Shape
Part of F5

Inherent Vulnerabilities Harming Financial Services Companies

An overview of attacks that exploit *inherent* vulnerabilities. What they are, why they happen, how to protect against them, and why traditional countermeasures don't work.

INHERENT VULNERABILITIES OFTEN INCLUDE OFFERING CUSTOMERS THE ABILITY TO:

- Create online accounts and log into those accounts
- Pay bills
- Add and send money to individual payees
- Transfer money between accounts
- Apply for credit
- Access documents that detail accounts, loans, and other financial instruments
- Change their address
- Reset their password
- Chat with customer service

Financial services companies are among the most targeted companies in the world attracting the most sophisticated and well-resourced attackers. One misstep by a skilled and otherwise perfect security team could result in millions of dollars in losses, embarrassing headlines, costly fines, and brand damage lasting decades. It's not enough for security teams to patch vulnerabilities like those described in the OWASP Top Ten. It's not enough to have red teams and blue teams. It's not enough to have a robust bug bounty program. These are all steps in the right direction, but they still fall short, even if perfectly executed, of providing lasting protection for your organization.

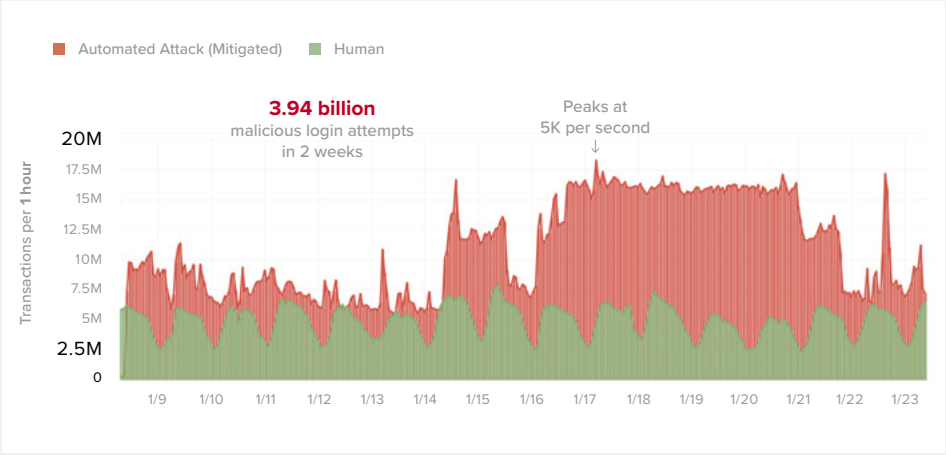
The entire organization–not just the security team–must work together to identify and protect against attacks that exploit *inherent* vulnerabilities. These vulnerabilities are not due to coding errors, misconfigurations, or other mistakes. They cannot be patched in a traditional sense because they result from valid and often critical business requirements. Otherwise known as business logic attacks or application abuse, attackers can exploit features or functions in unintended ways.

Shape Security, now part of F5, protects its financial services customers from online fraud and abuse. A byproduct of this protection is visibility–billions of transactions from web and mobile applications pass through the Shape Network every day. This visibility gives Shape unparalleled insight into the sorts of attacks that target these *inherent* vulnerabilities across the entire financial services industry. The key is for security teams to understand these attacks, why they happen, how to protect against them, and why traditional countermeasures alone don't work.

Credential Validation Attacks, A.K.A Credential Stuffing

Bad actors obtain hundreds of thousands, millions, or even billions of username password pairs from the dark web or a compromise at a poorly secured organization. They then use automation to try those username password pairs against the login applications of entirely different organizations (those that were not part of the original compromise). Because of consumer habits to reuse passwords, these attacks typically identify valid credentials anywhere from 0.1% to 3% of the time. This success often leads to account takeover (ATO) and subsequent monetization of whatever assets are within the compromised account leading to fraud losses, loss of customer trust, and damaged brand.

Figure 1: Credential Stuffing attack against a top tier financial services company in the United States.



FINANCIAL SERVICES COMPANY ASSETS CAN INCLUDE:

- Currency
- Credit
- Access to accounts for other malicious activities (e.g., money laundering, synthetic identities, bust-out fraud)
- Details about loans or other financial instruments
- Personally identifiable information (PII)

ATTACKS VIA FINTECHS

Attackers don't always launch Credential Stuffing attacks directly against the Login application. They sometimes launch their attacks through a fintech, such as a loyalty point or financial aggregator, or a bill payment service. Examples include Mint, Prism and AwardWallet. To use these free services, subscribers must first "link" their accounts at their bank, retailer, hotel, airline, telecommunications provider, etc., by giving the fintech their username and password for each account. The fintech then attempts to programmatically log into each account. If the subscriber provides the correct username and password, the linking process continues. After an account is linked, the fintech uses automation to repeatedly log into the account and scrape the contents–sometimes thousands of times per day.

Bad actors have learned to exploit this linking process to engage in Credential Stuffing. They simply create an account at the fintech and then give the fintech the username and password they are trying to validate. If the linking process continues, the bad actor knows the username and password are correct. Fintechs are aware of this activity but most are doing little to stop it.

Figure 2 shows the login traffic at a top three bank in the United States. Notice the small anomalous spike in the fintech traffic.

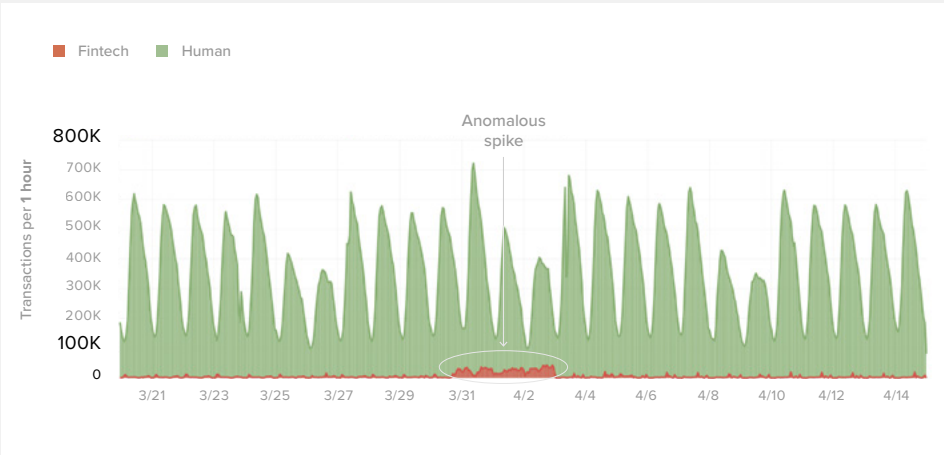


Figure 2: Login traffic at a top three bank in the United States. Human logins are in green and fintech logins are in red.

Figure 3 shows only the Fintech traffic, which makes the anomalous spike more pronounced.

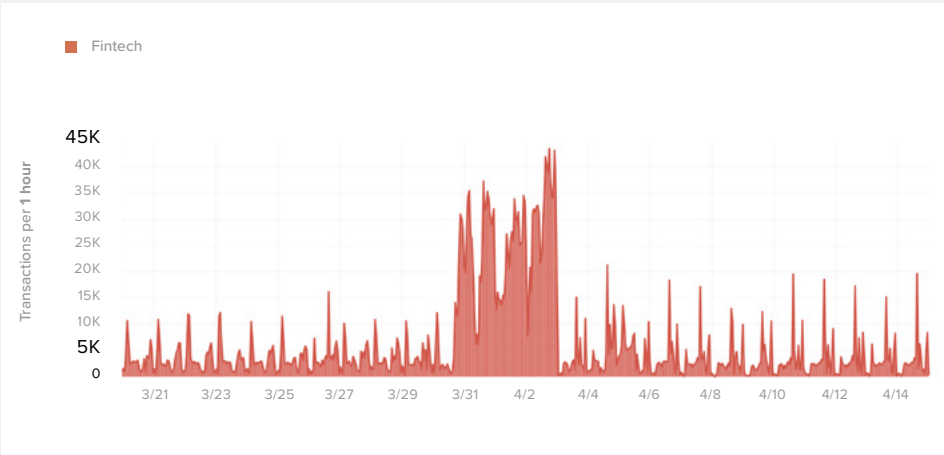
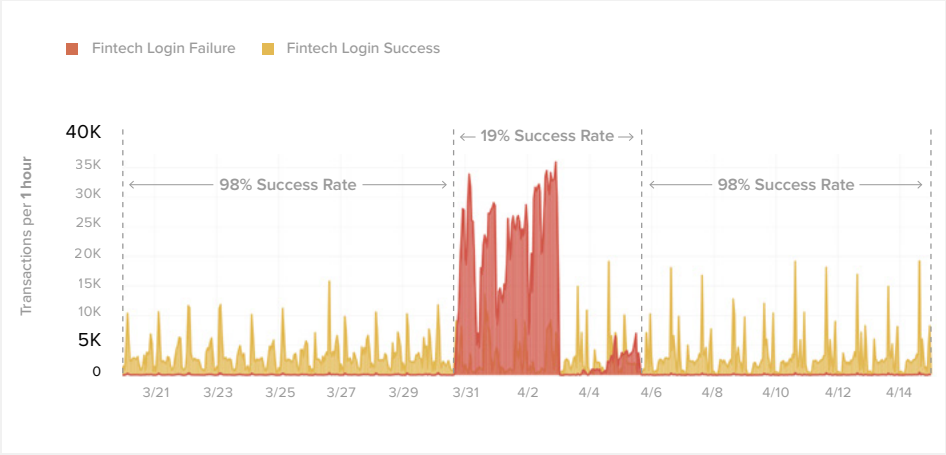


Figure 3: Fintech traffic at a top three bank in the United States.

Finally, Figure 4 shows the login success rate for the fintech traffic. Since fintechs are logging in programmatically and have the username and password stored, their login success rates are typically 98%-99%; however, during the spike in fintech traffic the login success rate dropped to 19%. This rate is a combination of the 98% login success rate of the legitimate fintech traffic and the 0.1-3% login success rate of the credential stuffing attack. As further evidence that this was a credential stuffing attack through a fintech, 71% (54K) of the accounts attempted during the anomaly had never before been attempted. The low login success rate, the introduction of new accounts, the fintech's inability to prevent malicious automation, and

Figure 4: Login success rate of the fintech traffic.



subsequent confirmation from the Shape customer all indicate this was a credential stuffing attack *through* a financial aggregator.

SHAPE HAS SEEN FINTECHS PROGRAMMATICALLY LOG INTO AND SCRAPE CUSTOMER ACCOUNTS THOUSANDS OF TIMES PER DAY.

One secure way for fintechs to gain access to customer accounts is using OAUTH 2.0, but unfortunately, many fintechs refuse to adopt it.

It's important to remember that consumer logins are not the only login applications targeted by attackers. It is quite common for attackers to also target logins for employees, partners, suppliers and contractors. Taking over these sorts of accounts often leads to invoice/payment fraud, theft of intellectual property, breaches of information security, or the advancement of other complex multi-step schemes. These schemes are typically executed by well-resourced criminal organizations familiar with the targeted enterprise's business processes and industry-specific terms of art, rely heavily on social engineering, and often lead to significant or even incalculable losses.

As one example, a criminal organization in the United States used the information found on rejected mortgage modification applications to target the applicants with an elaborate social engineering scheme. The criminals called the victims, used the information in the applications to gain the applicants' trust, and then exploited that trust by instructing the applicants to stop making their mortgage payments, and instead, make lower "trial" mortgage payments to an alternate address. The fraudsters promised their victims that if they successfully made three trial payments it would lead to a permanent mortgage modification. Several thousand people were defrauded out of millions of dollars and many lost their homes to foreclosure. Loss of personally Identifiable Information (PII) and "inside" information about loans or other financial instruments can lead to massive amounts of fraud, class action lawsuits, and embarrassing headlines.

As described above, Credential Stuffing attacks typically exhibit a login success rate between 0.1% and 3.0%. Attackers know this is anomalous and could reveal their attack to security personnel, so they look for ways to increase their login success rate. Common techniques for doing this are using canary accounts or launching an account enumeration attack prior to the Credential Stuffing attack.

CANARY ACCOUNTS

A canary account is an account that is reliably under the control of the attacker; that is, they know the correct username and password. During the attack, the cyber criminal tries usernames and passwords from the list of stolen or purchased credentials, which exhibit a 0.1 to 3.0% login success rate. They then use the same infrastructure to successfully log into one or more canary accounts several times, which exhibit a 100% login success rate. This results in an increased net login success rate for all transactions from the same infrastructure.

There is an additional benefit to an attacker who uses canary accounts. If during an attack, the attacker attempts to log into a canary account and receives the server response, “Failed login. Incorrect username or password,” the attacker now knows the attack is being mitigated. As soon as an attacker received that feedback they would know to retool to circumvent security defenses. The goal is to mitigate logins into customer accounts but to not mitigate logins into canary accounts in order to minimize the information provided to bad actors.

Accounts with thousands of successful logins with little to no legitimate customer activity might be canary accounts. The metadata surrounding the creation of a canary account can sometimes shed light on the true identity of the attacker. Since creating an account is typically an innocuous activity, bad actors don’t always use the same operational security (opsec) they use during actual attacks.

ACCOUNT ENUMERATION ATTACKS

Another technique used to increase login success rate for bad actors is to first launch an attack against an application that will confirm whether or not the username corresponds to a valid account. If the username doesn’t correspond to a valid account, then the attacker knows not to even try the password because doing so would certainly result in a failed login. Applications frequently targeted with account enumeration attacks are Forgot Username, ID, or Password, and Create Account.

Forgot Username, ID, or Password

Whether or not these applications are vulnerable to an account enumeration attack depends on the feedback they generate when a user attempts to recover the username, ID, or password for an account that doesn’t exist. If the application provides the feedback,

“Sorry, we have no record of that email address!” then it is vulnerable and likely under attack. If it provides more generic feedback, such as, “If we have a record of that email address, we’ll send you an email with instructions on how to proceed,” or something similar, then it is not vulnerable.

Figure 5: Forgot User ID attack against a Global 2000 Shape customer.

The key is to not provide feedback that confirms whether or not the account exists. This can create some friction for customers who maintain several email addresses and don’t remember which email address they used when creating the account. But absent a reliable means for stopping automated attacks, you should keep the feedback generic in order to avoid providing attackers with valuable feedback.

ATTACKS AGAINST THE CREATE ACCOUNT OR FORGOT PASSWORD APPLICATIONS ARE OFTEN A PRECURSOR TO A CREDENTIAL STUFFING ATTACK.

Create Account

Another type of account enumeration attack is when an attacker uses automation to try to create accounts using the same usernames from the list of credentials they plan to use during a subsequent Credential Stuffing attack. If the attacker is able to successfully create the account, that means the account did not previously exist. The attacker removes that specific username password pair in the Credential Stuffing attack because it will certainly result in a failed login. If the attacker receives the message, “Sorry, that account already exists.” or similar feedback, they now know the account exists and it is worth trying the associated password in a subsequent Credential Stuffing attack.

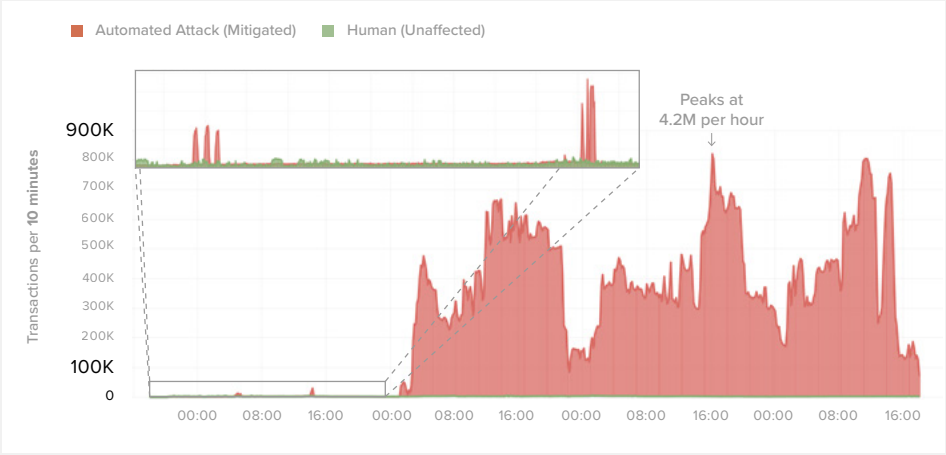


Figure 6: Attack against the Create Account application. The attacker attempted to create roughly 160 million fake accounts.

Attackers don't always have to complete the account creation process to learn if an account already exists. Many applications give the attacker feedback immediately upon entering the desired username in the "choose username" field. If the attacker does complete the account creation, it makes several canary accounts available for their use in subsequent attacks.

OTHER AUTOMATION AGAINST CREATE ACCOUNT

Shape frequently sees automation against the Create Account application for reasons other than account enumeration. This is especially true for financial services organizations that are exploited for money laundering and for the creation and maintenance of synthetic identities.

Money Laundering

Financial institutions require a lot of information during the Create Account process–much more than, say, an online retailer that often requires only a name and an email address, which can be entirely fabricated. In order to open an online checking account at a local bank, for example, the bank might ask for full name, date of birth, taxpayer identification, address, government issued identification, and answers to knowledge-based authentication (KBA) questions. Criminal organizations can obtain answers to all these questions from money-mules–the people who for a small fee will access the account and move money. Sometimes money-mules are asked to create and manage the accounts themselves, but when criminal organizations want maximum control over thousands of accounts, they gather all the required information from the money-mules and use automation to navigate most or all of the Create Account workflow.

Synthetic Identities

Synthetic identities are people who exist only online. They have a face created using artificial intelligence; they have a name, address, phone number, email address, and a credit history; and they have online accounts at banks, retailers, hotels, airlines, telecommunications companies, etc. These synthetic identities are used for both malicious (bust-out fraud) and somewhat benign (influence amplification) reasons. Automation is often used to create and interact with the online accounts to keep the synthetic identities active, and therefore, more realistic.

BUST-OUT FRAUD IS WHEN AN ATTACKER POSES AS A LEGITIMATE CUSTOMER FOR YEARS, BUILDING TRUST AND INCREASING CREDIT LIMITS, AND THEN BORROWS THE MAXIMUM ACROSS ALL ACCOUNTS AND DISAPPEARS.

Automation Against Add Payee and Send Money

There are two reasons why a bad actor would use automation during an attack–*iteration* and *manipulation*. Iteration is when an attacker needs to iterate through a large number of steps in order to realize value, such as during a credential stuffing attack where the attacker needs to try millions of username password pairs and trying each manually would not be feasible. Manipulation is when an attacker needs to perform only one or a few steps in order to realize value, but the targeted web or mobile application is inaccessible to the attacker at the time of the attack. An example of manipulative automation is a Man-in-the-Browser (MitB) attack.

One Shape customer, a top tier financial organization in the United States, asked Shape for help with a specific MitB attack. When banking customers used an infected web browser to log into their accounts, the malware would hijack the login, automatically add a payee and send money to that payee without the victim knowing, and then return control of the browser to the victim.

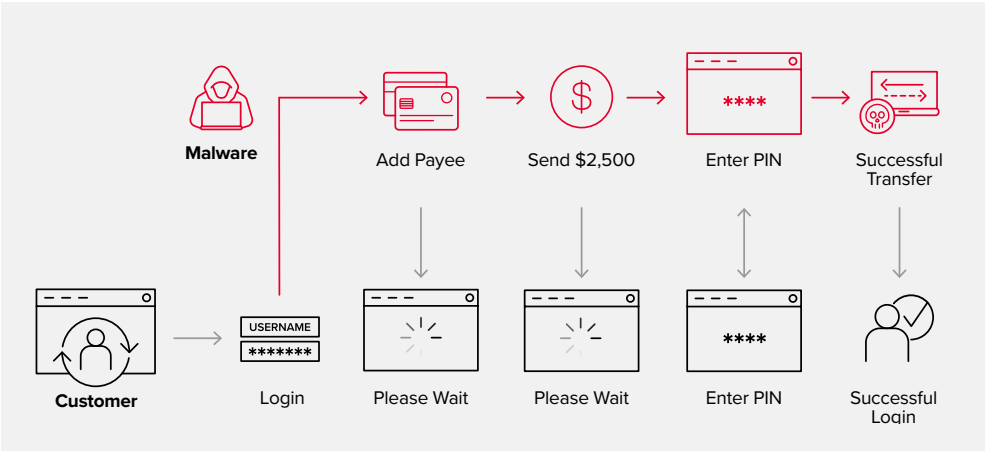
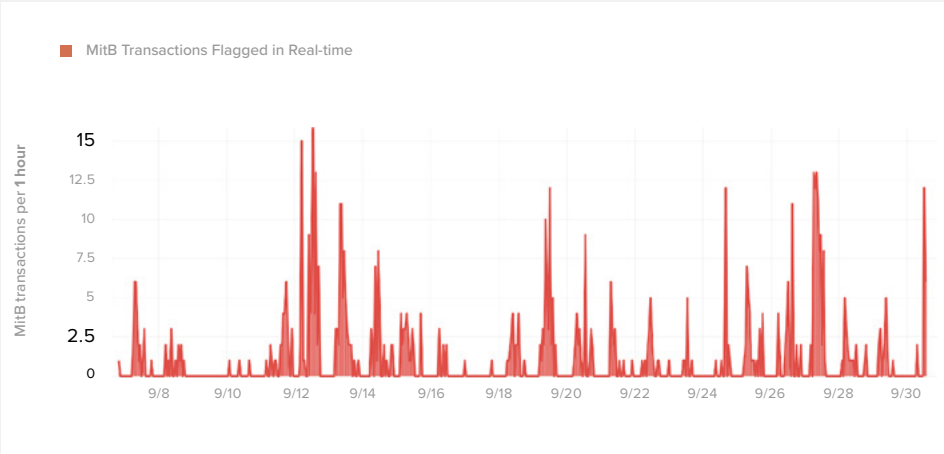


Figure 7: Illustration of how a Man-in-the-Browser (MitB) attack adds a payee, sends money to the payee, and defeats 2FA.

Note the banking customer's experience in Figure 7. The customer enters his username and password and clicks "login." A few seconds later he is asked to enter a 2FA code, which he contemporaneously receives as a text message. The customer enters the code believing it is

part of the login process, but in the background, the malware uses the code to send money to the new payee. As a final step, the malware dynamically edits the HTML to change the account balance shown to the victim.

Figure 8: Transactions associated with a MitB attack against a top three bank in the United States. Shape identified the attack in real time.



SHAPE CAN HELP YOU IDENTIFY THE APPLICATIONS LIKELY TO EXPERIENCE AUTOMATION, QUANTIFY THE AUTOMATION AGAINST THOSE APPLICATIONS, AND CONFIDENTLY CATEGORIZE IT AS GOOD OR BAD.

How to Protect Your Organization

There is no magic bullet that will protect your organization from attacks against inherent vulnerabilities. Protecting your organization from these sorts of attacks results only from a strict adherence to a perpetual three-step process.

Step 1: Gain Visibility

The first and most important step toward protecting your organization is to gain visibility. This requires that you identify the applications likely to experience automation, quantify the automation against those applications, and then understand that automation sufficiently to

confidently categorize it as good or bad. You won't find automation if you are not looking in the right places.

Identifying the applications likely to experience automation requires informed and objective answers to several questions for each and every application. What are the reasons why someone might launch automation against the application? Is there a reason why someone might perceive a monetary or intelligence gathering benefit from using automation against the application? Absent a short term incentive for automation, is there a long term, more strategic incentive? Finding answers to these questions requires fluency in the applications and workflows, as well as a comprehensive understanding of the automation seen in the wild against other financial organizations.

Quantifying the automation is a significant challenge for many organizations for a few reasons. For many years, organizations have blocked unwanted automation by IP address, region, or some other attribute that is trivial for actors to change.

Over time, being blocked by IP address caused actors to become more distributed. Today, rather than submitting 10M transactions using five user agents from five IP addresses in one country, actors submit 10M transactions using nearly 10M user agents from over a million IP addresses in hundreds of countries. The evolution of attackers and the rate at which they retool is nothing short of remarkable.

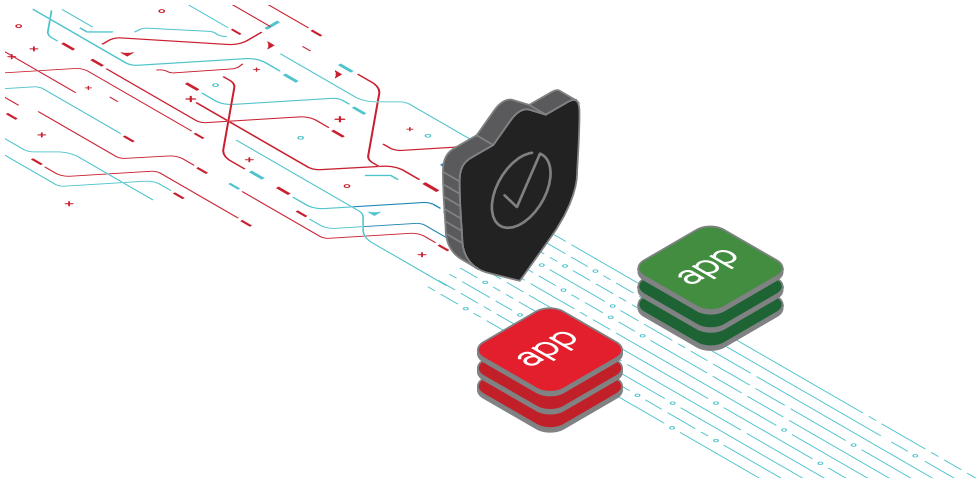
"I THINK 20%–30% OF OUR TRAFFIC IS AUTOMATED."

–Shape customer that was found to have 98% automated attack traffic.

Organizations typically identify the few hundred or few thousand noisiest IP addresses from which automation originates, but they miss the long tail of IP addresses that don't trigger any rate limits, and it's from these IP addresses that most of the automation originates.

Understanding the automation well enough to confidently categorize it as good or bad is also a challenge for many organizations. Identifying approved traffic from known testing and performance monitoring tools is not typically difficult. Identifying the obviously malicious automation, such as Credential Stuffing, is not typically difficult. However, there are often large volumes of automation against some applications where the actors' intentions are not clear. Unsanctioned testing from an employee? A fintech? A competitor? Perhaps a loyal customer who has a legitimate reason to use automation? It's important to understand the automation prior to mitigating it.

Step 2: Take Action on Automation



ONE SHAPE CUSTOMER, A U.S. GOVERNMENT ORGANIZATION WAS UNDER ATTACK FROM IP ADDRESSES LOCATED IN MORE THAN 50 DIFFERENT COUNTRIES. THE SECURITY TEAM DECIDED TO BLOCK ALL TRAFFIC THAT ORIGINATED FROM OUTSIDE THE UNITED STATES. THE NEXT DAY THE ATTACK RETURNED FROM IP ADDRESSES WITHIN THE UNITED STATES.

The second step is to allow-list the good automation and to mitigate the bad automation. There are good and bad ways of doing both.

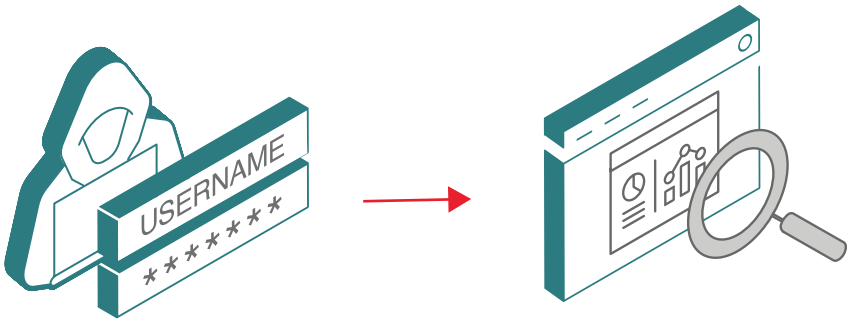
Allow-Listing

You should avoid allow-listing transactions using an attribute that can be easily spoofed, such as a user agent string. Ideally, you should allow-list transactions using a shared secret somewhere in the HTTP header.

Mitigation

You should not mitigate bad automation by dropping the session because that gives the actor immediate feedback that the automation is being mitigated, which incentivizes retools. You want to delay the actor's realization that the automation is being mitigated. Shape accomplishes this by offering several additional mitigation options, such as the ability to redirect or forward the transaction, inject or change something in the transaction and pass it to origin, or respond with a fully formed HTML page. The appropriate action is influenced by the application being protected and whether or not the transaction is associated with the monetization step of a multi-step scheme.

Step 3: Conduct Ongoing Retrospective Analysis



Motivated actors retool. Organizations must conduct ongoing retrospective analysis on transactions hitting the targeted application to quickly identify retools or other unwanted automation. This cannot be done effectively without human-supported artificial intelligence and machine learning systems operating on aggregate transactions.

It is not sufficient to just find unwanted automation–you must also have a means for quickly updating your real-time defenses without impacting legitimate customers.

WHY TRADITIONAL COUNTERMEASURES DON'T WORK

Traditional countermeasures include layer 5–7 defenses, such as web application firewalls (WAF), requiring a second factor of authentication (2FA) for the targeted application, or bot detection and prevention tools such as a CAPTCHA.

Web Application Firewalls (WAF)

A WAF provides robust protection from exploits that target software vulnerabilities or weaknesses. For example, protection from automated attacks that scan the Internet looking to exploit systems susceptible to a newly released Injection vulnerability. However, attacks against inherent vulnerabilities exploit business logic and abuse applications, and typically require additional defenses.

Traditional WAFs look primarily at the Application Layer to defend against the OWASP Top Ten. The Application Layer provides insufficient signal to reliably detect sophisticated automation. Detecting automation today requires additional signals, such as those obtained by collecting behavioral biometrics and interrogating the browser/device environment.

Second Factor Authentication (2FA)

2FA plays an important role in security but widespread use is expensive and creates a lot of friction for customers. While 2FA does make account takeover (ATO) more difficult, it does not always prevent Credential Stuffing.

For example, most 2FA implementations work like this: A customer submits a username and password and if those are correct, the customer is asked to enter the second factor of authentication. If the submitted username and password are incorrect, the customer receives an error message and is not asked to enter the second factor of authentication. The difference in the server response between correct and incorrect credentials tells the attacker if the credentials are correct. Admittedly, the attacker has not taken over the account, but the attacker can now sell the known good credentials to another bad actor who specializes in defeating 2FA (e.g., via port-outs, SIM swaps, SS7 compromises, or social engineering).

CAPTCHA

CAPTCHA is a backronym for Completely Automated Public Turing test to tell Computers and Humans Apart. It comes in a variety of forms as shown in Figure 9.

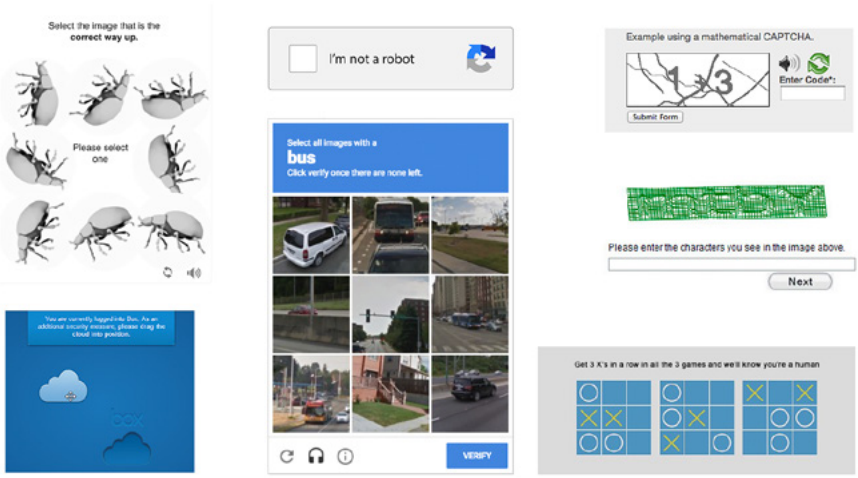


Figure 9: Various forms of CAPTCHA.

NOT ALL AUTOMATION IS MALICIOUS AND NOT ALL MALICIOUS OR UNDESIRABLE TRAFFIC IS AUTOMATED. ORGANIZATIONS NEED SOLUTIONS THAT CAN TELL THE DIFFERENCE.

CAPTCHA, in all of its forms, causes undesirable friction for your loyal customers leading to session and revenue abandonment much like 2FA. Even worse, they do not stop bots. There are dozens of companies that use optical character recognition (OCR), machine learning, and even human click farms to solve CAPTCHAs for a nominal fee. For example, Russian human click farm, 2CAPTCHA, charges as little as 0.5 USD for 1,000 solved CAPTCHAs. Shape produced two short videos to demonstrate just how easy it is to use human click farms to solve CAPTCHAs. The first video is a high level summary and the second video provides a line by line explanation of an actual automation script.

What to Expect After Mitigating Unwanted Automation

Shape has found that after successfully mitigating unwanted automation against web and mobile applications, and after doing battle with actors as they retool over weeks and even months, that many actors continue their activity manually using human click farms. For this reason, Shape offers multiple products to protect its customers from all of these threats.

Shape invisibly protects every application from attack, fraud, and abuse. By defending the world's largest companies, Shape has developed expertise in not just knowing whether a request is coming from a bot or human, but whether the request was made with malicious or benign intent. This ability allows Shape to prevent fraud in real-time, manage fintech and partner access to your applications, and provide new data that powers customer and business insights.

To learn more, explore F5 Application Security.