Machine learning and Security of web applications

I. Introduction

- The consumer web has many different attack surfaces:
  - Account access, payment interfaces, and content generation,...
- *Social networks* provide another dimension of vulnerability, as attackers can take advantage of the social graph to achieve their goals.
- Types of Abuse:
  - Account takeover, account creation, financial fraud, and bot activity

Authentication and Account Takeover

- **Differentiating the experience for different users**
  - You need to be able to determine which user is making each request
  - This requires some form of *user authentication*.
- **Authentication** on the internet today overwhelmingly takes the form of *passwords*
  - In practice, however, passwords suffer from many flaws

Authentication and Account Takeover

- **Biometric identifiers** such as fingerprint or iris scans have been shown to be uniquely identifying.
- **Multi-factor authentication** has been introduced
- A website that wants to protect its users from ***account takeover*** must implement some mechanism to distinguish legitimate logins from illegitimate ones.
  - This is where machine learning and statistical analysis come in.

Authentication and Account Takeover: Features used to classify login attempts

- If we want to keep attackers out of accounts that they do not own:
  - whenever we see a correct username–password pair we must run some *additional logic* that decides whether to require further verification before letting the request through
- This logic must run in a blocking manner and use only data that can be collected from the login form
  - What are the main classes of data (signals) to be collected???

Authentication and Account Takeover: Features used to classify login attempts

- **Signals indicating a brute-force attack on a single account:**
  - Velocity of login attempts on the account—simple rate limiting can often be effective
  - Popularity rank of attempted passwords—attackers will try common passwords
  - Distribution of password attempts on the account—real users will attempt the same password or similar passwords

Authentication and Account Takeover: Features used to classify login attempts

- **Signals indicating a deviation from this user's established patterns of login:**
  - Geographic displacement from previous logins (larger distances being more suspicious)
  - Using a browser, app, or OS not previously used by the account
  - Appearing at an unusual time of day or unusual day of the week
  - Unusual frequency of logins or inter-login time
  - Unusual sequence of requests prior to login

Authentication and Account Takeover: Features used to classify login attempts

- **Signals indicating large-scale automation of login:**
  - High volume of requests per IP address, user agent, or any other key extracted from request data
  - Unusually high number of invalid credentials across the site
  - Requests from a hosting provider or other suspicious IP addresses
  - Browser-based telemetry indicating nonhuman activity—e.g., keystroke timing or device-class fingerprinting

Authentication and Account Takeover: Features used to classify login attempts

- Engineering all of these features is a nontrivial task
  - For example, to capture deviations from established patterns, you will need a data store that captures these patterns (e.g., all IP addresses an account has used).
- Assuming you have the ability to collect some or all of these features.
  - The next step depends on the labeled data you have available.
- Labeling account compromise is an imprecise science

Authentication and Account Takeover: Features used to classify login attempts

- **Assuming you don't have any labeled data.** How do you use the signals described here to detect suspicious logins?
- T1: The first technique you might try is to set thresholds on each of the aforementioned features
  - **For example**: you could require extra verification if a user logs in more than *10 times/one hour*, or more than *500 miles* from any location at which they were previously observed, or using an operating system they haven't previously used

Authentication and Account Takeover: Features used to classify login attempts

- T2: a more principled, statistical approach
  - the probability that the person logging in is the account owner

$$\Pr\left(\text{IP} = x \mid \text{User} = u\right) = \frac{\#\text{ logins with IP} = x \text{ and User} = u}{\#\text{ logins with User} = u}$$

Authentication and Account Takeover: Features used to classify login attempts

- We can use similar techniques for all of the features labeled earlier as "signals indicating a deviation from established patterns of login":
  - In each case, we can compute the proportion of logins from this user that match the specified attribute.
- How about the "large-scale automation" features?
  - To calculate velocity features, we need to keep *rolling counts* of requests over time that we can look up on each incoming request

Authentication and Account Takeover: Building your classifier

- Now that you have all these features, how do you combine them?
  - Without labels, this task is a difficult one.
- Each feature you have computed represents a probability,
  - the naive approach is to assume that all of the features are independent and simply multiply them together to get a total score (equivalently, you can take *logs* and add).
- A more sophisticated approach would be to use one of the anomaly detection techniques,
  - such as a one-class support vector machine, isolation forest, or local outlier factor.

Authentication and Account Takeover: Building your classifier

- With labels, you can train a supervised classifier.
- Typically, your labels will be highly unbalanced (there will be very little labeled attack data),
  - so you will want to use a technique such as under-sampling the benign class or adjusting the cost function for the attack class.

Account Creation

- **If attackers can create a large number of accounts, your site could potentially be overwhelmed by illegitimate users.**
  - Thus, protecting the account creation process is essential to creating a secure system.
- **Two general approaches to detecting and weeding out fake accounts:**
  - Scoring *the account creation request* to prevent fake accounts from being created at all,
  - Scoring *existing accounts* in order to delete, lock, or otherwise restrict the fake ones

Account Creation

- **Example**:
  - Suppose that you have concluded from your data that if more than 20 new accounts are created from the same IP address in the same hour, these accounts are certain to be fake.
  - If you score at account creation time, you can count creation attempts per IP address in the past hour and block (or perhaps show a CAPTCHA or other challenge) whenever this counter is greater than 20
- **Suppose considering a scoring model that runs at account creation time**
  - The features that go into such a model fall into two categories: *velocity features* and *reputation scores*.

Account Creation: Velocity features

- The basic building block of a velocity feature is a *rolling counter.*
- The counter records key information, such as
  - IP address or IP subnet
  - Geolocation features (city, country, etc.)
  - Browser type or version
  - Operating system and version
  - Success/failure of account creation
  - Features of the account username (e.g., substrings of email address or phone number)
  - Landing page in the account creation flow (e.g., mobile versus desktop)
  - API endpoint used
  - Any other property of the registration request.

Account Creation: Velocity features

- Now that you've implemented counters, you can compute velocity features.
- Simple velocities are straightforward:
  - just retrieve the counts from a certain number of buckets (*time intervals*), sum them up, and divide by the number of buckets
- Velocities can be very effective when combined into *ratios*.

Account Creation: Velocity features

- Spike detection on a single key uses the ratio of recent requests to older requests.
- But any of the following ratios should have a “typical” value:
  - Ratio of success to failure on login or registration (failure spikes are suspicious)
  - Ratio of API requests to page requests (the API being hit without a corresponding page request suggests automation, at least when the user agent claims to be a browser)
  - Ratio of mobile requests to desktop requests
  - Ratio of errors (i.e., 4xx response codes) to successful requests (200 response code)
- You can use rolling counters to compute these ratios in real time and therefore incorporate them as features in your classifier.

Account Creation: Reputation scores

- Velocity features are good at catching high-volume attacks.
- What if the attacker slows down and/or diversifies their keys to the point where the requests are coming in below the threshold for any given key?
- How can we make a decision on a single request?
- The foundation here is the concept of *reputation*

Account Creation: Reputation scores

- Reputation quantifies the knowledge we have about a given key (e.g., IP address)
  - That knowledge is either from our own data or from third parties, and
- Reputation gives us a starting point (or more formally, a prior) for assessing whether a request from this key is legitimate

Account Creation: Reputation scores

- The simplest reputation signal for a registration defense model is *the fraction of accounts* on that IP that have been previously *labeled as bad*.
- A more sophisticated system might capture other measures of “badness”:
  - how many account takeover attempts have been seen from the IP, or how much spam has been sent from it.

Account Creation: Reputation scores

- **If these labels are not readily available, you can use proxies for reputation:**
  - How long ago was the IP first seen? (Older is better.)
  - How many legitimate accounts are on the IP?
  - How much revenue comes in from the IP?
  - How consistent is traffic on the IP? (Spikes are suspicious.)
  - What's the ratio of reads to writes on the IP? (More writes = more spam.)
  - How popular is the IP (as a fraction of all requests)?
- There is a great deal of external data that can go into your reputation system, especially with regard to IP addresses and user agents

Account Creation: Reputation scores

- You can subscribe to a database that will inform you as to *whether any given IP address belongs to a hosting provider, is a VPN or anonymous proxy, is a Tor exit node, and so on*.
- There are a number of vendors who monitor large swaths of internet traffic and compile and sell IP blacklist feeds.

Account Creation: Reputation scores

- With regard to user agents, there are published lists of known *bot* user agents and web scripting packages.
  - For example, requests from *curl, wget*, or *python-requests* should be treated with extra suspicion.
  - Similarly, if a request advertises itself as *Googlebot*, it is either the real Googlebot or someone pretending to be Googlebot.

Account Creation: Reputation scores

- Now let's consider how to combine all of these features into a reputation score, again using IP address as an example.
- Our supervised learning problem is now as follows:
  - Given the past *m* days of data for an IP, predict whether there will be a fake account created on it in the next *n* days.

Account Creation: Reputation scores

- To train our supervised classifier:
  - We compute features for each IP for a period of *m* days and labels for the *n* days immediately following.
- Certain features are
  - Properties of the IP, such as “is hosting provider.”
  - Others are time based, such as “number of legitimate accounts on the IP.”
  - We could compute the number of accounts over the entire *n* days, or for each of the *n* days, or something in between.
  - If there is enough data, we could compute *n* features, one for each day, in order to capture trends.

Financial Fraud

- If your audience is large enough, there will be some people who try to steal your product regardless of what it is.
  - To stop these people, you need to determine whether each purchase on your site is legitimate.
- The vast majority of financial fraud is conducted using stolen credit cards.
- In any case, after you have made this decision, you will need to collect *features* that are indicative of fraud.

a)

- **Spending profiles of customers:**
  - How many standard deviations from the customer's average purchase a given transaction is
  - Velocity of credit card purchases
  - Prevalence of the current product or product category in the customer's history
  - Whether this is a first purchase (e.g., a free user suddenly begins making lots of purchases)
  - Whether this payment method/card type is typical for the customer
  - How recently this payment method was added to the account

b)

- **Geographical/time-dependency correlation:**
  - All the correlation signals for authentication (e.g., geographic displacement, IP/browser history)
  - Geographic velocity (e.g., if a physical credit card transaction was made in London at 8:45 PM and in New York at 9:00 PM on the same day, the user's velocity is abnormally high)
- **Data mismatch:**
  - Whether the credit card billing address matches the user's profile information at the city/state/country level
  - Mismatch between billing and shipping addresses
  - Whether the credit card's bank is in the same country as the user

c)

- **Account profile:**
  - Age of the user's account
  - Reputation from account creation scoring (as just discussed)
- **Customer interaction statistics:**
  - Number of times the customer goes through the payment flow
  - Number of credit cards tried
  - How many times a given card is tried
  - Number of orders per billing or shipping address

Financial Fraud

- The “gold standard” for labeled data is chargebacks—purchases declared by the card owner to be fraudulent and rescinded by the bank.
  - However, it usually takes at least one month for a chargeback to complete, and it can even take up to six months in many cases.
- As a result, chargebacks cannot be used for short-term metrics or to understand how an adversary is adapting to recent changes.
- Thus, if you want to measure and iterate quickly on your fraud models, you will need a supplementary metric. i.e:
  - Customer reports of fraud
  - Customer refunds
  - Purchases made by fake or compromised accounts

Bot Activity

- It follows that finding abuse in many cases is equivalent to finding automated activity (aka *bots*) on your site or app.
- *Bots* might try to take any of a number of actions, including the following:

Bot Activity

- Account creation: This was discussed in depth earlier.
- Credential stuffing: Running leaked lists of username/password pairs against your login infrastructure to try to compromise accounts.
- Scraping: Downloading your site’s data for arbitrage or other illicit use.

Bot Activity

- Click fraud: Inflating click counts to bring extra revenue to a site that hosts advertisements, or using artificial clicks to deplete a competitor's ad budget.
- Ranking fraud: Artificially increasing views, likes, or shares in order to make content reach a wider audience.
- Online gaming: Bots can simulate activity that would be tedious or expensive for humans to take.

Bot Activity

- Assuming that you have aggregated requests that you believe to be from a single entity. How do you determine whether the activity is automated?
- The key idea here is that the pattern of requests from bots will be different from the patterns demonstrated by humans.
- Specific quantitative signals include the following

Bot Activity

- Velocity of requests: Bots will make requests at a faster rate than humans.
- Regularity of requests, as measured by variance in time between requests: Bots will make requests at more regular intervals than humans.
- Entropy of paths/pages requested:
  - Bots will focus on their targets instead of browsing different parts of the site.
  - Scraping bots will request each page exactly once, whereas real users will revisit popular pages.
- Repetitive patterns in requests: A bot might repeatedly request page A, then B, then C, in order to automate a flow.

Bot Activity

- Unusual transitions: For example, a bot might post to a content generation endpoint without loading the page containing the submission form.
- Diversity of headers: Bots might rotate IP addresses, user agents, referrers, and other client-site headers in order to seem like humans,
  - but the distribution generated by the bot is unlikely to reflect the typical distribution across your site.
- Diversity of cookies: A typical website sets a session cookie and can set other cookies depending on the flow.
  - Bots might ignore some or all of these set-cookie requests, leading to unusual diversity of cookies.
- Distribution of response codes: A high number of errors, especially 403 or 404, could indicate bot requests whose scripts are based on an older version of the site.

Bot Activity

- Sometimes, you will not be able to aggregate requests on user ID or any other key
  - That is, you have no reliable way to determine whether any two requests come from the same actor.
- In these cases, you will need to use request-based bot detection;
  - That is, determine whether a request is automated using only data from the request itself, without any counters or time series data

Bot Activity

- Labeling bot requests in order to compute metrics or train supervised models is a difficult endeavor.
- Unlike spam, which can be given to a human to evaluate, there is no reasonable way to present an individual request to a reviewer and have that person label the request as bot or not.
- We thus must consider some alternatives: self advertise, automated writes/reads,…

II. Supervised Learning for Abuse Problems

- There are some subtleties that you will need to address in order to achieve good performance.
- We will use *the account creation classifier* as an example, but these considerations apply to any classifier you build when using adversarial data.

Labeling Data

- In the ideal case, you will have a large set of data that was sampled for this particular project and manually labeled.
  - If this is the case for your site or app, great!
  - In real life, however, hand-labeled data usually isn't presented to you on a platter.
  - Even if you can get some, it might not be enough to train a robust classifier.
- At the other extreme, suppose that you are unable to sample and label any data at all.
  - Is supervised learning useless?

Labeling Data

- There are some steps that you can take to mitigate risk when using imperfectly labeled data:
  - Oversample manually labeled examples in your training data.
  - Oversample false positives from this model when retraining (as in boosting).
  - Undersample positive examples from previous iterations of this model (or closely related models/rules).
  - If you have resources to sample and manually label some accounts, use them to adjudicate cases near the decision boundary of your model (as in active learning).

Cold Start Versus Warm Start

- If you have no scoring at account creation (“cold start”), model training is straightforward:
  - use whatever labels you have to build the model.
- If there is already a model running at account creation time and blocking some requests (“warm start”),
  - the only bad accounts that are created are the ones that get through the existing model (i.e., the false negatives).
  - If you train v2 of your model on only this data, v2 might “forget” the characteristics of v1

Cold Start Versus Warm Start

- There are a few different approaches to avoid this problem:
- **Never throw away training data:** Collect your data for the v2 model after v1 is deployed, and train on the union of v1 and v2 training data.
- **Run simultaneous models:** Train and deploy the v2 model, but also run the v1 model at the same time. Most likely the reason you were building v2 in the first place is that the performance of v1 is degrading.
- **Sample positives from the deployed v1 model to augment the v2 training data:** The advantage of this approach is that all the training data is recent; the disadvantage is that it's difficult to know what the v1 false positives are.

III. Clustering Abuse

- For any fraud, It only works if the attacker can execute a large amount of the fraudulent actions in a reasonably short time span.
- Fraudulent activity on a site thus differs from legitimate activity in the crucial sense that it is coordinated between accounts.
- The more sophisticated fraudsters will try to disguise their traffic as being legitimate by varying properties of the request.
- However, there is almost always some property or properties of the fraudulent requests that are “too similar” to one another.

III. Clustering Abuse

- The algorithmic approach to implementing this intuition is clustering:
  - Identifying groups of entities that are similar to one another in some mathematical sense.
- But merely separating your accounts or events into groups is not sufficient for fraud detection—you also need to determine whether each cluster is legitimate or abusive.
- Finally, you should examine the abusive clusters for false positives—accounts that accidentally were caught in your net

III. Clustering Abuse

- The clustering process is thus as follows:
  1. Step 1. Group your accounts or activity into clusters.
  2. Step 2. Determine whether each cluster, as a whole, is legitimate or abusive.
  3. Step 3. Within each abusive cluster, find and exclude any legitimate accounts or activity.

III. Clustering Abuse: issues

- **How large does a cluster need to be to be significant?**
- **How "bad" does a cluster need to be to be labeled abusive?**
- This is mostly significant for the supervised case, in which your algorithm will learn about clusters as a whole.
- In some cases, a single bad entity in a cluster is enough to "taint" the entire cluster.
  - One example is profile photos on a social network; nearly any account sharing a photo with a bad account will also be bad.
- In other cases, you will want a large majority of the activity in the cluster to be bad.

Conclusion

- The consumer web (and associated apps) provides a vast array of surfaces that bad actors can use to monetize.
- Machine learning for abuse problems comes with its own set of challenges: getting ground truth data is difficult, and models must achieve a delicate balance between finding what is already known and uncovering new attack techniques.
- Although supervised learning is very powerful for addressing abuse problems, in many cases clustering techniques can be used to address attacks at scale