



E-BOOK

12+ Ways to Hack Multi-Factor Authentication

by Roger Grimes

Do You Know the 12+ Ways to Hack MFA?

KnowBe4

All multi-factor authentication (MFA) mechanisms can be compromised, and in some case, it's as simple as sending a traditional phishing email. Want to know how to defend against MFA hacks? [Learn more now>](#)



SOCIAL ENGINEERING HACKS

- Fake the Authentication Page 11
- Recovery Question Attacks Page 23
- Social Engineer Technical Support Page 25



TECHNICAL HACKS

- Session Unique Identifier Prediction Page 9
- Man-in-the-Endpoint Attacks Page 11
- Malicious MFA Software or Hardware Modification Page 12
- Duplicate Code Generators Page 18
- Skimming Attacks Page 21
- Subject Hijacks Page 25
- Brute Force Attacks Page 34
- Buggy MFA Page 35
- Physical Attacks Page 36



MIXED HACKS

- Session Hijacking Page 9
- SIM Swaps Attacks Page 13
- Downgrade and Recovery Attacks Page 22

KEY TAKEAWAYS



MFA isn't unhackable

- MFA does not prevent phishing or social engineering from being successful.

MFA is good

- Everyone should use it when they can, but it isn't unbreakable.
- If you use MFA, security awareness training still has to be a big part of your overall security defense.

E-BOOK: 12+ Ways to Hack Multi-Factor Authentication, by Roger Grimes

Contents

Summary	2
Introduction	2
Multi-Factor Authentication Basics	3
Authentication Basics	3
Identity	3
Authentication	4
Access Control Token	4
Authorization	4
One-Way vs. Two-Way Authentication	6
Authentication Factors	6
One-Factor to Multi-Factor	6
In-Band vs. Out-of-Band Authentication	7
Hacking Multi-Factor Authentication	8
General Ways to Hack MFA	8
Session Hijacking	9
Session Unique Identifier Prediction	9
Session Hijacking Proxy Attack	9
Kevin Mitnick Hacking MFA Video	10
Fake the Authentication	11
Man-in-the-Endpoint Attacks	11
Banco Trojans	12
Malicious MFA Software Modification	12
Malicious MFA Hardware Modifications	13
SIM Swaps Attacks	13
SMS Rogue Recovery	15
Duplicate Code Generators	18
Shoulder Surfing	19
Skimming Attacks	21
Downgrade and Recovery Attacks	22
Recovery Question Attacks	23
Social Engineer Tech Support	25
Subject Hijack	25
Microsoft Smartcard Identity Hijack Attack	26
Re-Created Biometrics	33
Stolen Biometrics	34
Brute Force Attacks	34
Buggy MFA	35
ROCA Vulnerability	36
Other Physical Attacks	36
Electron Microscope Attack	36
Cold Boot Attacks	36
Summarizing Defenses Against MFA Attacks	38
Social Defenses	38
Technical Defenses	38
Conclusion	38

Understanding the local threats which have made it past your existing deployed defenses and how your employees are responding is far more important than relying on global statistics of all the world's threats against all organizations.

Summary

All multi-factor authentication (MFA) mechanisms can be compromised, and in some cases, it's as simple as sending a traditional phishing email. This white paper covers over a dozen different ways to hack various types of MFA and how to defend against those attacks.

Introduction

Decades of successful attacks against single-factor authentication methods, like login names and passwords, are driving a growing widespread movement to more secure, multi-factor authentication (MFA) solutions. Although MFA solutions have been available for decades, due to a variety of reasons, there is now an ongoing, wide scale, rapid adoption of MFA in both corporate environments and by internet websites.

This trend is exemplified by the fact that over the last few years, the most popular websites and services, including those owned by Google, Microsoft, Facebook, and Twitter, have offered MFA solutions to their customers. Many internet sites and services now offer both traditional login name/password solutions and more secure, MFA options.

Some large companies, like Google (<https://krebsonsecurity.com/2018/07/google-security-keys-neutralized-employee-phishing/>), are reporting great success in defending against some common hacking attacks by moving their user base from single-factor to multi-factor authentication. MFA solutions are supported by default in the most popular operating systems, and additional MFA solutions are offered by hundreds of third-party vendors. Common open MFA standards, such as those promoted by the FIDO Alliance (<https://fidoalliance.org/>), are being widely adopted.

MFA was previously used (mostly) for organizations and websites needing the highest security assurance. Today, MFA tokens are being offered or used by ordinary organizations and websites, and MFA tokens can be purchased as low as a few dollars per device. Many consumers trust the security of MFA solutions so much that they are purchasing and using MFA, when possible and allowed, on all the websites and services which allow it.

The broader adoption of MFA is a positive development for computer defenses and will defeat many of the threats that would otherwise be more readily successful against single-factor authentication solutions. All other

things considered equal, all admins and users should consider and use MFA solutions instead of single-factor authentication solutions to protect sensitive data.

With that said, the ability of MFA to reduce computer security risk has been overly stated by many vendors and proponents, leading to a misunderstanding that the application of MFA means all attacks that were successful against single-factor authentication cannot be successful against MFA. For example, many MFA admins and users believe that email phishing is no longer a threat because users cannot be phished out of their login credentials. This is not true.

While MFA does reduce, and in some cases, significantly reduce particular computer security risks, most of the attacks that could be successful against single-factor authentication can also be successful against MFA solutions. In this paper, we introduce over a dozen ways to attack different MFA solutions. Often, a single MFA solution is susceptible to multiple exploitation methods. While this paper is not inclusive of all methods, the hacking methods included are representative enough to demonstrate that MFA, while more secure than single-factor authentication solutions, is not unhackable.

This white paper is dedicated to describing over a dozen different ways to hack MFA solutions and how to defend against those attacks. In order to understand the different methods used to hack MFA, it helps to understand the basic components of authentication and MFA.

Multi-Factor Authentication Basics

This section discusses authentication basics, in a general sense, and then covers MFA in particular.

Authentication Basics

Authentication is the process of a subject (e.g., user, device, group, service, etc.) proving ownership of a particular identity. Let's break it down further.

Identity

An identity is any unique (to the involved namespace) label identifying a specific subject. An identity is often represented by a login name (ex. rogerg), email address (ex. rogerg@knowbe4.com), or unique series of characters, but can be any unique, previously agreed upon, label within the same namespace.

A namespace is an organized system to help collect, identify, and locate specific entities and their related attributes. Common name spaces are Domain Naming System (DNS), Microsoft Active Directory, and Lightweight Direct Access Protocol (LDAP) databases. A namespace may contain more than one identity label per subject (for example, Active Directory can use DNS, LDAP, email address, and User Principal Name (UPN), but each label must be unique in the same namespace and can only represent a single subject).

All authentication and access control steps involve one or more identities. All authentication involves an identity label, which uniquely identifies the subject doing the authentication. Identities have to be created prior to or as part of the initial authentication process. The identity should be different than what is being provided to prove ownership of the identity. For example, in Microsoft Windows, although a subject might use a fingerprint to authenticate (i.e., prove ownership of the identity), the label attached to an authentication attempt will probably be the user's Active Directory login name, their UPN, or their email address. The identity label is very important in the authentication process.

Authentication

Authentication is the process of a subject proving proof of (sole) ownership of an authentication identity within a namespace in order for the identity and its associated permissions, memberships, rights, and privileges, to be used in access control authorization operations related to that namespace.

The identity and the proof of ownership of the identity must have been previously, hopefully securely, stored in at least one location (e.g., table, database, registry entry, etc.), to be used in future authentication challenges. The storage of the authentication proofs is often not stored on the server/service/site directly involved in the authentication and is instead stored on third-party server/service/sites involved in the authentication, which both parties (server and client) trust.

Each storage location is a potential attack vector for compromising authentication. Anyone using authentication should think about where the authentication proofs are stored, who has access to those locations, and how trustworthy the storage of those credentials should be considered. Storage of authentication secrets should always be restricted to the bare essential number of administrators and aggressively monitored and audited. For if the authentication secrets are compromised, the authentication process can no longer be fully trusted.

Authentication can be successful or unsuccessful. Only successful, legitimate authentication is supposed to lead to the next process.

Access Control Token

After a successful authentication, in most cases, the access control process then associates an access control object (e.g., token, ticket, etc.) to the tested identity. What this access control token contains varies by system and protocol. In some systems, it may only contain another unique identifier, such as a series of numbers or characters. In other systems, it may contain a list of group memberships, permissions, privileges, and other needed information.

The token may or may not have a pre-determined maximum lifetime, which upon expiration, forces the subject to re-authenticate to remain in an “active” session. In Microsoft Windows, an access control token may arrive in the form of a Kerberos ticket or an NTLM or LM token. On websites and services, most access control tokens are represented by an HTML cookie, which is a simple text file.

After a successful authentication, the user is given a session access control token, which is then used for the rest of the authentication cycle.

Authorization

Authorization is the process of comparing the now successfully authenticated subject's access control token against previously permissioned/secured resources to determine the subject's access to those objects. In most cases, once a subject has been handed an access control token, the subject (or in reality, a process or program on behalf of the subject) submits the access control token for authorization and the subject does not need to reauthenticate until the expiration of the token. Once an access control token has been issued, authentication is not tested for each and every authorization access attempt. Possession of the access control token is considered proof of successful authentication.



HUGELY IMPORTANT POINT!

No matter how a person successfully authenticates, be it simple password, biometrics, or a multi-factor authentication token, once the authentication is successful, the authentication token assigned to the identity is usually the same for all authentication methods and often bears little resemblance to the authentication method used.

For example, suppose a subject uses his/her fingerprint to login to Windows and Active Directory using his/her laptop and the laptop's built-in fingerprint scanner. The authentication process happens locally on the laptop. The laptop's fingerprint recognition and authentication software and hardware combination successfully authenticate the user. At that point, the user's fingerprint is no longer used anymore. The fingerprint is not sent around the network to be involved in access control operations. The user's fingerprint is not copied or sent to another networked computer, so the user can access a file or folder.

Instead, once the user has been successfully authenticated using his/her fingerprint (or whatever), the Windows operating system issues them a Kerberos ticket or NTLM or LM token. It is that resulting ticket or token that the user (or more accurately written, processes or programs acting on behalf of the user) use for all access control authorizations. And if an attacker can access the access control token, they don't care how you authenticated. Possession of the token, from legitimate means or not, is usually treated by authorization processes the same as if the holder of that token successfully authenticated. The authorization process does not have a way of knowing whether or not the current holder of that access control token was the legitimate user or ever successfully authenticated. This key fact is often used by hackers to compromise multi-factor authentication.

There is a huge difference between the authentication method being used to authenticate and the resulting access control token that is used for authorization afterward.

This same concept applies more generally to the entire authentication process. Attackers exploiting authentication often look for weaknesses in implementations along the entire process. They will look to see if there are gaps in the linkages between the identity, authentication, and authorization...and there often is as you will see below.

A key way to hack MFA is to look for weaknesses in the entire authentication process, from identity registration, authentication secret storage, authentication, and authorization.

One-Way vs. Two-Way Authentication

Authentication is normally conducted between two or more parties, often referred to as the server (the object/application/process being authenticated to) and client (the object authentication to the server) and can be one-way or two-way. Many authenticating objects can act as both a server or a client depending on the reason for authenticating. This is to say that a physical server isn't always acting as a server and vice-versa. Additional servers may be involved in the authentication process, and so there may be multiple authentications occurring during a single authentication event. A good example of that is Kerberos, where the client must authenticate to the Kerberos authentication server as well as the intended target server.

Most authentication is one-way, meaning the client authenticates to the server or the server authenticates to the client, but the opposite is not true, at least during the same authentication event. A very common example of this is web servers using HTTPS. When HTTPS is involved, the web server has an HTTPS/TLS digital certificate, which is linked to its identity (usually its DNS address). When a client connects to the web server over HTTPS, the server sends its HTTPS digital certificate to the client, to prove its identity and to secure an encrypted channel in which to generate symmetric keying material. The client receives the web server's HTTPS digital certificate and verifies its trustworthiness. If successful, the client will trust the server to be the server it says it is (based on the subject's identity). In one-way authentication, the client does NOT prove its identity to the server, at least within the same transaction.

With two-way, "mutual" authentication, both the client and server authenticate to each other as part of the same authentication process. If one side fails, the other side automatically fails.

Authentication Factors

Proof of ownership of an identity is made by a subject supplying the identity and one or more authentication factors. An authentication factor is something supplied which only the subject knows or can supply, and by doing so, proves sole ownership of the authenticated identity. In general, there are only three basic types of authentication factors, widely known as:

- Something You Know
 - Examples include: Password, PIN, Connect the Dots*
- Something You Have
 - Examples include: USB token, smartcard, RFID transmitter, dongle*
- Something You Are
 - Examples include: Biometrics, fingerprints, retina scan*

There are only three major types of authentication factors, as described above. You will sometimes hear about MFA solutions that have more than three factors (e.g., five-factor), but what these solutions are referring to are multiple instances of the same three factors. In order for the factors to be most protective in an MFA solution, the factors should be different.

One-Factor to Multi-Factor

The concept is that the use of two or three of these factors makes a hacker's job more difficult. For example, the hacker may be able to con you out of a password, but it will take additional effort to also steal your hardware token if that is used in a MFA solution. Or if a malicious individual picks up your MFA hardware token, it would be useless to him/her if he/she did not also have your associated PIN that is required to use it.

There are single-factor hardware solutions that look like an MFA solution, but don't require an additional factor. For example, existing versions of Google Security Keys™ and Yubikeys™, can be used for one-factor or multi-factor. In their one-factor implementations, it means that if a person finds those hardware devices, if he/she is not otherwise secured, it means he/she can use them and take-over the digital identity associated with the token. It might be more difficult for a hacker to obtain another person's single-factor hardware token than phishing him/her out of a password online, but once obtained, it would mean immediate compromise of that identity. All other things being equal, MFA is always better than single-factor authentication for better security, although MFA is rarely universally allowed across all scenarios.

Although MFA solutions should always strive to require multiple factor types, even multiple instances of the same type of factor can improve security over single-factor authentication solutions. But readers should not equate multiple uses of the same authentication factor as equivalent to the security given by additional authentication factor types. For example, if a user is required to use both a password and a PIN to login (both the same type of authentication factor ("Something You Know")), then he/she can be phished out of both almost as readily as one. It's the additional factor types that provide the most protection because they require that the hacker do something completely different in order to be successful.

In-Band vs. Out-of-Band Authentication

Authentication factors can be considered in-band or out-of-band. In-band authentication means that the authentication factor method being used is conducted over the same communications channel as the primary login method. Out-of-band authentication is when the authentication factor is being sent over a channel different than the primary login channel.

For example, if you're trying to login to an internet service application and you are required to type in a password and a password recovery answer within the same browser, this is considered two instances of the same factor, both in-band. If, however, you are required to type in a password on your computer and also a second PIN code that was sent to your external cell phone, the second factor is considered out-of-band.

Even better, if you are required to respond to both separate band authentication factors ONLY in those channels, and they aren't "cross-channel" (i.e., authentication factor sent to you out-of-band can only be responded to in the same band as the other factor), then it provides even more security assurance. Authentication factors sent on the same device, even if in different channels, are not considered as secure as authentication methods using different channels over different devices.

All things being equal, MFA is always better than single-factor authentication for security reasons.

As the number of separate authentication factors and communication bands increase, so too, does security assurance. In most scenarios, using an MFA solution can only improve security, and MFA should be used where and when it makes sense to do so. Unfortunately, not all authentication scenarios allow MFA, and often times not the same MFA solution. At least for now (2018 and the next few years), users will still be required to use a single-factor authentication method in many scenarios.

Even when MFA is allowed and used, it can be hacked, sometimes just as easily as single-factor authentication solutions. MFA is good, but don't over rely your security assurance on it. It's a good tool to increase security, but there is a huge difference between MFA improving security assurance and MFA being unhackable. Understanding the difference is crucial to all entities and security administrators relying on MFA solutions. The key is not to overly rely on MFA as a security savior.

To put this in perspective, most companies that use MFA solutions still get hacked. This is because the most popular reasons for being compromised (e.g., social engineering, client-side attacks, unpatched software, and coding bugs) are not fully mitigated by MFA. MFA can reduce, sometimes significantly, some forms of hacking. But if the companies involved don't put down the biggest reasons why they are successfully hacked, then MFA will not prevent the hackers and malware from being successful. MFA is good, but it is just one piece of a big puzzle to solve. MFA cannot, by itself, make a company "unhackable". Indeed, MFA itself is not unhackable.

Most companies that use MFA are still successfully hacked.

Hacking Multi-Factor Authentication

This section of the white paper will discuss over a dozen ways to hack MFA solutions. The majority of these attacks have been successfully used against millions of MFA-protected users. Most of the attack methods will include links to news reports and examples of their exploitation. The theoretical attacks that have not been used in a public attack, yet, are noted as such.

In most cases, a particular type of MFA solution is susceptible to multiple hacking methods, and thus the attacks are not 1:1 only against a single type of MFA solution. Each attack is shown against the MFA method that it is often used against, but often can be used against other MFA solutions.

General Ways to Hack MFA

When thinking about how MFA solutions are hacked there are three general ways:

- Social Engineering
- Technical
- Mixed

Social engineering refers to the involved human element using the MFA solution inadvertently in a way that results in its bypass or misuse. Technical manipulation refers to the methods of exploitation and manipulation that did not require that the human user make a mistake. Many of the hacking methods presented below require a mixture of both human and technical weaknesses.

No matter what the hacking methods are, they are attempts at taking advantage of weaknesses between the steps of authentication: identity, authentication secret storage, authentication, or authorization. The attacks are malicious interruption, modification, or false representation of one or more of those steps or transitioning between those steps.

Note: Often times an MFA solution provider will defend their solution against a successful demonstrated hack by saying that their MFA solution, itself, didn't fail. And while this may be true in the technical sense, MFA solutions are not ultimately tested in sterile laboratories where only direct attacks count. If the MFA solution fails the user for any reason, in the user's mind, the MFA solution has failed. He/she doesn't care so much about the details of whether or not the MFA

solution itself was technically responsible. The user only knows that it failed him/her.

Session Hijacking

Session hijacking is a hacking method where after a successful, legitimate authentication, the legitimate user's session is hijacked by an unauthorized party. It is often due to the resulting access control token being stolen. It can be initially transparent to the user or the user may unwittingly participate in his/her own hacking by responding to something as simple as a traditional phishing email. No matter how it is done, once the unauthorized attacker has either gained control over or copied the access control token, the unauthorized intruder can seize the session away from the legitimate user or fraudulently manipulate it. When a session has been hijacked, the attacker essentially assumes the hacked user's identity for the entirety of the session. Session hijacking has been around for decades and is one of the most common forms of authentication hacking, and it can be just as successful when used against MFA as well. Session hijacking can be accomplished using a variety of different methods, including:

- Session Unique Identifier Prediction
- Theft of the session token on the network communication channel
- Theft of the session token on the end-point

Session Unique Identifier Prediction

Every time a user successfully authenticates to a website, using MFA or not, he/she gets sent back what is supposed to be a unique session token (i.e., cookie) or URL string, both of which are supposed to contain a randomly selected, unique identifier which specifies the legitimate user and his/her session to the website. It's important that the unique identifier not be predictable enough that other third parties (i.e., hackers) can predict what other people's tokens or URL strings are or will be.

Session hackers look for websites with predictable unique identifiers. Hackers usually do this by joining a targeted website as multiple, different, authenticated users, and look for commonalities between the unique identifiers put in the cookie or URL string for each user. Sometimes there is no randomness at all, and the numbers are perfectly sequential and predictable between different users. Once an attacker recognizes the pattern, he/she will simply try different identifier numbers to see which ones will give him/her the desired target account or access.

User session identifiers must always be unique and randomly generated so that hackers can't predict them.

With this sort of attack, the attacker can "guess" every user from within the safety of their own location without ever involving the targeted victim. So, with or without MFA being involved, if the attacker can predict the supposedly unique identifying information after a successful authentication, they essentially "become the user", at least for the session, if not for all sessions. While this type of security coding mistake isn't as popular as it was decades ago, it still happens frequently enough because not all website coders are aware of the issue.

Defense: User session identifiers must be unique and randomly generated.

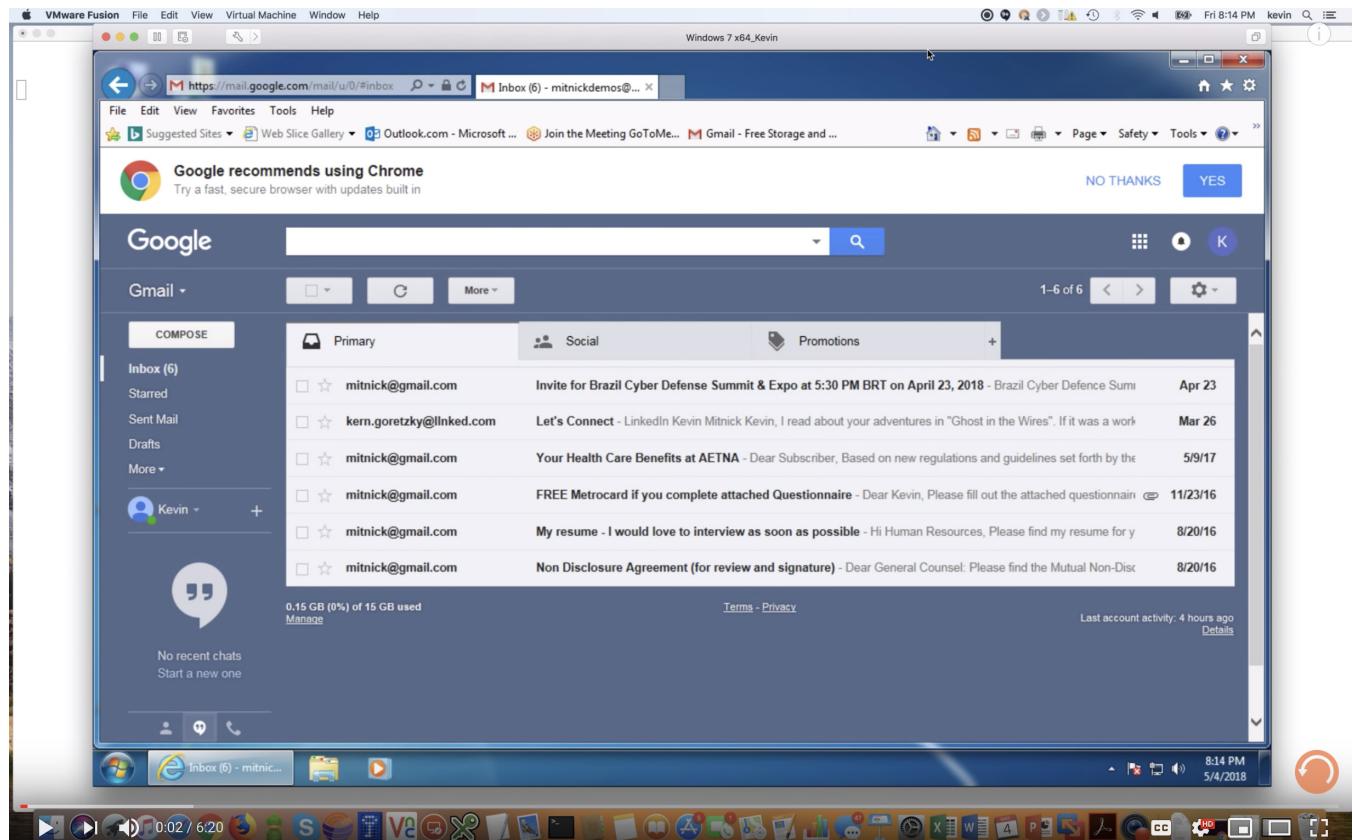
Session Hijacking Proxy Attack

With this type of hack, the hacker must first successfully establish themselves in between the client and server (i.e., a man-in-the-middle attack (MitM)). Establishing a MitM attack isn't as difficult as most people think. It can be accomplished locally in any shared wireless network

environment, like in a coffee shop, using a variety of free hacking tools. Remotely, across the internet, it can be accomplished by sending the victim a phishing email to entice them to visit a fake “look-alike” or “sound-alike” URL website. Once the user has been a victim of a MitM, everything the user sends can be intercepted by the MitM proxy service.

Kevin Mitnick Hacking MFA Video

But the best way to explain this type of attack is to watch one in action.



Here is a video by KnowBe4’s Chief Hacking Officer, Kevin Mitnick, (<https://blog.knowbe4.com/heads-up-new-exploit-hacks-linkedin-2-factor-auth.-see-this-kevin-mitnick-video>) showing how to phish around a victim’s MFA solution. Kevin will show you in six minutes how easy it is to do. After you watch the video, come back here to see the summary steps of how it was done. In this example, Kevin does the following:

1. Kevin set up a fake look-alike/sound-alike website that was really an evil proxy
2. Tricked user into visiting evil proxy website
3. User typed in credentials, which proxy, now pretending to be the legitimate customer, presented to legitimate website
4. Legitimate website sent back legitimate session token, which Kevin then stole and replayed to take over user’s session

Kevin used Evilginx (<https://breakdev.org/evilginx-advanced-phishing-with-two-factor-authentication-bypass/>) for his MitM proxy hacking tool, but there are dozens to choose from. This is but one example hack out of the dozens, if not hundreds, of ways to do session hijacking, even if MFA is involved. Common to all of them is the capture of the session information in order to steal the session.

Defenses: Use mutual, two-way authentication and educate end users to avoid social engineering attacks.

Fake the Authentication

One of the easiest to accomplish hacks is no hack at all. In this particular scenario, the entire authentication experience the user is presented with is completely or partially faked by a bogus look-alike website. The victim is tricked into visiting a site for which they have an MFA token. The fake site then simulates the entire normal login experience. The user may put in a login name and/or password, and then be (fake) prompted for their MFA solution answer. The fake website then acts like the login experience was successful and then takes the victim to whatever landing page the attacker wants.

For example, the user thinks they are authenticating to a website that participates in the Google Authenticator MFA app. When the user opens up the Google Authenticator MFA app, it presents a 6-digit code, which can be used and typed into any website, whether or not the website actually participates in Google Authenticator MFA authentication. The user is prompted to enter his/her Google Authenticator codes and enters them without knowing the entire experience has been faked.

Using this sort of attack, the fake, rogue website doesn't capture the user's resulting session token. And because of this, they can't log in to the user's real website or take control over the user's real session. They also cannot display the normal, user-related content that the user would usually expect to see. The attacker's website can act like additional security information is needed, and prompt the user to enter in additional "qualifying" information which the attacker wishes to capture, such as password recovery reset questions and/or answers, social security information, credit card information, etc. After capturing the user's confidential information, the attacker can make it seem like the website has experienced an error and redirect the user to the real website.

Some MFA solutions try to avoid this sort of trap by not sending an MFA answer unless the real, previously registered website is involved. That way, the user isn't given a code to even participate in the fake authentication unless the request is initiated by a trusted website. Rogue websites have gotten around this protection by sending the user's typed in information to the user's real, previously registered, website, to get the MFA solution to generate a code as the user expects. This hack can even be used to then log in to the other intended, legitimate website as the user interacts with the fake web page.

Some MFA solutions fight this type of attack by using a variety of different methods. One, the MFA solution can send the URL of the website being logged into along with the user's detected location. A MitM attacker would likely have a location different than the victim. Thus, if the user sees his/her supposed logging-in-device as being in a completely different location, he/she can suspect a MitM attack is occurring. However, hackers can pass along the user's current location as well, and the real website would be none the wiser. There are other possible protections, but each creates increasing user friction, frustration, and lengthens the login process. This type of MFA hack is very difficult to avoid as the MFA solution is not really being hacked.

Man-in-the-Endpoint Attacks

This is a general attack category, which basically says, if the attacker gets admin access on a device, nothing the device does can be trusted. The hacker can do anything the logged-on user can do, so if the user authenticates to a website or application, the hacker can essentially piggyback in on that authentication to do whatever the legitimate user can do. A "local" hacker can even directly steal the issued session tokens, and mimic the attacks described in the previous section, but without having to do a MitM attack first.

Banco Trojans

A Man-in-the-Endpoint attacker can also start a second hidden browser session, while the legitimate authenticated user uses the first session. This is a method often deployed by banking trojans (also better known as bancos trojans, which are very popular in South America). The bancos trojan would break in and exploit the local computer using any method that traditional malware uses to break in, such as social engineering or unpatched software. Then the trojan monitors the current user's browsing selections, looking for previously defined keywords, such as "bank", "Bank of America," and so on.

When the bancos detects a monitored keyword indicating the user is logging into a targeted financial institution, the bancos trojan starts a second, hidden, browser session. And while the legitimate user, using MFA or not to logon, simply looks at his/her account, the bancos trojan is changing his/her contact information and initiating a large wire transfer of the user's funds to another rogue bank account. If the bank calls or emails to confirm, they end up using the new, fraudulent contact information.

Banks fought back by sending "authentication codes", very similar to today's SMS MFA messages, that would only be good for a particular transaction. Bancos trojans responded by waiting for the user to conduct a transaction needing a code, and then simply sending the fraudulent transaction only. The bank, not knowing that the banco trojan transaction isn't what the user meant, sends a MFA code that works only for the banco trojan's transaction, to the end user. The end user is unaware that there is a second, hidden browser session, and types in the code, which the bancos trojan happily uses.

Bancos trojans pushed South American banks to use MFA solutions early and often. In what is a harbinger for the world, the bancos trojans just adapted to MFA use, and kept on stealing hundreds of millions of dollars. The defense against these particular types of attacks, is for the banks to send all the details of the supposed transaction (e.g., dollar amount, type of transaction, etc.) along with the approval code and not just the approval code. The user needs to understand what they are getting an approval code to do. Neither side should trust the other.

Financial transaction MFA approval messages should always send the critical details of the purported transaction so that the user can see the details before approving.

Malicious MFA Software Modification

If a hacker has admin access to a victim's device or OS, they can do anything the software and hardware is capable of. All MFA solutions require a related software piece (e.g., program, API, interface, etc.) to be able to activate and utilize the MFA option. Even if you don't install and "initialize" your MFA option in software, it was done by someone or enabled by default.

Hackers can maliciously modify the MFA software program or interface (known in Microsoft Windows as Cryptographic Service Providers (CSP) or Key Storage Providers (KSP)) so that the protection provided by the MFA program is weakened or completely disabled. This particular type of attack is not known, by the author of this paper, to have been used publicly, but easily could be. A related attack, used by law enforcement and intelligence agencies, is to compromise a participating target's network node and steal the private or symmetric keys used to encrypt communications so they can read the target's encrypted content.

Malicious MFA Hardware Modifications

Law enforcement and intelligence agencies have modified otherwise trusted MFA hardware equipment so that targets' reliance on that equipment can be more easily compromised. In some instances, the MFA hardware solutions were physically modified to not provide any of the normally-provided protection. In others, predefined encryption keys, known by the authorities, were placed into MFA solutions, which were then directed toward the intended targets. The intended targets then use the MFA solutions thinking they are completely secure, when they either provide little to no protection, or allow compromise at will by the authorities.

Defense: Includes preventing malicious exploitation of end-point, including ensuring that end-users don't get socially engineered into installing something malicious, and making sure the device and software is fully patched.

In one major case, UK and US government spies were documented as having compromised over five billion cell phone SIM card private encryption keys manufactured at the world's largest SIM card maker, Gemalto, in 2011 (https://www.theregister.co.uk/2015/02/19/nsa_and_gchq_hacked_worlds_largest_sim_card_company_to_steal_keys_to_kingdom/) (covered in more detail in the next section below). It likely includes your cell phone's encryption key. Below is a headline from *The Register* in 2015 recalling the theft:

Did NSA, GCHQ steal the secret key in YOUR phone SIM? It's LIKELY

Snowden leaks reveals how spies can crack encryption on calls worldwide

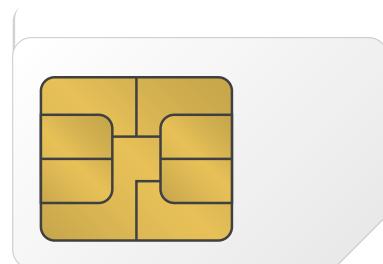
By Iain Thomson in San Francisco 19 Feb 2015 at 22:50

149 

SHARE ▾

SIM Swaps Attacks

Most cell phone and cellular network providers store a subscriber's personal and cell phone unique identifiers in a physical (or increasingly virtual) small memory card known as the Subscriber Identity Module (SIM). It can also act as storage for the cell phone, holding application data such as the user's pictures and contact information. Most look similar to the picture below:

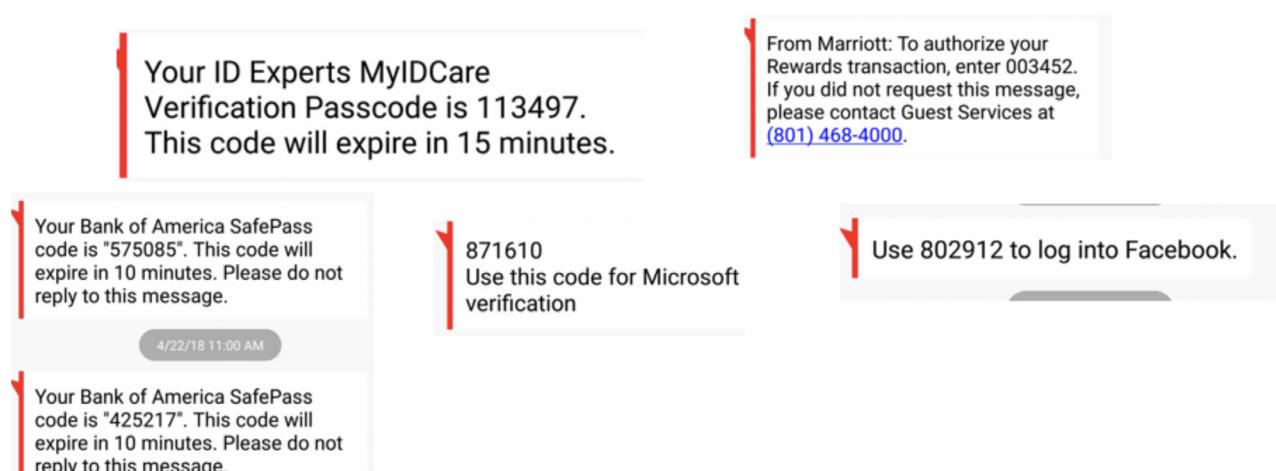


When a user gets a new cell phone, they usually have to move their existing SIM card to the new phone or transfer the information on the SIM to the new phone (known as a SIM swap). The SIM is what "hooks" the cell phone to a particular cellular network provider (e.g., AT&T, Verizon Wireless, etc.), and attaches the subscriber's cell phone number to the cell phone.

For well over a decade, hackers have been obtaining a legitimate subscriber's SIM information, and transferring it to a phone in their possession. This can be done many ways, including the hacker going in-person to a local cell phone store and pretending to be the legitimate subscriber trying to upgrade or replace his/her cell phone. It has also been done remotely thru the cellular network provider's tech support, and employees at cell phone stores have even been bribed to knowingly participate in the malicious SIM swaps. When the SIM swap is made, your cell phone stops working and every call and Short Messaging Service (SMS) sent to your phone is now sent to the hacker's phone.

Malicious SIM swaps usually require that the hacker first gather some private information from the victim being targeted. In order to fool a cellular network provider's tech support or go into a local store, they will usually need the victim's phone number, name, online login name and/or credentials, and home address. He/she usually accomplishes this by getting the needed information from one or more previous phishing attacks against the victim or by obtaining the information from another compromised database.

Rogue SIM swaps have happened millions of times. While this used to mostly just affect a user's voice call service, it is being done more often to maliciously re-route the user's SMS messages. This is a problem because the most popular MFA option across the planet, used by almost every global service provider is SMS-based. Every cell phone user is used to the messages they get from either trusted vendors asking him/her to verify suspected rogue transactions or to type MFA-generated codes into their browsers to complete an MFA-login. Below are some common examples:



So, when a hacker does a malicious SIM swap, those SMS-based MFA messages are sent to the hacker's cell phone instead of yours. The hacker can then compromise your online account using the misrouted SMS-based MFA message. This has happened so much that the U.S. National Institute of Standards & Technology (NIST), which issues federal guidelines for computer security and authentication, said in its recent Digital Identity Guidelines, NIST Special Publication 800-63 (<https://pages.nist.gov/800-63-3/>), that it will not accept SMS-based MFA solutions as legitimate authentication. This is complicated by the fact that, years later, nearly every major vendor uses SMS-based MFA solutions, even those which have stronger, better solutions. SMS-based MFA solutions are either the default, or the backup option, for most of the world's largest vendors using MFA.

Some of the world's largest and most notorious hacks have involved SIM swapping. One cryptocurrency millionaire had over \$24M stolen from his crypto-wallet (<https://www.bankinfosecurity.com/att-sued-over-24m-cryptocurrency-sim-hijack-attacks-a-11365>)

because it relied upon SMS-based MFA. He sued AT&T for \$224M because they transferred his SIM information without authorization. A SIM-based attack was also used in 2018 to compromise Reddit's company network, an attack which led to Reddit's source code and network login credentials being compromised. Here are some related attack links:

- Reddit attack info:
<https://www.wired.com/story/reddit-hacked-thanks-to-woefully-insecure-two-factor-setup>

- Another great SIM swap example:
<https://krebsonsecurity.com/2018/08/florida-man-arrested-in-sim-swap-conspiracy/>

<https://coolwallet.io/smartphone-crypto-hack/>

https://motherboard.vice.com/en_us/article/a3q7mz/hacker-allegedly-stole-millions-bitcoin-sim-swapping

<https://krebsonsecurity.com/2018/08/reddit-breach-highlights-limits-of-sms-based-authentication/>

There is a very real claim that at least SMS-based MFA solutions are better than non-MFA solutions (e.g., passwords). It does take an attacker a lot more work to do a SIM-swap than to just phish someone out of their login name and password, but according to NIST, it's just not trustworthy enough. And if it's not trustworthy enough, why use it? Unfortunately, in today's world, you will likely be forced to until some other better, just-as-pervasive solution, comes along.

Defenses: Don't get socially engineered into handing out your personal information, make sure your cell phone vendor has policies and procedures which prevent malicious SIM swaps, and more importantly, use application-MFA instead of SMS-based MFA whenever possible.

SMS Rogue Recovery

There is an inherent problem in that SMS message origination legitimacy cannot be easily authenticated by the viewer within SMS itself. Anyone can claim to be anyone and send any message. This weakness gives hackers an opportunity to send rogue instructions to potential victims. An example of this type of attack is called SMS Rogue Recovery hacking. For SMS Rogue Recovery hacking to work, the hacker only needs to know the victim's email address (of a service which allows forced SMS recovery) and associated phone number. These are not difficult pieces of information to gain access to.

With this information, the hacker sends a fake SMS recovery message to the victim claiming to be from the victim's email provider. The message falsely indicates that some event is taking place, which will require the victim to type in a sent authorization code back to the SMS-originator to confirm the victim's legitimate ownership and use of the email account (see example rogue SMS recovery message below).

Step 1: Example: Hacker sends rogue SMS recovery message to victim to prepare them for forthcoming legitimate SMS recovery code.

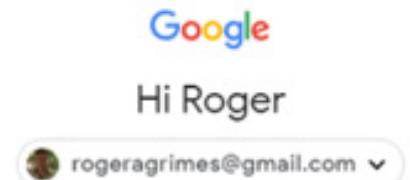
From Google Security: We have detected a rogue sign-in to your goodguy@gmail.com account credentials. In order to determine the legitimate login we're going to send a verification code to your previously registered phone number from another Google support number. Please re-type the sent verification code in response to this message or your account will be permanently locked.

Note that this fake message could be nearly anything. SMS messages themselves are just plaintext with maybe HTTP/HTTPS links or email addresses listed. There is nothing in an SMS message to indicate whether the sender or his/her message is legitimate or not.

After sending the pre-warning message, the hacker purposefully sends the victim's email account into SMS recovery mode by acting to the email provider as if he/she is the legitimate user who has forgotten his/her password (see steps below): The victim's email provider has no way of knowing that the hacker forcing the email account into recovery mode is or isn't the legitimate user.

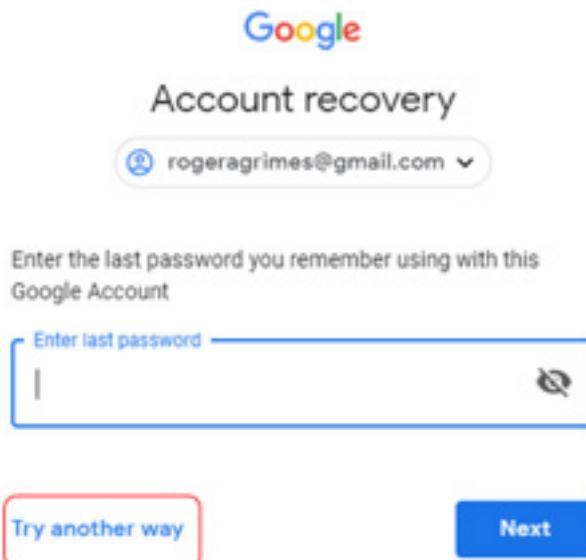
Step 2: Example: Hacker goes to victim's email provider's logon page and acts like victim has forgotten their logon password.

The service will often offer up one or more other login recovery methods.



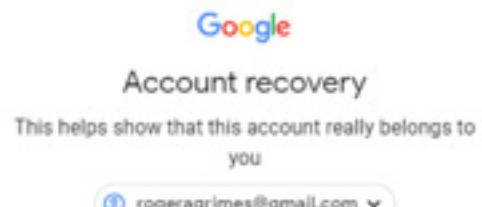
Step 3: Example: Hacker chooses alternate recovery option from victim's email service.

Hacker chooses to send an SMS verification code to the user's predefined phone number.



Step 4: Example: Hacker chooses to send an SMS verification code to the user's predefined phone number.

Legitimate user gets SMS recovery verification code from his/her email vendor.



Example: Step 5: User gets sent legitimate recovery method verification code.

Your Google verification code is
954327

Example: Step 6: User types in legitimate SMS recovery verification code back in response to hacker's original pre-warning SMS message.

From Google Security: We have detected a rogue sign-in to your goodguy@gmail.com account credentials. In order to determine the legitimate login we're going to send a verification code to your previously registered phone number from another Google support number. Please re-type the sent verification code in response to this message or your account will be permanently locked.

954327

Sent

Hacker takes the sent legitimate recovery SMS verification code and types it into email provider's web form, thus taking control over the account.

Defenses to SMS Rogue Recovery Hacking

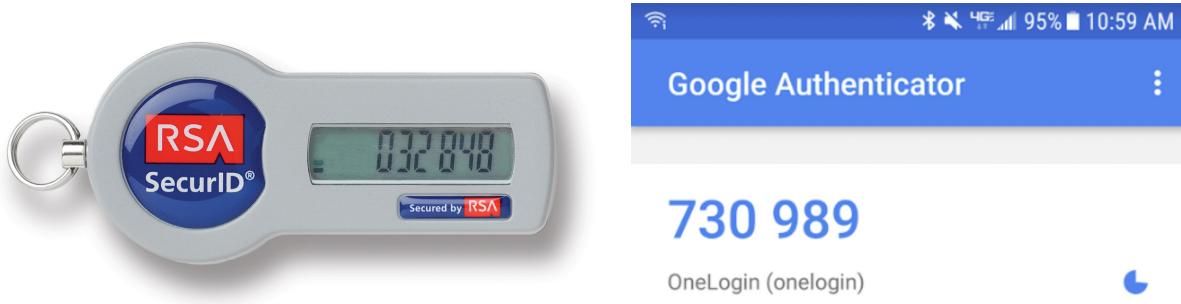
- Be aware of rogue recovery messages
- Recognize when SMS recovery PINs should be typed into browsers, not (usually) back into SMS
- Use MFA when possible
- Try to avoid alternate email-based recovery methods
- Try to avoid SMS-based recovery-based methods
- Try to minimize public posting of phone numbers related to your recovery account methods

Note: Google, which is used in this example, in particular, offers up many different recovery methods beyond SMS verification codes, which the user could decide to only accept as legitimate, thus preventing this particular example of the attack. But there is usually no way to force Google, and other similar email services, only to use a particular recovery method with you if they offer many different methods by default.

Note: I've seen public demonstrations of this method using FIDO U2F and associated recovery methods as well.

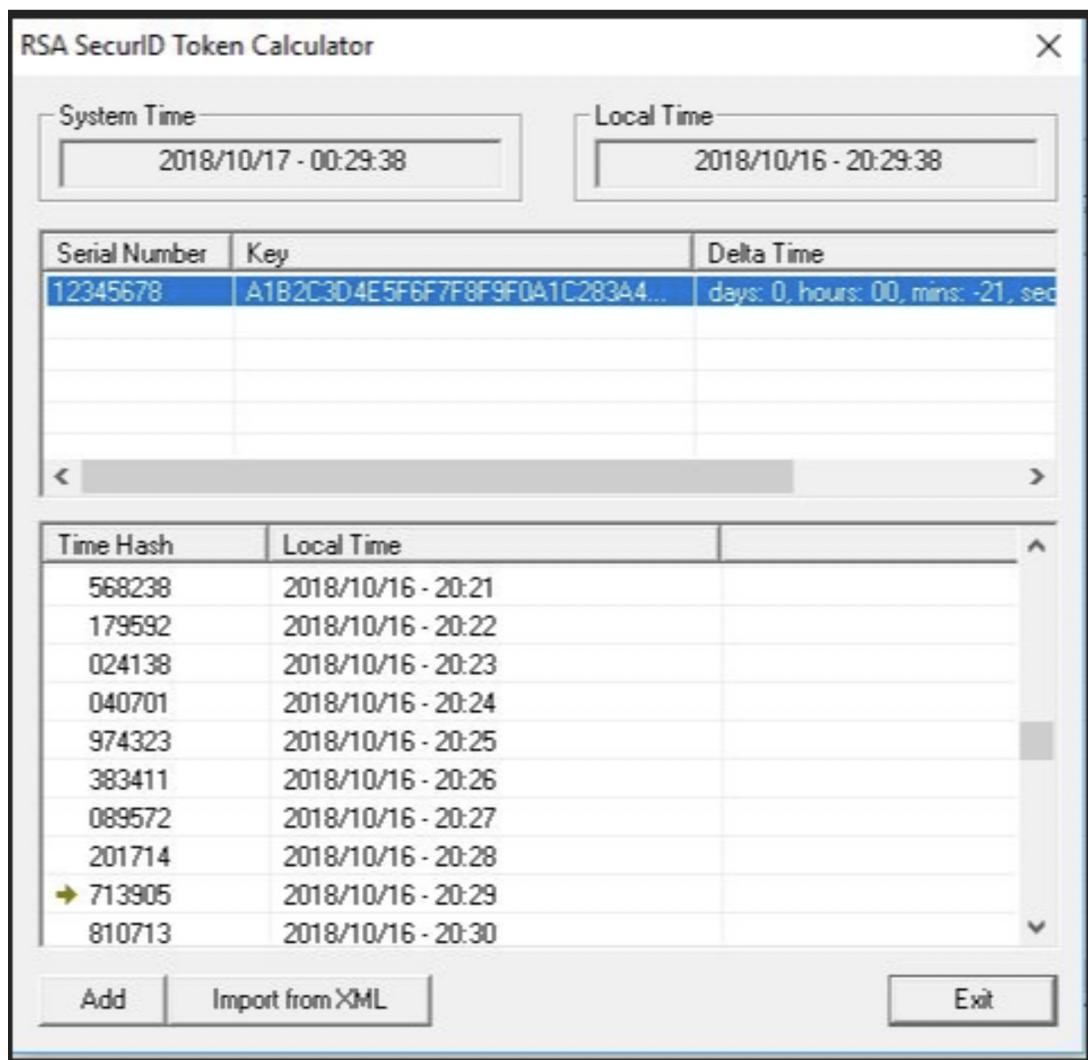
Duplicate Code Generators

Many MFA solutions involve "code generators" which the user is presented with a time-valid code which the user then enters when prompted as part of his/her login experience. The codes often appear as random digits or characters and must be entered in within 30 – 600 seconds of being shown, in order for the MFA solution to work. Popular examples are RSA SecurID™s (hardware) and Google's Google Authentication (software).



These time-valid codes are also known as "one-time-passwords" (OTP) or "time-based-one-time-passwords" (TOTP). In each instance, the user's device or app instance is uniquely identified (ex., Serial number, etc.), and includes a starting (randomly-generated) "seed value". This seed value, the ultimate shared authentication secret for the system, is stored in one or more controlling authentication database, and tied to the related device's unique ID. The seed value database is owned/secured/protected by one or more stakeholders, such as the device's vendor manufacturer and/or corporate user environment. If an attacker accesses the seed value database, he/she can create a duplicate, simulated code generator for any included user. The attack can take the seed value, the device's unique identifier number, current timestamp, and use those values along with the device's known generation algorithm to create the valid OTPs. For example, there are free software hacking tools, like Cain & Abel (<http://www.oxid.it/cain.html>),

which has been available since at least 2001, that have included a RS SecurID emulator. Below is a screenshot of that functionality.



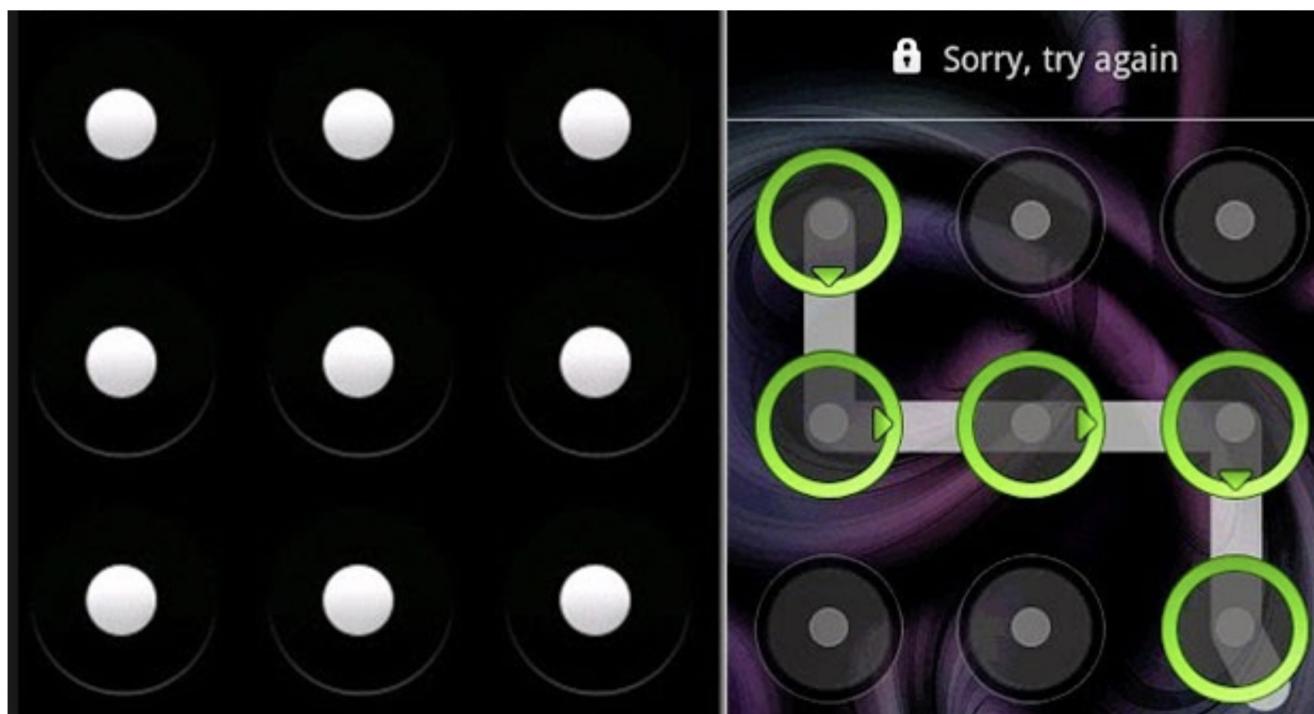
These types of attacks are not theoretical. For example, in 2011, Chinese Advanced Persistent Threat (APT) attackers broke into RSA (<https://gcn.com/articles/2011/06/07/rsa-confirms-to-kens-used-to-hack-lockheed.aspx>) and stole the RS SecurID seed values for several customers, including Lockheed Martin. Then the Chinese APT subsequently broke into Lockheed Martin using these stolen codes and copied very sensitive military secrets.

Defense: Protect and monitor MFA seed value databases as strongly as you do password secrets.

Shoulder Surfing

Some MFA solutions are so bad that you can see the “secret” simply by casually looking at what the user is typing in or doing while they are doing it. The MFA in this case is usually a token, card, device, or other object the user has in their possession along with something the user needs to type in (e.g., PIN, etc.) or accomplish (“connect the dots” “picture password”, etc.).

The first graphic below is an example of the Microsoft Windows Picture Password™ and the second is an example Apple iPhone login.



If the intruder can shoulder surf the secret and access the device, they can login using the victim's MFA solution. This has long been a valid form of attack against passwords and PINs, and hackers have used many different methods in order to be successful, including memorizing what they observe, filming what the user types, or guessing later on by observing a "wear pattern" on the device where the code or action is being accomplished.

I almost didn't include this type of attack because it first requires that the attacker obtain both factors (the user accomplished portion) and the involved device, in order to accomplish. But one growing popular MFA solution is that of "connect-the-dots" or predefined "quick swipe" patterns,

as shown in the graphics above, where the user uses his/her finger to swipe a particular pattern across dots or a chosen picture, in order to login. These types of MFA solutions are the worst MFA solutions possible.

How any user chooses to swipe is highly predictable, even if the hacker doesn't observe it. But if observed, is often readily, easily, even-if-not-a-hacker-simple for others to replicate. There is no other "authentication" option, if you can call it that, where a large percentage of the world can see the swipe from ten feet away and then walk over and recreate a successful login. Do not use these "MFA" solutions.

Defense: Do not use swipe, pattern-based MFA solutions for any critical data.

Skimming Attacks

Skimming is a hacker attack method where the secrets of the MFA solution are stolen during use. In most of these scenarios, the secrets on the MFA solution are very weakly protected or not protected at all. The secrets stored on the MFA card are revealed during a transaction in such a way that they can be recorded while in their unprotected state. Skimming can be done in many different ways, including physically (using a device that records the information directly off of the MFA device) or wirelessly (e.g., RFID or NFC skimming).

Although you may not normally think of skimming attacks as an attack against MFA, they often are. For example, a very common skimming attack is against ATM cashflow machines. The legitimate user must present both the ATM card (i.e., first physical factor) and type in a PIN (the second factor) to access his/her account information and money. The attacker usually places a physical recording device, which mimics the normal ATM keypad area, to record information off of the card's magnetic stripe, at the same time as he/she records, the button pushes electronically or using a hidden "peep hole" camera (see example skimming devices below).



Skimming attacks are also very popularly used at convenience stores and gasoline stations. Most of the time, the banks and stores (and employees) that are part of the skimming attacks are not criminally involved with placing the skimming devices. Many skimming devices can be placed in a short period of time, without the company or employees being aware. Here's a great video (<https://www.youtube.com/watch?v=5b1axnNK-wI>) showing a skimming device being secretly installed in a very populated convenience store in under three seconds while the store clerk's attention is being diverted by an accomplice.

Computer security columnist Brian Krebs has probably researched and published more material on skimming than any other journalist. Check out his information on skimming at:
<https://krebsongsecurity.com/category/all-about-skimmers/>.

Defense: Be aware of skimming and skimming hardware. Try to use vendors that deploy anti-skimming technologies.

Downgrade and Recovery Attacks

Most publicly-available, broadly used MFA solutions (e.g., from Google, Microsoft, etc.) can be required by users as their primary login method. Unfortunately, all those same solutions also have a far less secure backup authentication alternative that you cannot disable. This is because these MFA solutions are more complex and fail more often because of those complex interactions. Physical MFA devices are lost, transferred, and broken. Software-based MFA solutions stop working, lock up, and have to be re-configured due to a host of other reasons beyond their control. The end result is that if these mega-MFA providers didn't have an automated or manual, alternative login or recovery method, the primary MFA solution would be too expensive to provide.

For example, suppose you "require" MFA to login to your Microsoft O365 or Google Gmail accounts and services. A hacker can act as if he/she is failing logging on using the MFA method (maybe he/she tries three times), and the MFA host service will automatically offer to send you an alternative way to confirm your authentication (usually via email, automated voice confirmation, or SMS message). The following graphic shows the backup authentication methods offered by one popular MFA provider.

Account recovery options

If you forget your password or cannot access your account, we will use this information to help you get back in.

Recovery email

roger@██████████



Recovery phone

(████) █████



Below is an example of the resulting recovery security code, which if obtained by a hacker, would allow that account, protected by MFA, to be compromised.

Microsoft account

Security code

Please use the following security code for the Microsoft account ro*****@hotmail.com.

Security code: **0152772**

If you don't recognize the Microsoft account ro*****@hotmail.com, you can [click here](#) to remove your email address from that account.

Thanks,
The Microsoft account team

Recovery Question Attacks

Recovery question attacks are an especially bad extension of the downgrade class of attacks. When signing up for many websites, they will REQUIRE that you create multiple "recovery questions" and/or answers (see example below). You can't create the initial account without agreeing to use and populate the answers to those questions. These recovery questions often include questions such as "Mother's Maiden Name," "Father's Middle Name," "Favorite Teacher," "First Car," and so on.

Your Security Questions

Question:	<input type="text" value="What is the name of the camp you attended as a child?"/>
Answer:	<input type="text" value="*****"/>
Repeat Answer:	<input type="text" value="*****"/>
Question:	<input type="text" value="What is the first name of your favorite Aunt?"/>
Answer:	<input type="text" value="*****"/>
Repeat Answer:	<input type="text" value="*****"/>
Question:	<input type="text" value="What is the zip code of the address where you grew up?"/>
Answer:	<input type="text" value="****"/> <small>▪ Special characters, such as / and -, are not allowed</small>
Repeat Answer:	<input type="text" value="****"/>
Question:	<input type="text" value="What is the name of the street where you grew up?"/>
Answer:	<input type="text" value="****"/>
Repeat Answer:	<input type="text" value="*****"/>

The problem is that recovery questions and answers are often easily guessed and often not correctly remembered by the legitimate users who posted them. There is a great, landmark, Google white paper called Secrets, Lies, and Account Recovery: Lessons from the Use of Personal Knowledge Questions at Google (<http://www.a51.nl/sites/default/files/pdf/43783.pdf>). In it, Google revealed that recovery questions are often fairly easy to guess by complete strangers (e.g., hackers) and are often not correctly remembered by the users who created or answered them. For example:

- Some recovery questions can be guessed on the first try 20% of the time
- 40% of people were unable to successfully recall their own recovery answers
- 6% of answers could be found in a person's social media profile

Note: Google, Microsoft, and other vendors who understand how bad recovery questions are for authentication no longer use them.

If your MFA solution allows less secure alternative authentication methods to be used, your authentication is only as strong as the weakest method.

The solution is never to use them if you can avoid them. If they are required, never answer them correctly. Instead, make up something similar to a long password (complexity is not needed in most scenarios), unique for each recovery answer, never repeating an answer, and store it in a password manager or in "representative" form elsewhere.

An example of the latter is to choose a recovery answer of pizzapizza\$vgad2@M1 (see example below), and if you need to write down the recovery answer, store it as pp\$vgad2@M1, where only you know that pp stands for pizzapizza. That way if an unauthorized person accesses your recovery answer list, they will not learn the correct answer.

Question:	<input type="text" value="What was your high school mascot?"/>
Answer:	<input type="text" value="pizzapizza\$vgad2@M1"/>
Repeat Answer:	<input type="text" value="*****"/>
Question:	<input type="text" value="What is your mother's middle name?"/>
Answer:	<input type="text" value="*****"/>
Repeat Answer:	<input type="text" value="*****"/>
Question:	<input type="text" value="What is your father's birthdate? (mmdd)"/>
Answer:	<input type="text" value="*****"/>
Question:	<input type="text" value="What is the name of your best friend from high school?"/>
Answer:	<input type="text" value="*****"/>
Repeat Answer:	<input type="text" value="*****"/>

Most password managers have “notes fields where you can securely store your questions and related answers for each site needed, and some will actually store and automatically re-populate the correct “answers” when needed.

When an MFA provider allows recovery questions and answers as an alternative authentication method, they are essentially taking you from a somewhat more secure solution to something that is far worse than using a normal single factor authentication method like a login name and password. If your MFA solution allows less secure alternative authentication methods to be used, your authentication is only as strong as the weakest method.

Defenses: Try to prevent downgrade authentication technologies from being allowed by an MFA vendor and never put your real answers in for recovery answers.

Social Engineer Tech Support

A company's tech support engineers are only human, and humans that are trained to be as helpful and solution-providing as they can. Even when a company has specific technical and policy controls to prevent MFA from being socially engineered around, overly helpful technical support employees can bypass them. This is a risk that will not be going away as long as human beings are told to interact with other human beings.

There are many social engineering techniques to get tech support to disable or bypass existing MFA solution protections. These include the “user” (i.e., the hacker) saying:

- He/she lost or damaged his/her existing MFA solution
- He/she doesn't remember his/her existing PIN or password
- Boss is very mad at him/her for not getting something done that requires bypassing MFA solution
- He/she is the boss and he/she needs immediate bypass to do something critical

Social engineering hackers often use artificially created “stressor events” like a critical task or financial issue that they are involved in, which needs to be solved immediately. One of my favorite social engineering demonstration hacks involves a female using social engineering to take over a user's cellular phone account by claiming to be a mother with a crying baby that is in trouble with her husband for not doing something.

Check it out: <https://www.youtube.com/watch?v=lc7scxvKQOo>. The very end of the video is NSFW (Not Safe For Work), as it includes a cuss word from the guy whose account was taken over.

Defense: Make sure your MFA vendor understands the risk of social engineering against their tech support, and that they use tools, policies, and procedures, to prevent malicious social engineering.

Subject Hijack

Every MFA solution is tied to a unique identity, such as the user's or device's login name, email account, user principal name (UPN), or some other unique identifier in the related namespace. In some MFA scenarios, if you can modify the MFA user's identifier in the namespace, you can take over or bypass the related MFA. This is especially true of MFA solutions which do not look to see if the offered, successful MFA solution, actually ties to the identity being used (i.e., the MFA identifier is not stored in the namespace or is not verified as belonging to a particular user).

Here's a great example of this type of attack. It involves Microsoft Active Directory, the user's UPN, and smartcards. This attack has been possible for decades, although as far as this author knows, it

has not been knowingly exploited “in-the-wild” against a real-world target. It has been reported to Microsoft, verified, and was given the “as-designed” resolution. It’s also a potential insider evaluation of privilege attack. Here’s how it works:

Microsoft Smartcard Identity Hijack Attack

For this particular attack scenario, we are going to assume a standard Active Directory forest environment which requires smartcards for admins to authenticate. Bad guy needs two critical requirements:

- Any user smartcard and its authenticating PIN trusted by the victim’s Active Directory forest
- Ability to change intended victim’s UPN from one value to another

Note: That the user smartcard as required in the first bullet point can be any trusted user smartcard issued by any certification authority (CA) trusted by the forest. The smartcard could even contain a user and a UPN that does not actually exist in the targeted forest, although for this demonstration, the user will be an existing, valid, trusted user.

The key to this hack is that the attacker simply updates the victim’s UPN with his/her UPN. In Active Directory, many admins can update user account information. By default, according to Microsoft, the following groups and individual permissions can update UPNs: Administrators (on domain controllers), Domain Admins, Enterprise Admins, Schema Admins, and any user with Write Public Information over the involved user accounts.

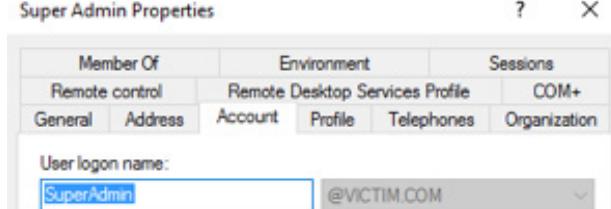
For this hacking scenario we are going to assume that a lower-privileged-level user, named HelpDesk, has the ability to update a highly-privileged admin, called Super Admins, UPN. HelpDesk can use this one change to completely take over the Super Admin’s security account permissions, group memberships, and privileges. Here’s a summary of the hack steps:

1. Low-privileged HelpDesk admin switches UPNs with Super Admin.
2. HelpDesk admin logs in using his/her own HelpDesk smartcard and PIN.
3. Viola! HelpDesk admin becomes Super Admin, including all group memberships.
4. HelpDesk performs malicious actions.
5. System tracks all actions as Super Admin.
6. When HelpDesk is finished, he/she logs out, and switches UPNs back. No one knows the difference.

Does your log management system track and alert on UPN updates? You can watch a recorded demo of this attack at: <https://youtu.be/OLQ3IAMuokI>.

All unique subject/identity attributes used in the authentication process, such as UPN, need to be as protected and monitored as the other authentication secrets like password hashes.

First, let’s verify Super Admin’s UPN (which is represented as User Logon Name in Active Directory) is SuperAdmin:



Let's verify that SuperAdmin belongs to all sorts of elevated groups (see below).

Verifying that SuperAdmin belongs to elevated groups.

The screenshot shows the 'Super Admin Properties' dialog box with the 'Member Of' tab selected. The table lists various groups and their corresponding Active Directory paths:

Name	Active Directory Domain Services Folder
Administrators	VICTIM.COM/Builtin
Domain Admins	VICTIM.COM/Users
Domain Users	VICTIM.COM/Users
Enterprise Admins	VICTIM.COM/Users
Schema Admins	VICTIM.COM/Users
Server Operators	VICTIM.COM/Builtin

Next, we verify that Super Admin's UPN is tied to the Super Admin's smartcard, including the unique digital certificate thumbprint related to Super Admin's smart card.

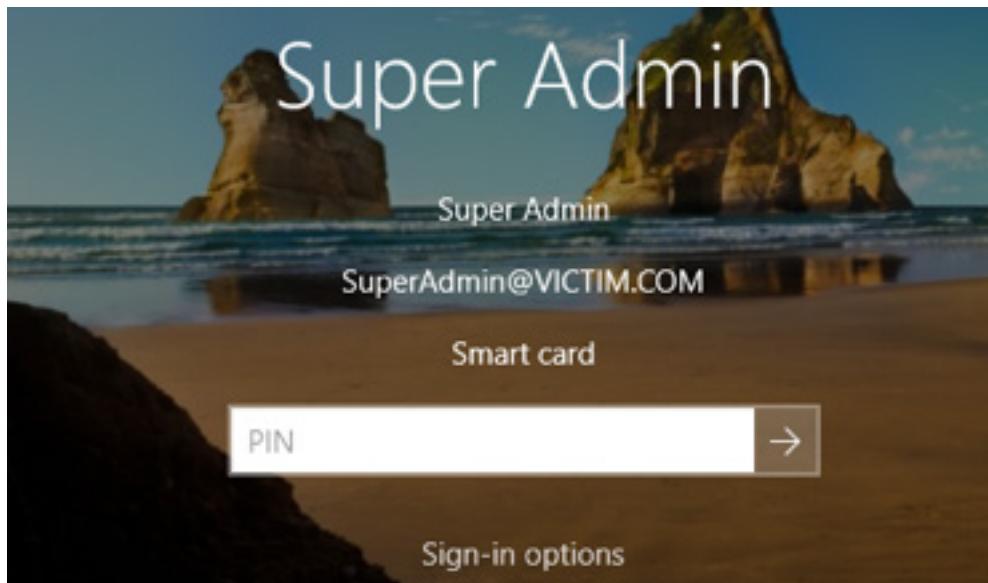
Verifying Super Admin's smartcard's UPN and unique digital certificate thumbprint.

The screenshot shows the 'Certificate' dialog box with the 'Details' tab selected. It displays the following details:

Field	Value
Authority Key Identifier	KeyID=8a 16 c7 63 26 1c 8d d...
CRL Distribution Points	[1]CRL Distribution Point: Distr...
Authority Information Access	[1]Authority Info Access: Acc...
Subject Alternative Name	Other Name-Principal Name=5...
Key Usage	Digital Signature, Key Enipher...
Thumbprint algorithm	sha1
Thumbprint	55 fa d6 56 67 c3 3e 23 d1 22 ...

Below the table, the 'Other Name' field contains the value 'Principal Name=SuperAdmin@VICTIM.COM'. At the bottom, there are buttons for 'Edit Properties...', 'Copy to File...', and 'OK'.

Logging on as Super Admin using Super Admin's smartcard and PIN.



Verifying that Super Admin has successfully logged on using Super Admin's smartcard and PIN.

Running Whoami command to verify that Super Admin is logged in as SuperAdmin.

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\windows\system32>whoami
victim\superadmin

C:\windows\system32>
```

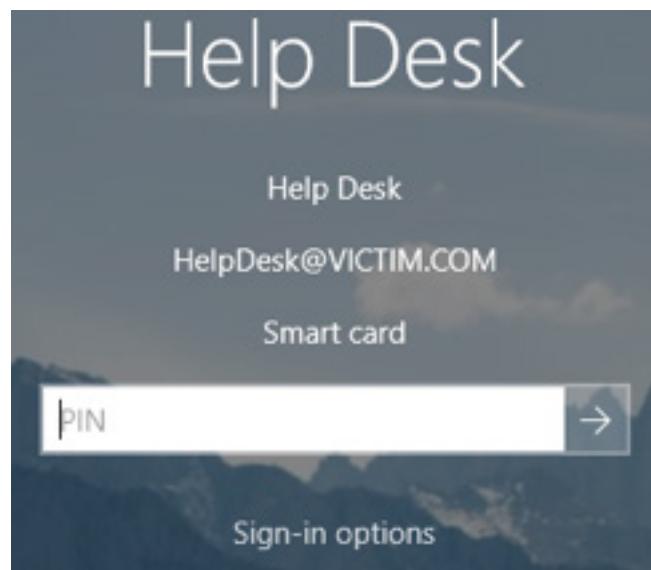
Verifying SuperAdmin has elevated group memberships.

Using Whoami /groups to verify the elevated groups Super Admin belongs to.

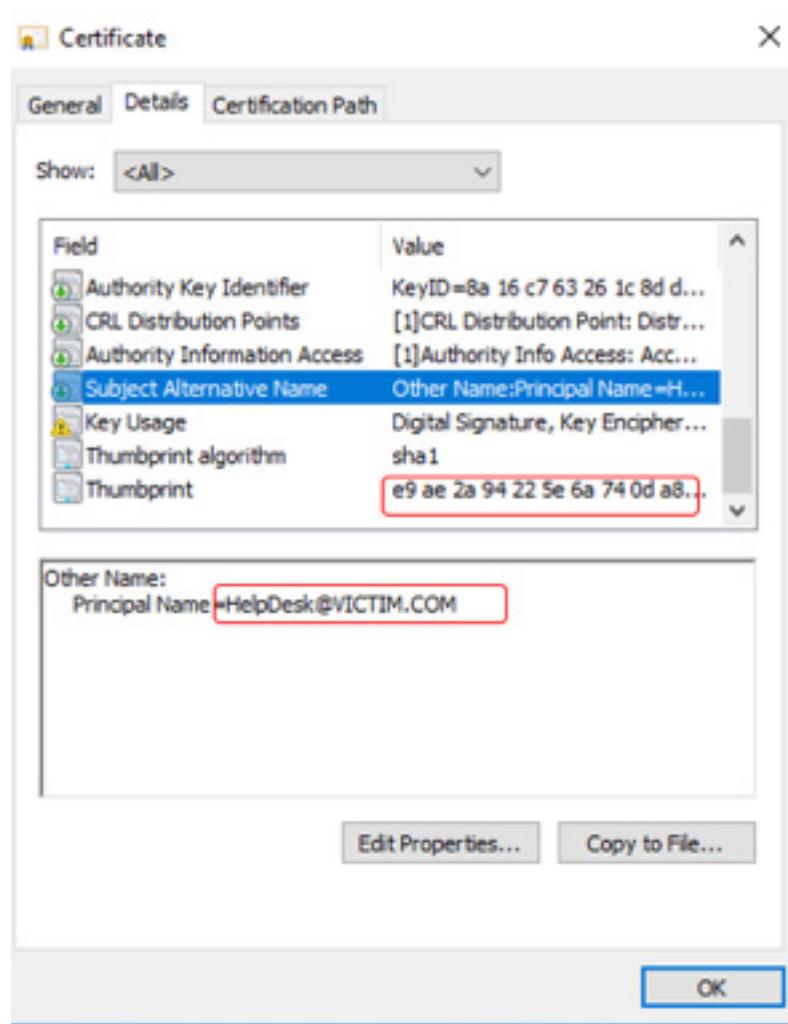
```
Administrator: Command Prompt
-----
Everyone                                Well-known group S-1-1-0
BUILTIN\Administrators                  Alias          S-1-5-32-544
BUILTIN\Users                            Alias          S-1-5-32-545
NT AUTHORITY\INTERACTIVE                 Well-known group S-1-5-4
CONSOLE LOGON                            Well-known group S-1-2-1
BUILTIN\Authenticated Users              Well-known group S-1-5-11
NT AUTHORITY\This Organization          Well-known group S-1-5-15
LOCAL                                    Well-known group S-1-2-0
VICTIM\Domain Admins                    Group         S-1-5-21-98619
VICTIM\Enterprise Admins                Group         S-1-5-21-98619
VICTIM\Schema Admins                   Group         S-1-5-21-98619
Authentication authority asserted identity Well-known group S-1-18-1
```

Next, the example will show the less privileged HelpDesk user logging on using their own smartcard and PIN (before the hack is accomplished to show current state).

HelpDesk user logging on using Help Desk smartcard and PIN.



HelpDesk's smartcard information showing helpdesk@victim.com UPN and HelpDesk's unique digital certificate thumbprint.



Using Whoami to show HelpDesk logged on to victim.com domain.

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\windows\system32>whoami
victim\helpdesk

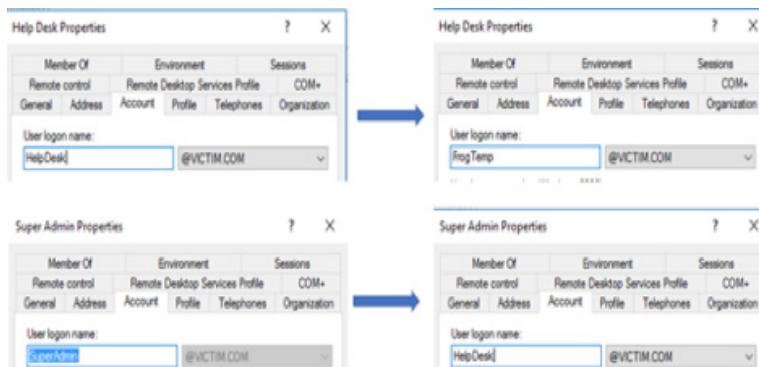
C:\windows\system32>
```

Using Whoami /groups to show the lower privileged groups that the HelpDesk admin belongs to.

Group Name	Type	SID
Everyone	Well-known group	S-1-1-0
oup		
BUILTIN\Administrators	Alias	S-1-5-32-544
oup, Group owner		
BUILTIN\Users	Alias	S-1-5-32-545
oup		
NT AUTHORITY\INTERACTIVE	Well-known group	S-1-5-4
oup		
CONSOLE LOGON	Well-known group	S-1-2-1
oup		
NT AUTHORITY\Authenticated Users	Well-known group	S-1-5-11
oup		
NT AUTHORITY\This Organization	Well-known group	S-1-5-15
oup		
LOCAL	Well-known group	S-1-2-0
oup		
Authentication authority asserted identity	Well-known group	S-1-18-1
oup		
NT AUTHORITY\This Organization Certificate	Well-known group	S-1-5-65-1
oup		
Mandatory Label\High Mandatory Level	Label	S-1-16-12288

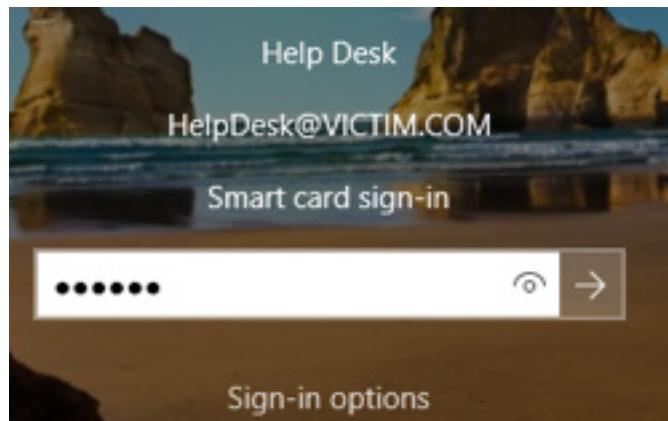
Now the lower-privileged HelpDesk user is going to swap it's UPN with the SuperAdmin user using Active Directory Users and Computers. To do this, the HelpDesk user must first change to their UPN with any other value because Active Directory will not let two accounts share the same UPN at the same time. Then the HelpDesk user updates the SuperAdmin's UPN to read helpdesk@victim.com, to be the same as the UPN it holds on its own smartcard.

HelpDesk changing its UPN to anything else and then updating the SuperAdmin UPN to the UPN as stored and validated on the HelpDesk smartcard.



Next, the HelpDesk User logs out, waits a few minutes for Active Directory replication to occur, and then logs backs in.

HelpDesk user logs back in using its valid smartcard and PIN.



Details of HelpDesk smartcard used to log back in to verify that only HelpDesk smartcard and PIN is being used.

A screenshot of the Windows Certificate dialog box. The "Details" tab is selected. A table shows certificate fields like Authority Key Identifier, CRL Distribution Points, Subject Alternative Name (highlighted in blue), Key Usage, Thumbprint algorithm, and Thumbprint. Below the table, the "Other Name" field is expanded to show "Principal Name=HelpDesk@VICTIM.COM".

Although the HelpDesk user logged on using their associated HelpDesk smartcard and PIN, Active Directory has now associated them with the SuperAdmin user because SuperAdmin now has the helpdesk@victim.com UPN.

Using Whoami to verify that HelpDesk user is now seen as SuperAdmin by Active Directory.

A screenshot of an Administrator Command Prompt window. The command "whoami" is run, showing the output: "C:\windows\system32>whoami victim\superadmin C:\windows\system32>".

Using Whoami /groups to show that the logged in HelpDesk user now has the group memberships of SuperAdmin.

```
Administrator: Command Prompt
C:\windows\system32>whoami /groups

GROUP INFORMATION
-----
Group Name          Type
-----
Everyone           Well-known group
BUILTIN\Administrators   Alias
BUILTIN\Users         Alias
NT AUTHORITY\INTERACTIVE  Well-known group
BUILTIN\SYSTEM          Well-known group
CONSOLE LOGON          Well-known group
NT AUTHORITY\Authenticated Users  Well-known group
BUILTIN\SYSTEM          Well-known group
NT AUTHORITY\This Organization  Well-known group
BUILTIN\SYSTEM          Well-known group
LOCAL                Well-known group
VICTIM\Domain Admins    Group
VICTIM\Enterprise Admins  Group
VICTIM\Schema Admins    Group
```

At this point, the HelpDesk user can do anything SuperAdmin could currently do, and Microsoft Windows and Active Directory would track all Windows Event Log events as if they were occurring from SuperAdmin, not HelpDesk user. After committing other unauthorized actions, such as confirming he/she can always elevate his/her security credentials, the HelpDesk user could swap back the UPNs, and unless UPN updates are being logged and the importance of those particular actions were noticed, it would be difficult to easily see what actually happened.

Note: There are some logon events registered at the domain controllers which would include each supplied smartcard's thumbprint, which could be reviewed to reveal that the HelpDesk user's smartcard was now somehow associated with the SuperAdmin account, but forensics investigators would have to be aware of and seeking to validate this particular hack scenario. It would not be easy to discover when trying to discuss what really happened.

To be clear, this is not really a hack or bug. This is an "as-designed" outcome in which the way Active Directory-integrated smartcards work. Microsoft will not likely be "fixing" this, although there are ways to minimize this sort of hacking hijinks. Ultimately, the valid, trusted smartcard is saying the successfully authenticated user of it is helpdesk@victim.com. Active Directory now understands helpdesk@victim.com to belong to SuperAdmin.

As covered earlier on the active of validating an authentication credential scam, and often is, separate from the resulting authorization and access control processes. In this particular hacking scenario, the smartcard authentication process (i.e. that the user is presenting a valid, trusted

smartcard with a correct UPN and knows the associated PIN) and the access control and authorization process (where the “validated” user is handed their group memberships and privileges in an access control token” are, to Active Directory, almost completely separate events. Once a valid, trusted smartcard and the correct PIN is typed in, there is no way (in the most common smartcard scenarios) for Active Directory to know the validated UPN being supplied shouldn’t be mapped to the user account that contains it.

The general protection against these sorts of MFA abuses is to realize that any time you use an attribute, such as subject name, as part of an authentication solution, the involved attribute needs to be protected and monitored as if it were an authentication secret. Most administrators are taught to secure other authentication secrets, such as password hashes, as if they were the keys to the kingdom; and they are. But most admins are not taught to protect and monitor the other involved authentication attributes, even though their malicious modification can have similar security impacts.

Re-Created Biometrics

Biometric attributes include (supposedly) universally unique physical traits, which do not match other subjects. These include: fingerprints, retina scans, finger-hand geometry, DNA, voice, smell, face, vein patterns, body shapes, body part (i.e., ears) shapes, and even action-based biomechanics such as signature, mouse clicks, and keyboard typing attributes (such as how long it takes you to get from the “e”-key to the “s”-key when typing “es”), etc. Ignoring for the moment that no biometric attribute has ever been proven to be globally unique, there are other issues. Whenever we’ve been told that a particular biometric identifier is unforgeable, someone has readily forged them, often within a day, taking less than \$100 to do so, and posted on YouTube for the world to see. Here’s a great example of someone breaking Apple’s iPhone’s facial recognition (<https://www.youtube.com/watch?v=sYSQBleC4fs>). Fingerprint scanners have been fooled by gelatin and Silly Putty™ re-creations.

Successfully forging biometric identities is made easier because even though the biometric identities may be (almost) globally unique, biometric reading devices must be “de-tuned” to be less sensitive at detection changes, or else they will have too many false-negatives (i.e., declining a legitimate holder of the biometric identity).

For example, even though your fingerprints may be globally unique, its recording and subsequent measurement cannot be done at the finest possible distinction resolution. This is because our fingerprints have many “micro-changes” every day, due to small cuts, abrasions, and adhesiveness to the things we touch, or even sweat. If the fingerprint reader was tuned too sensitive, there would be far too many false-negative denials for the admins and users to accept. So, each biometric device is de-tuned, made far less sensitive than it could be, which makes them more susceptible to near likenesses of those same biometric identities.

As one issue, the de-tuning causes different biometric attributes, which are truly unique, to be able to be false-positively accepted for a particular unique identity. For example, I know of one company with just over 500 employees, which uses fingerprint analysis for physical authentication. Three employee’s fingerprints have already been found to have “matched” other previously registered employees. These three employees do not share the same fingerprints, not even close. But the de-tuned fingerprint reader thinks they are identical. The solution to these types of issues is to make the biometric readers more sensitive to changes, but to do so will cause far more false-negatives, which is a problem most environments are not willing to operate with. All biometric attributes can be replicated or stolen. And once done, the biometric attribute cannot be trusted for future authentication. Over decades, every supposedly unforgeable biometric

identity has proven to be completely reliable as having come from the legitimate person it is supposedly from.

Defenses: Recognize the inherent fallibility of biometrics, and if used, make sure they are always used with a second, non-biometric factor.

Stolen Biometrics

All biometric attributes, once recorded, are stored either locally on the computer where they were taken or saved to a network-reachable database. Oftentimes these biometric identities are stored on multiple databases around the country or world. Hackers are used to stealing databases. Once a biometric attribute has been stolen (or successfully re-created), it can no longer be trusted in any future authentication scenario.

For example, in a single 2015 attack, 5.6 million people's fingerprints were stolen from the U.S.

Office of Personnel Management

These stolen fingerprints included anyone who had ever applied for a U.S. government security clearance (https://en.wikipedia.org/wiki/Office_of_Personnel_Management_data_breach). It included the author of this paper's fingerprints. It included the author's wife's fingerprints, fingerprints which were recorded by in the early 1980s, when she worked in a shipyard. It included the fingerprints of our top-secret spies. In one attack, nearly six million people's fingerprints can no longer be trusted.

Today, our world is full of facial recognition scanners. The U.S. FBI has over 117 million faces (<https://www.digitaltrends.com/cool-tech/feds-facial-recognition-database-over-100-million/>). The UK and Chinese facial recognition systems are thought to contain tens of millions of faces as well. How many biometric attribute database compromises don't we know about?

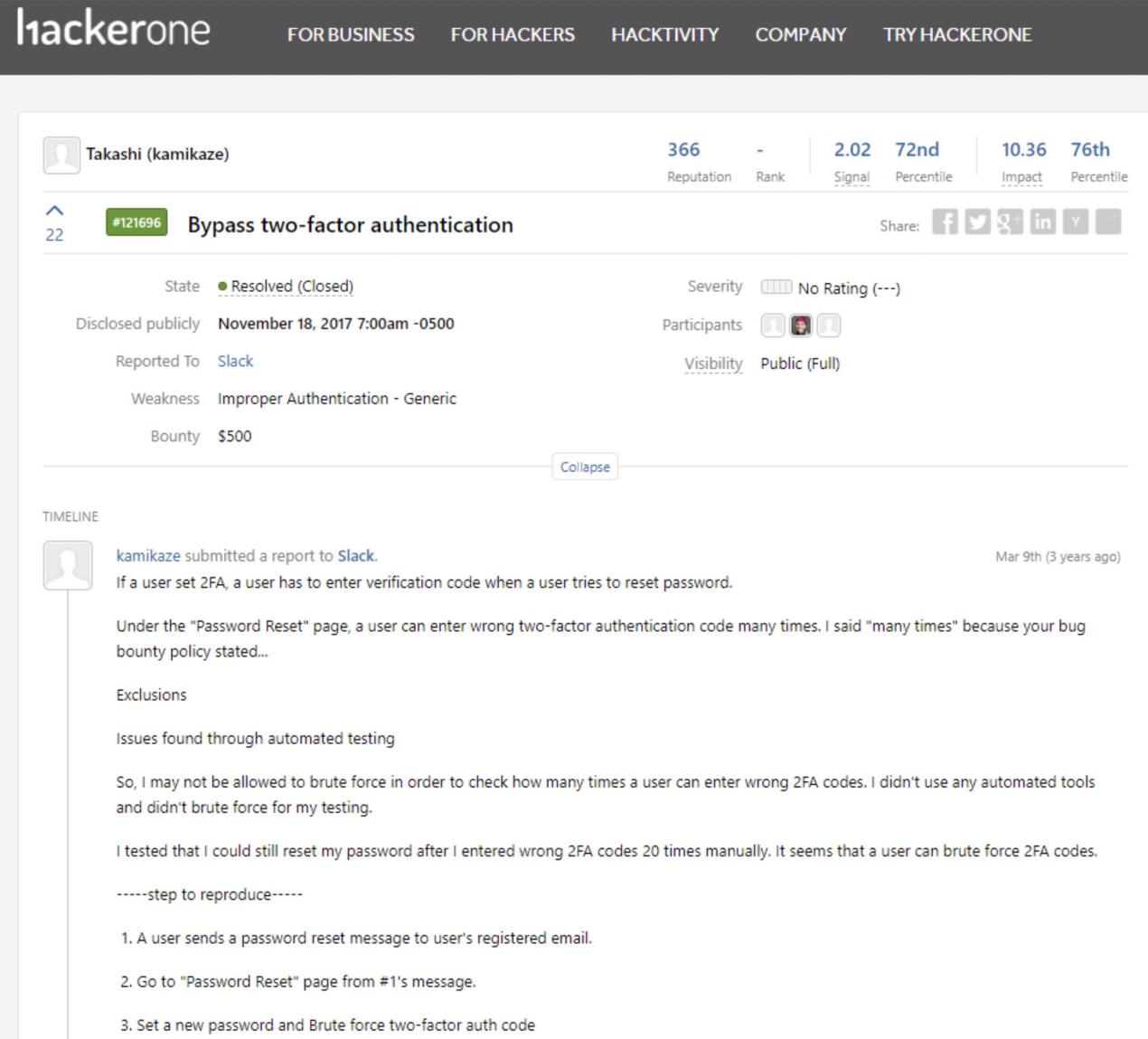
The problem with stolen biometric attributes is that once they are compromised, you can't change them. We can change our password, PIN, smartcard, or MFA token, but we can't (easily) change our body. This is why biometrics alone, without another non-biometric factor, must be part of any biometric-related MFA solution.

Defense: Protect and monitor biometric attribute databases as if they were critical password secrets.

Brute Force Attacks

Many MFA solutions have "something-you-know" factors, such as passwords and PINs. Oftentimes if a hacker gets a hold of the MFA device that is paired with a "Something-you-know" factor, he/she can guess at the "something-you-know" part until he/she breaks it. In the traditional password world, we are used to being limited to a small number of incorrect guesses before the account is locked out (known as account lockout) and some MFA solutions will artificially slow repeated incorrect guesses (called rate limiting). But often times, a new MFA solution is presented without either defense being included, so that the hacker can guess until he/she is right. It happens all the time.

Here's a bug report from November 2017 regarding Slack's MFA solution:
<https://hackerone.com/reports/121696>.



Takashi (kamikaze) 366 - 2.02 72nd 10.36 76th
FOR BUSINESS FOR HACKERS HACKTIVITY COMPANY TRY HACKERONE
Reputation Rank Signal Percentile Impact Percentile

#121696 Bypass two-factor authentication Share:

State: Resolved (Closed) Severity: No Rating (---)
Disclosed publicly: November 18, 2017 7:00am -0500 Participants:
Reported To: Slack Visibility: Public (Full)
Weakness: Improper Authentication - Generic
Bounty: \$500

TIMELINE

kamikaze submitted a report to Slack. Mar 9th (3 years ago)
If a user set 2FA, a user has to enter verification code when a user tries to reset password.

Under the "Password Reset" page, a user can enter wrong two-factor authentication code many times. I said "many times" because your bug bounty policy stated...

Exclusions

Issues found through automated testing

So, I may not be allowed to brute force in order to check how many times a user can enter wrong 2FA codes. I didn't use any automated tools and didn't brute force for my testing.

I tested that I could still reset my password after I entered wrong 2FA codes 20 times manually. It seems that a user can brute force 2FA codes.

-----step to reproduce-----

1. A user sends a password reset message to user's registered email.
2. Go to "Password Reset" page from #1's message.
3. Set a new password and Brute force two-factor auth code

Another brute force attack against another MFA solution in October 2018 is:
<https://www.cloudfoundry.org/blog/cve-2018-11082/>. They are not unusual in new MFA systems, where they remain until someone reports them as a bug.

Defense: All MFA options should include rate-throttling or account lockout features.

Buggy MFA

Human beings cannot write flawless code. MFA solutions always involve software, and that software will contain coding bugs. One or more of those bugs may contain errors so significant that they become security bugs. Some security bugs are so severe that they will allow a complete bypass of the MFA solution, and some are so bad that hundreds of millions of individual instances are compromised.

For example, here is an MFA bypass due to a coding bug involving Uber from January 2018: <https://www.zdnet.com/article/uber-security-flaw-two-factor-login-bypass/>. It took many weeks for Uber to even accept the submitted bypass bug as an actual issue that needed to be fixed. Here are some more MFA bypass bugs:

- <https://www.youtube.com/watch?v=eFD89QrcRg8>
- https://www.youtube.com/watch?v=lPrhlmqN_7E
- <https://hackerone.com/reports/264090>
- <https://blog.elcomsoft.com/2017/11/breaking-apple-icloud-reset-password-and-bypass-two-factor-authentication/>

ROCA Vulnerability

Perhaps the most infamous, widespread, MFA bypass bug (so far) is the 2017 ROCA vulnerability (https://en.wikipedia.org/wiki/ROCA_vulnerability). In this case, every 2048-bit RSA private/public key pair generated with an Infineon Technologies RSALib cipher library, would contain this vulnerability. Over a hundred million smartcards and other related crypto devices, such as Trusted Platform Module (TPM) chips. The bug was so severe that if an attacker could obtain the public key of the private/public key pair (which normally everyone in the world can have without creating a security issue), then he/she could readily regenerate the related private key. This had the impact of making over a hundred million smartcards and other devices essentially using very transparent encryption.

Defense: Use MFA solutions which use coding methods and developers which incorporate security and bug minimization from the ground up. There are several programming methods which do this, including the Security Development Lifecycle (<https://www.microsoft.com/en-us/sdl>), used by Microsoft and other leading vendors. Vendors incorporating SDL-like security design will still create products which have security vulnerabilities, but tend to have less of them than software that does not.

Other Physical Attacks

There is a saying in the computer security world that a computer device under physical control of an attacker is never secure. That saying is never so true as when applied to the world of cryptography and MFA solutions. The secret encryption keys of an MFA solution must be stored somewhere, and must be shown in their decrypted, plaintext-state, in order to be used successfully in authentication.

Electron Microscope Attack

Attackers have learned that secret encryption keys stored on chips or devices can be discovered by looking at the molecular level. In one case, a computer scientist used an Electron Microscope (<https://gcn.com/articles/2010/02/02/black-hat-chip-crack-020210.aspx>) to find the secret encryption keys on a specialized encryption chip designed purely to securely store the encryption keys. There is no MFA hardware solution that will not be vulnerable to this type of attack.

Cold Boot Attacks

Cold boot attacks are a class of attack where the stored secret keys can be captured, analyzed, and exposed by a range of different actions, some of them natural and usual events, and others unnatural and usual. Many of these sorts of attacks rely on inherent properties of other computing devices or chips involved in the protection process.

For example, if computer memory contains a copy of the unencrypted secret key used to decrypt data, the data in the memory can be “frozen”, then taken to another computer, and compromised. In one of the best-known examples, researchers simply sprayed regular, very commonly used, computer memory in a running computer with canned air, which you could buy at any office supply store, until frost appeared on the memory chips.

They then removed those frozen memory chips to another computer, which contained specialized forensic software. Using the new computer with the other memory chips, they were able to find and extract the encryption secret key. See the following links for more details on cold boot attacks:

- https://en.wikipedia.org/wiki/Cold_boot_attack
- <https://www.zdnet.com/article/cryogenically-frozen-ram-bypasses-all-disk-encryption-methods/>
- <https://www.wired.com/story/cold-boot-break-pc-encryption/>

Defense: Try to prevent MFA solutions from falling into the hands of hackers and use MFA solutions that require two different types of factors, so that if they physically gain one, they don't automatically have the other.

The previous list of over 15 attacks that have or could be successful against MFA is not an inclusive list. It did not cover many well-known methods such as side-channel attacks and electromagnetic radiation (EMR) eavesdropping. But it is a great representative summary of the type of attacks that users and admins should be aware of when they choose and implement MFA solutions.

This is not to say that some MFA solutions are not more secure than others. The trick is to pick the MFA solution that has the correct amount of protection for the majority of your scenarios. There is no “right answer” for all solutions and scenarios. Just pick the one that works best for you and your needs. A great place to start to find your best MFA solution is by visiting the following links:

Quest to Replace Passwords white paper

<https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/QuestToReplacePasswords.pdf>

Joseph Bonneau

<http://jbonneau.com/>

NIST Digital Identity Guides

<https://pages.nist.gov/800-63-3/>

Check to see if a website supports 2FA

<https://twofactorauth.org/>

Summarizing Defenses Against MFA Attacks

Each previously discussed MFA attack type included one or more summarized defenses at the end of the section. This section is just a re-summarization of those methods, to be used for quick referral.

Social Defenses

- Realize nothing, including any MFA solution, is unhackable
- Include MFA hacking awareness into your security awareness training
- Share this white paper with co-workers and management
- Don't get tricked into clicking on rogue links
- Block rogue links as much as possible
- Make sure a URL is legitimate

Technical Defenses

- Enable REQUIRED MFA whenever possible
- Don't use SMS-based MFA whenever possible
- Use "1:1" MFA solutions, which require client-side to be pre-registered with the server
- Use/require two-way, mutual, authentication whenever possible
- Ex. FIDO U2F's Channel or Token Binding
- Does your MFA solution specifically fight session token theft and/or malicious replays (i.e., replay resistant)
- Can your MFA vendor's support help be socially engineered?
- Make sure MFA vendors use secure development lifecycle (SDL) in their programming
- Make sure MFA has "bad attempt throttling" or "account lockout" enabled
- Spread factors across different "channels" or "bands" (in-band/out-band)
- Protect and audit identity attributes used by MFA for unique identification of MFA logins
- Don't answer password reset questions using honest answers
- Encourage and use sites and services to use dynamic authentication, where additional factors are requested for higher risk circumstances
- Understand the risks of "shared secret" systems
- For transaction-based authentication, need to send user all critical details out-of-band before confirmation is transmitted/required

Conclusion

The key takeaways from this paper include the following:

- MFA isn't unhackable.
- MFA does not prevent phishing or social engineering from being successful.
- MFA is good. Everyone should use it when they can, but it isn't unbreakable.
- If you use or consider going to MFA, security awareness training has still got to be a big part of your overall security defense.

Additional Resources



Phishing Security Test

Find out what percentage of your employees are Phish-prone with your free Phishing Security Test



Automated Security Awareness Program

Create a customized Security Awareness Program for your organization



Free Phish Alert Button

Your employees now have a safe way to report phishing attacks with one click



Free Email Exposure Check

Find out which of your users emails are exposed before the bad guys do



Free Domain Spoof Test

Find out if hackers can spoof an email address of your own domain



About KnowBe4

KnowBe4 is the world's largest integrated Security Awareness Training and Simulated Phishing platform. Realizing that the human element of security was being seriously neglected, KnowBe4 was created to help organizations manage the problem of social engineering through a comprehensive new-school awareness training approach.

This method integrates baseline testing using real-world mock attacks, engaging interactive training, continuous assessment through simulated phishing, and vishing attacks and enterprise-strength reporting, to build a more resilient organization with security top of mind.

Tens of thousands of organizations worldwide use KnowBe4's platform across all industries, including highly regulated fields such as finance, healthcare, energy, government and insurance to mobilize their end users as a last line of defense and enable them to make better security decisions.

For more information, please visit www.KnowBe4.com