the algorithm with 70% of the data and tested the classifier on the rest 30%.

The 4 most anomalous events detected for a random user were:

[country_1, windows 10, chrome, LOGIN, EVENT]
[country_1, windows 10, chrome, FOLDER, VIEW]
[country_1, windows 10, chrome, CONVERSATION, VIEW]
[country_1, windows 10, chrome, MESSAGE_DELETE]

Results - 2

- Security personnel gets a sorted list of top anomalous events to investigate.
- Feedback is necessary to exclude certain combinations of features (e.g. logout events, iOS usage by certain users, etc.).

Overview

Security overview in SaaS

Advantages:
- The server is administered by professionals.
- Web-based services are always patched.
- Easy to use from any place in the world.

Disadvantages (Main problem):
- If client’s computer is infected (or the source of client’s log is from infected internet cafe) then the malware can steal cookies or inject itself to a user’s session and put a hand on sensitive information.

How to solve the problem

Malware behaves very differently from a normal avg. user.

By Auditing All events on the SaaS we can learn user’s behavior and detect: suspicious Malware activity -
- Users that stole the credentials -
- Users which are trying to do malicious activity on purpose.

Machine learning to the rescue

- Anomaly detection with discrete probability distribution
- Markov-Chain
- Markov-Chain with memory

Technology

We implemented ML algorithms in python 2.7 on Ubuntu 12.04 V7.

Libraries were used: NumPy v1.7.1, SciPy v0.12, Matplotlib v1.2.1.
Big Data at the Service of SaaS Security Project

Presenting:
Evgeniy Pats - sjeka@t2.technion.ac.il
Michael Pekel - mipmip@t2.technion.ac.il

Under the guidance of Mr. Yonatan Most

Professor in Charge: Eli Biham

October 2014
Security overview in SaaS

Advantages:
- The server is administrated by professionals
- Web-based services are always patched
- Easy to use from any place in the world

Disadvantages (Main problem):
- If client's computer is infected (or the source of client's log is from infected internet cafe) then the malware can steal cookies or inject itself to a user's session and put a hand on sensitive information.
1. Break-in: Phishing and remote exploits to gain access
2. Latch-on: Malware and back doors installed to establish a foothold
3. Expand: Reconnaissance and lateral movement to increase access and maintain a presence
4. Gather: Acquisition and aggregation of confidential data
5. Exfiltrate: Data exfiltration to external networks
How to solve the problem

Malware behaves very differently from a normal avg. user.

By Auditing All events on the SaaS we can learn user's behavior and detect: suspicious Malware activity -
  • Other user that stole the credentials
  • Users which are trying to do a malicious activity on purpose.
System Design

- The analysis module can range from new policies enforced by IT to behavioural analysis and machine learning algorithms.
Machine learning to the rescue

- Anomaly detection with discrete probability distribution
- Markov-Chain
- Markov-Chain with memory
Technology

We implemented ML algorithms in python2.7 on Ubuntu 12.03 v11

Libraries were used: NumPy v1.7.1, SciPy v0.12, Matplotlib v1.2.1
Data

The data we are working with is sanitized by Adallom (in Json format) for different applications.

user activity example:

```json
{
    "username": 5,
    "failed_username": null,
    "timestamp": 1394230859937,
    "country": 2,
    "IP": 4,
    "app": 11114,
    "OS": "Windows",
    "event": "EVENT_SF_REPORTS_VIEW",
    "browser": "Chrome"
}
```
Anomaly detection - Discrete Distribution

**Motivation**
- Attack typically consists by sequence of events, when each event may considered as legal, but sequence of events are illegal.
- Each session described by a sequence of finite events

We wanna test it by our algorithm

**Algorithm**
- Correlated features
  - The simplest feature vector is
    - [country, OS, browser, event]
- Uncorrelated features
  - We selected the following features:
    - [country, OS x browser, OS x event]

For example:
- Browser: [Mobile Safari, Chrome Mobile,...]

Increasing the number of features typically increases classification accuracy to a point, but when it continue to increase, classification accuracy will eventually decrease because we are then undersampled relative to the large number of features

**Results**
- The dataset same as before.
- Here we can see the following 2 anomalous events:
  - [SEND, LOGIN] - repeated 4 times
  - [LOGIN, LOGOUT] - repeated 1 time

We were executed the algorithm on a specific user, OS and browser.

**Results - 1**
- We worked with approx. 11,000 user's activities in office365
- We trained the algorithm with 70% of the data and tested the classifier on the rest 30%
- The 4 most anomalous events detected for a random user were:
  - [country_2, Windows_8, Chrome, LOGIN, EVENT]
  - [country_2, Windows_8, Chrome, FOLDER, VIEW]
  - [country_2, Windows_8, Chrome, MESSAGE, DRAFT]
  - [country_1, Windows_8, Chrome, CONVERSATION, VIEW]
  - [country_2, Windows_1, Chrome, MESSAGE, DRAFT]

- Security personnel gets a sorted list of top anomalous events to investigate.
- Feedback is necessary to exclude certain combinations of features (login event, iOS usage by certain users, etc.)
Algorithm

Given a data set: \( x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)} \)

Where each example is a vector \( \forall x_i \in \mathbb{R}^n \).

We modeled our feature vectors to be presented by enuns.

The probability of occurance for each event is measured by:

\[
\begin{pmatrix}
  x_1^1 & x_2^1 & x_3^1 & x_4^1 & x_5^1 \\
  x_1^2 & x_2^2 & x_3^2 & x_4^2 & x_5^2 \\
  x_1^3 & x_2^3 & x_3^3 & x_4^3 & x_5^3 \\
  x_1^4 & x_2^4 & x_3^4 & x_4^4 & x_5^4 \\
  x_1^5 & x_2^5 & x_3^5 & x_4^5 & x_5^5
\end{pmatrix} = (p(x_1) \ p(x_2) \ p(x_3) \ p(x_4) \ p(x_5)) = \prod_{j=1}^{n} p(x_j)
\]

(1)

\[ p(x) = \prod_{j=1}^{n} p(x_j), \text{ whereas } p(x_j) = \frac{1}{m} \cdot \{\forall x_i | \# x^{(i)}\}_{i=1}^{m} \]

: \( m \) – total samples; \( n \) – number of features

Let\( s\)say we choose a threshold to be \( \epsilon \).

Then anomaly will be detected when: \( p(x) < \epsilon \)
\[
\begin{bmatrix}
  x_3^3 & x_4^3 & x_5^3 \\
  x_3^4 & x_4^4 & x_5^4 \\
  x_3^5 & x_4^5 & x_5^5 \\
\end{bmatrix}
= (p(x_1) \ p(x_2) \ p(x_3) \ p(x_4) \ p(x_5))
\]

\[
p(x) = \prod_{j=1}^{n} p(x_j) , \text{ whereas } p(x_j) = \frac{1}{m} \cdot \left\{ \forall x_i \mid \# x^{(i)} \right\}_{i=1}^{m}
\]

: \( m \) – total samples ; \( n \) – number of features

Lets say we choose a threshold to be \( \epsilon \),

Then anomaly will be detected when: \( p(x) < \epsilon \)
Feature selection

- correlated features
  The simplest feature vector is 
  [country, os, browser, event]

- uncorrelated features
  We selected the following features: 
  [country, OS x browser, OS x event]

For example:
browser : {Mobile Safari, Chrome Mobile,...}

Increasing the number of features typically increases 
classification accuracy to a point, but when it continue 
to increase, classification accuracy will eventually 
decrease because we are then undersampled relative to 
the large number of features
Results - 1

We worked with approx. 11,000 user's activities in office365

We trained the algorithm with 70% of the data and tested the classifier on the rest 30%

The 4 most anomalous events detected for a random user were:

[country_2, windows_8, chrome, LOGIN_EVENT]
[country_2, windows_8, chrome, FOLDER_VIEW]
[country_2, windows_8, chrome, CONVERSATION_VIEW]
[country_1, windows, chrome, MESSAGE_DELETE]
Results - 2

The 4 most anomalous for another random user:

[country_1, Android 4.0.x, chrome, LOGIN]
[country_1, Windows 7, MESSAGE_VIEW]
[country_1, windows 7, LOGOUT]
[country_1, windows 7, MESSAGE_DRAFT]

• Security personnel gets a sorted list of top anomalous events to investigate
• Feedback is necessary to exclude certain combinations of features (logout events, iOS usage by certain users, etc.)
Markov Chain

**Motivation**
- Attack typically consists by sequence of events, when each event may considered as legal, but sequence of events are illegal.
- Each session described by a sequence of finite events

We wanna test it by our algorithm

**Algorithm**

1. Markov chain is a sequence of random variables with the Markov property, meaning that the future state depends only on the current state and not on the past states.

   \[ P(X_n | X_1, X_2, ..., X_{n-1}) = P(X_n | X_{n-1}) \]

   The transition matrix of the Markov chain is calculated and stored in the Markov state for future events. If the state is unchanged, the event is C.

We were executed the algorithm on a specific user, OS and browser.

**Results**

The dataset same as before.

Here we can see the following 2 anomalous events:

- [SEND, LOGIN] - repeated 5 times
- [LOGIN, LOGIN] - repeated 1 time

\[(\text{EVENT}_03665\_OUTLOOK\_SEND', 'timestamp': '2014-05-01 22:15:53.067000'},\]
\[\{\text{EVENT}_03665\_USER\_LOGIN', 'timestamp': '2014-05-01 22:15:53.010000'}\]

2 anomalous events we discovered but can be ignored were:

\[u'EVEN'_0365\_OUTLOOK\_SEND', 'timestamp': '2013-02-22 02:24:05.887000'}\]
\[u'EVEN'_0365\_USER\_LOGIN', 'timestamp': '2014-02-24 00:25:28.585000'}\]

To eliminate these false positives we can define our session to be more precise, two consecutive events are in same session only if the interval is less than 30 minutes (depends on the app).
Motivation

- Attack typically consists by sequence of events, when each event may considered as legal, but sequence of events are illegal.
- Each session described by a sequence of finite events

We wanna test it by our algorithm
Algorithm

A Markov chain is a sequence of random variables with the Markov property, namely that, given the present state, the future and past states are independent.
Formally,

\[ Pr(X_{n+1} = x \mid X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = Pr(X_{n+1} = x \mid X_n = x_n) \]

The possible values of \( X_i \) from a countable set \( S \) called the state space of the chain.
Markov chain often described by a transition matrix \( M \) from state \( n \) to \( n+1 \), i.e

\[ M[i][j] = Pr(X_{n+1} = i \mid X_n = j) \]

We were executed the algorithm on a specific user, OS and browser.
A Markov chain is a sequence of random variables with the Markov property, namely that, given the present state, the future and past states are independent.

Formally,

\[ Pr(X_{n+1} = x \mid X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = Pr(X_{n+1} = x \mid X_n = x_n) \]

The possible values of \( X_i \) from a countable set \( S \) called the state space of the chain. Markov chain often described by a transition matrix \( M \) from state \( n \) to \( n + 1 \), i.e

\[ M[i][j] = Pr(X_{n+1} = i \mid X_n = j) \]
Algorithm

A Markov chain is a sequence of random variables with the Markov property, namely that, given the present state, the future and past states are independent.

Formally,

\[ Pr(X_{n+1} = x | X_1 = x_1, X_2 = x_2, ..., X_n = x_n) = Pr(X_{n+1} = x | X_n = x_n) \]

The possible values of \( X_i \) from a countable set \( S \) called the state space of the chain.

Markov chain often described by a transition matrix \( M \) from state \( n \) to \( n+1 \), i.e.

\[ M[i][j] = Pr(X_{n+1} = i | X_n = j) \]

We were executed the algorithm on a specific user, OS and browser.
**Results**

The dataset same as before.

Here we can see the following 2 anomalous events:

[SEND, LOGIN] - repeated 4 times
[LOGIN, LOGIN] - repeated 1 time

[[{'EVENT_O365_OUTLOOK_SEND',
  'timestamp': u'2014-05-01 22:15:54.067000'},
  {'EVENT_O365_USER_LOGIN',
   'timestamp': u'2014-05-01 22:15:55.010000'}]]
2 anomalous events we discovered but can be ignored were:

u’EVENT_O365_OUTLOOK_SEND’, 
’timestamp’: u’2014-02-22 02:24:05.897000’], 
[ u’EVENT_O365_USER_LOGIN’, 
’timestamp’: u’2014-02-24 00:25:28.385000’]

- To eliminate these false positives we can define our session to be more precise, two consecutive events are in the same session only if the interval is less than 30 minutes (depends on the app)
Markov Chain - variant

**Problems and Improvements**

- Predict always between 2 events the occurrence and probability of getting them in the same order is not enough
- we calculated conditional probability and collaborated each time set of 10 events (in a row)

**Results and Analysis**

The 2 most mentioned versions of a specific user

![Graph showing results and analysis](image-url)
Problems and Improvements

- Predict always between 2 events the occurrence and probability of getting them in the same order is not enough.
- We calculated conditional probability and collaborated each time set of 10 events (in a row).

The probability that a sequence of states $X_1, X_2, ..., X_T$
When we chose $T = 10$ means we inspect each time a sequence window of 10 events in a row when they occur in the context of the stationary Markov - Chain and can computed as follows:

$$P(X_1, X_2, ..., X_{10}) = q_{x_1} \cdot \prod_{t=2}^{10} P_{X_{t-1},X_t}$$
them in the same order is not enough

- we calculated conditional probability and collaborated each time set of 10 events (in a row)

*The probability that a sequence of states* \( X_1, X_2, \ldots, X_T \)

*When we chose* \( T = 10 \) *means we inspect each time a sequence window of 10 events in a row when they occurs in the context of the stationary Markov - Chain and can computed as follows:

\[
P(X_1, X_2, \ldots, X_{10}) = q_{x_1} \cdot \prod_{t=2}^{10} P_{X_{t-1} \cdot X_t}
\]
Results and Analysis

The 2 most anomalous sessions of a random user

- [ATTACHMENT_DOWNLOAD, ATTACHMENT_DOWNLOAD', SEND, CONVERSATION_VIEW, DOWNLOAD, ATTACHMENT_DOWNLOAD, CONVERSATION_VIEW, ATTACHMENT_DOWNLOAD, LOGOUT, SITE_VIEW]
- [CONVERSATION_VIEW, FOLDER_DELETE, FOLDER_VIEW, FOLDER_DELETE, FOLDER_VIEW, FOLDER_DELETE, FOLDER_VIEW, FOLDER_DELETE, FOLDER_DELETE]

Figure 4: Markov-Chain anomaly distribution between events
Figure 4: Markov-Chain anomaly distribution between events
Project in Computer Security
Thank You