How To Count To Two: What “Two Factor Authentication” Misses

Eugene Shablygin, Sergey Bratus

February 9, 2015

... Double, Double Toil and Trouble

The Bard

Abstract

Getting a “second factor” for authenticating users in addition to a username-and-password is marketed as the go-to protection against attacks on corporate IT systems. However, this “rule-of-thumb” recommendation represents a missed opportunity for improving authentication security non-incrementally and also misses an important aspect of how attackers operate. In this paper we diagnose the problem–the inherent weakness of using *shared and predictable* username spaces. We also recommend a solution: eliminating sharing by *starting* an authentication transaction by applying the non-guessable stronger factor first instead of relegating it to the “second” status.

A classic joke asserts that there are two really hard problems in computer science: naming things and off-by-one errors. Neither of these problems seems to directly relate to corporate IT, but we argue that at least the first of them is actually central to enterprise IT (in)security and has been overlooked only because of mental inertia fostered by legacy authentication schemes.

In a nutshell, legacy authentication starts with presenting a name from a relatively small and easily guessable namespace: a *username*. We use the term username to mean a token of identity intended to be readable to humans, entered by them, and, generally speaking, coming from the natural language, possibly with the addition of a few simple morphology rules.[1] Needless to say, if these rules are conventional for system administrators, they can also be second-guessed by attackers.

A username, presented before any actual credentials, is typically related to the user's actual name by a very simple rule, or describes his role on the system, with even less variety.[2] This easily guessable nature of a username is thought to not be a problem, since the strength of authentication depends on the strength of the credential that comes next, such as a strong password, protected by a strong hash and salting while stored.

Yet this view ignores a crucial fact about legacy user accounts: they are not alone, and they *share namespace* with any other such accounts, including those managed badly and vulnerable to trivial attacks, exfiltration, and malware infection.

1 “Three may keep a Secret, if two of them are dead”[3]

Ubiquitous sharing. A modern attacker rarely lacks the information about the identities of a targeted user on a variety of systems that authenticate that user. Often, the attacker can also collect the authenticating

[1]E.g., a Unix username is typically either a user's first name, last name, a concatenation of the first name's first letter and the last name, a concatenation of initials, the user's intended role or function, the name of a program, etc.; similarly but generally more strictly for Windows and MacOS X user account names.

[2]Weakest implementations can be exploited even at this point via crafted names with improperly recognized special characters; examples of SQL injection in web app authentication dialogs are many and varied. The authentication screen was never exempt from the pithy wisdom of the iconic “Little Bobby Tables” cartoon, `http://xkcd.com/327/`.

[3]Benjamin Franklin, Poor Richard's Almanack (1735).

credentials for these identities from a compromised platform, and pivot on these to compromise further platforms.

Credential sharing is a fact of life. Current user education practices are unlikely to curb it. An overwhelming majority of users are continually bombarded with requests to register for this or that service. Users are pushed to create accounts even to read their news, see cooking recipes, or get discounts on their grocery shopping. Not surprisingly, users employ eminently guessable password schemes or simply keep their passwords in an easily findable plain text file.

It is important to note that users are typically left on their own when creating a password even when password-creation software modules filter out and deny "weak" passwords such a dictionary words, short or all-letter passwords, etc. The choice is still the user's problem. Not surprisingly, users push back with a variety of heuristics that reduce their cognitive load. When users get no help with a task, it is futile to expect anything but the minimal amount of user effort to pass the bar set on the outcome.

It is impossible to expect that all users will stick to a strict "no shared credentials" discipline in creating these everyday accounts. Even if the anti-sharing policy is robust, and the users are persuaded to follow it, the platforms from which they access these accounts are likely to be shared—and are likely to retain tokens of user identity such as usernames that are usable by attackers. Moreover, since many services allow or encourage using an email address as either the primary or the alternative username, the sharing problem is considerably exacerbated.

Identity namespace sharing and attack planning. Even though password sharing has been recognized as a significant problem, username sharing, to the best of our knowledge, has not. Yet it is username sharing that provides the attacker with the next selection of targets for developing his attack tree. Indeed, without preserved identities of the user on a compromised platform planning of next targets to compromise would be at a disadvantage. Knowing or guessing the identity of the next hop on an attack path at the very least suggests that hop and allows the attacker to engage that hop node in a non-trivial protocol interaction. In other words, this kind of sharing informs the *lateral movement*, a standard attack tactic for exploring weaknesses in the target network, in which the attacker explores for weaknesses in lower value targets rather than going straight at the higher-value ones, and accumulates compromised accounts and systems. As such, it should not be overlooked.

2 Lateral movement and the unholy trifecta of sharing

According to Verizon's 2014 data breaches report [1], a predominant share of breaches involve compromised credentials of an authorized user. This result may be surprising, considering the apparent effort spent in user education, automated password strength testing at creation, explicit anti-sharing policies, and the like. We argue, however, that this result is predictable.

Specifically, we argue that the high incidence of account credentials being compromised is *an effect of scale* in the current mesh of predictable-namespace sharing. In a nutshell, it doesn't matter how strong individual account authentication is *on average*; it is the network-amplified cumulative effect of *the weakest* that adds up.

2.1 What enables lateral movement

The concept of lateral movement—compromising systems of equal or lower value to the attacker as a means of discovering an easier route to a higher value target through one of them—has long been a part of attacker practice. Although lateral movement is not free, as it increases the penetration footprint and with it the chances of accidental discovery, it is often a winning strategy, rather than going up against a high-value target directly.

Moreover, selection of "next-hop" targets for lateral attack planning rarely presents a challenge to the attacker. For example, a company's email or database servers may come hardened, auto-updated, and protected by some network IDS/IPS devices. The attacker may lack the means of compromising them. However, the personal laptop of the company's administrator may lack such protections, whereas the administrator's identity and various modes of online contact may be trivial to deduce from public DNS or WHOIS records.

Similarly, the identities of the C-level company officers are typically public record, and their contacts are likely to be easy to guess if not already public. In each case, the attacker has a clear choice of targets to attack via old-fashioned "social engineering", much cheaper than developing or acquiring exploits against maintained production software.

We distinguish the three points at which username sharing enables lateral movement.

1. User's client system. This platform is a commodity OS shared between professional activities, web shopping, social networking, and leisure web browsing. Even when issued strictly for professional activities, this platform is often used for web-based or HTML-based mail, de facto sharing the browser between data exchanged with employers' IT systems and advertisement-driven "free" websites in professional context.

   This OS-level sharing means that the attacker who gains control of the platform will also likely gain access to all credentials stored on it, even if the user follows all recommended steps to safeguard them. There are many examples of such attacks on commodity platforms, since they provide little or none hardware support to protect lower-privilege processes and applications from higher-privileged ones, or user applications from the operating system kernel. Thus an attacker who is able to compromize a device and to elevate privilege is automatically in possession of the all lower-privilege secrets. Although newer security technologies such as Intel's Software Guard Extensions (SGX) seek to change this, their proposed solutions require significant changes in how software is designed and written, and will need time to change the tide.

2. IT server systems. These systems accumulate both employee and customer information. A compromise of these systems, even if it doesn't directly yield credentials shared with other accounts, identifies many new targets for lateral attacks. These targets come with the additional information that they have a trusted, pre-existing business relationship with the compromised server and vendor, which the attacker will surely exploit. A careful attacker may even refer to specific assets of the targeted customer, previous transactions with the vendor, and so on—thus creating an illusion that the attack is just another regular transaction with the vendor.

   The customer may be more or less suspicious, may use some rules-of-thumb to screen scammers, or some other sound human-scale trust protocols [2]; still, compromising an endpoint of an existing relationship is likely to significantly increase the attacker's chances of success. Details of a previous business relationship are a natural "something you know" of the classic authentication triad. They will be used for on-the-spot trust decisions whether the official policy training blesses them or not. Certainly, such training will fail to anticipate all possible heuristic uses.

3. User's mind. Regardless of what cryptographic and systems security primitives an authentication scheme builds on, it also typically requires some user actions, such as clicking on buttons or links, entering usernames, emails, or passwords into text boxes, plugging in security tokens, checking email for additional information, etc. Depending on their context and circumstances, these actions are either correct or not; in the right context, they advance the authentication process as designed; in others, they lead to information disclosure and compromise.

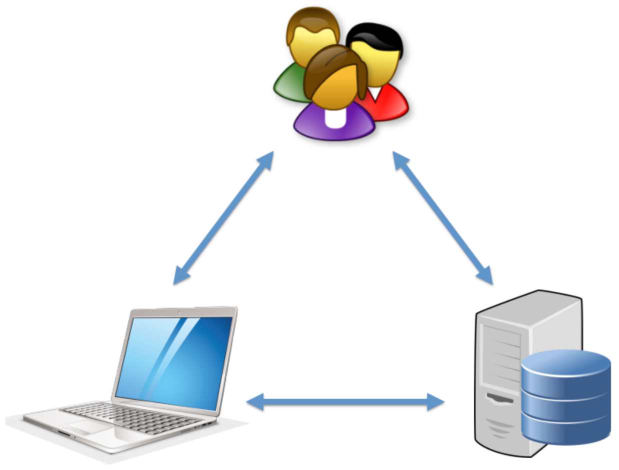
   Users judge these circumstances and select their actions based on their mental models of the authentication process. These models may be far from technological reality. Whenever the scheme requires that a username or password be explicitly entered under some circumstances, the mental model will create and "store" some representation of these circumstances—such as when a new authentication action may be required, what the text boxes look like, and what a successfully authenticated connection should look like. Unlike cryptographic protocols that can be specified precisely, these models are open-ended, and depend on the users' broadly varying experiences and environments.

   In fact, it is more accurate to speak of users' mental model or "trust signals" as a necessary component of an authentication scheme, just as important as the credentials and protocols used. When an attacker manages to replicate these signals for a user, this user will happily be led into any diversion of the protocol.

   Not only mental models likely differ from the technical reality, but the "mental storage" scheme also entangle all of its elements, making them a lot easier for the attacker to second-guess. Essentially, mental credential storage is not just a bottleneck for the entropy of secrets, but also the destroyer of protocols.

Thus user misperception of visual cues of the trusted path implementations is the other aspect of the "trusted path" challenge. For example, a specific GUI cue such as SSL's "golden lock" icon or a special window boundary, may be unforgeable by the system means in the designer's view, but a "similar enough" approximation may still trick the user. In particular, the SSL lock and other trust indicators in browsers could be successfully spoofed (see, e.g., [10, 11] for discussion).

It is typical for attackers to leverage any of these three loci of sharing to achieve compromise against the others. Visualizing the three items above as the vertices of a triangle, we can say that attackers use every side of the triangle to develop their penetration of a targeted organization.



2.2 Sharing with "second factor"

Sharing doesn't automatically go away with two-factor authentication (2FA) schemes. There are two main scenarios that allow it to persist.

Downgrading 2FA to the weakest single channel. The intelligible and predictable username—which is usually *the first thing to be transmitted* in most two-factor schemes regardless of what second factor is used—identifies the user's identity to attackers. The attackers then target that user with a custom scenario to downgrade the user's 2FA to the weakest single authentication factor. The downgrade attack specifically aims to provide apparent replication of messages or circumstances that the user trusts. Often, the attacker exploits subtle differences between an actual trusted channel used by 2FA and the user's concept of it.

Compromising the trusted path. A vulnerability in the general-purpose computing device used as a second factor—such as a smart phone—allows the attacker to compromise the second-factor credential while in transmission to the user. In this case, the user's only mistake is to assume that the delivery path for the "stronger" credential is immune from the attacks on the "first" authentication factor.

Thus the mechanics of this downgrade can exploit either the user's imperfect perception of what a trusted channel is, or imperfect technical reality of the channel itself. For example, Mulliner [4] describes compromising the path from the authentication server to the phone serving as the second-factor device in an OTP authentication scheme, realizing the first of the above scenarios. The second could be realized by a MITM attack on the smartphone's internal inter-app communication mechanisms such as an Android Binder man-in-the-middle (MITM) attack [5].

In both cases, the attacker targets the second-factor device to gain the shared information. We note that these "off by one-factor" attacks naturally fit within the above sharing trifecta.

OTP: is it really 2-factor? The above scenarios come into special prominence when the second factor scheme uses a smart phone to deliver a One-Time-Password (OTP). OTP occupies a curious place in the classic "something you know, something you have, something you are" authentication triad.

If the reader forgives us a passing Harry Potter allusion, the OTP's relationship to two-factor is somewhat like that of the platform $9\frac{3}{4}$ to its mundane neighbors, except that the journey that originates there often leads not to a world of magic but to the plain old Muggle reality of compromise. Indeed, an implicit assumption in the discussion of the two-factor schemes is that they draw their power from using completely *orthogonal* and independent mechanisms that cannot be compromised together by any single method (for example, a user can be socially engineered by a remote attacker to divulge his password, but to also gain the use of his hardware token the attacker must physically steal it, and so on). It is these orthogonality and independence that double the attacker's toil.

However, OTP is not exactly a combination of the independent "something you know" and "something you have". In fact, it is a combination of "something you know" and another "something else you know, if only for a brief period of time". This combination is subject to all the attacks both on the human user (the one whose knowledge is used authentication) and on the channel through which this new knowledge is transmitted. The human must correctly recognize, identify, and interpret both the OTP message as such and the interface to (manually) relay it into—which makes OTPs vulnerable to misrecognition, misidentification, and to erroneously divulging it.

Moreover, such messages are delivered to a device that the attacker possesses considerable knowledge of, rather than to an opaque physical token perfectly unknowable to attackers. Indeed, the user may in fact use the same email and web app accounts from the very same phone that is used to receive OTP! This is a far cry from the presumed orthogonality of the authentication triad.

It should be noted that modern smart phones possess powerful side-channels through which critical OTP information ("something else you know") can be captured by attackers. In particular, screen capture can decode even considerably distorted symbols [6, 7], a user-facing camera can be used to capture reflected images at much higher resolutions than expected [8], and even user typing can be captured via accelerometer sensor streams [9]—all via unprivileged apps conceptually isolated from any kind of user authentication.

Designing a visually unmistakable *and* trustworthy trusted path from the OS foundations to the user GUI in an environment shared by applications of inherently different security levels has been a research challenge since at least the time of the DoD "Rainbow Series". Despite many proposed solutions, this goal remains elusive. We posit that at this point a practical attitude must shift from trying for yet another isolation mechanism to *reducing the sharing.*

3 Sharing at scale, on the back of an envelope

Once attackers compromise a user account—any account—and thus acquire a selection of the next round of targets, how lucky do they need to be to succeed if they studiously explore each link and find the weakest?

Detailed evaluation would involve estimating the likelihood of success along each side of the above triangle. Instead, let us ask: what is the lowest probability of a user account compromise spreading to other user accounts at which a large IT system communicating with other large IT systems would see a 50% chance of being penetrated? Recall that, according to the Verizon report, credential compromise was involved in $3/4 = 0.75$, i.e., 75% of actual compromises. We will now sketch out a back-of-the-envelope estimate, following Dan Geer's very similar estimates of monoculture effects [3].

Assume that the probability of a successful lateral movement—i.e., using a known account name or identity to compromise another account used by the same individual or group—is ε. In a universe of N accounts, the probability of *none* being compromised is $(1-\varepsilon)^N$. Solving for 50%, we get

$$1 - \varepsilon = 0.5^{\frac{1}{N}}$$

or

$$\varepsilon = 1 - 0.5^{\frac{1}{N}}.$$

Thus for $N = 5,000$ accounts (that is a 1000 employee business, assuming that employees, on average, have 5 accounts shared and targetable via lateral attacks), we only need the infection probability ε to be 0.00014, that is, the odds of more than 1 in 10000.

How scary are these odds, and do they account for the observed high number of account credential compromises? Back in the day when the odds of an account compromise depended only on the strength of that account's credentials, it may not have been scary – after all, the odds of dying in a car accident in a year are estimated at being around 1 in 5000, less than twice as much. However, if we figure in all the accounts that a typical employee has, and the percentage of accounts compromised (and thus usable for attacker's lateral movement), the odds of a company's account being compromised are not good.

If we estimate that an average US employee of interest to an attacker has at least 20 different accounts and uses no more than 10 different usernames, we get the re-use factor of a username as 2. If we assume that the number of such employees over all industries using modern IT is around 100 million worldwide, we'll estimate that they have 2 billion accounts, with only 1 billion unique usernames. If the attackers manage to assemble a list of only 10 million compromised usernames, it means that they can target at least 1 percent (or 1 out of every 100 accounts). Thus, they only need the odds of one in a thousand to succeed to get to the compromise levels of the Verizon report.

Conclusion

The best way to address credential-compromise vulnerabilities is to limit the attacker's capability of lateral movement, and thus of finding and following weaker links until some path leads to a compromise. With today's mesh of accounts and shared environments, the strength of the strongest authentication links hardly matters anymore; it's the mass of weak(est) links that define the security landscape, as the above calculation shows. Moreover, in this mesh of accounts, two-factor authentication can be degraded to a vulnerable one-factor easily enough by targeting the identified user's "wetware". Systems where parts of credentials are intelligible to attackers essentially make it easy for the attacker to select the next round of links to probe for weakness.

One mitigation strategy to make sure that such targeted lateral movement doesn't happen is to ensure that *no part of user credentials is intelligible to the attackers.* We note that this also fulfills the implied promise of 2FA: the factors being orthogonal and retaining their independence no matter what set of remote communications the attacker may concoct.

From this perspective, systems where credentials are never exposed in an unencrypted (and not just human readable) form and are handled within trusted hardware at all times are preferred. Of course, this requirement means complete elimination of "something you know" as the "first factor". It may seem that introduction of a hardware token strengthens authentication by itself, but it doesn't, as the various downgrade schemes show.

"Something you know" can still be used for protection of the hardware factor from unauthorized use—but not to locate the user across systems. It should be limited to the most direct interaction between the hardware first factor and the owner (which is still susceptible to eavesdropping, but on a much lower scale), and to the addition of multiple additional verification steps (like biometrics, time and location limits and such), based on specific knowledge about the user by an application.

The obvious benefit of the unexposed credential is that the user can explicitly control its sharing. When one reuses a hardware token, with multiple applications, one must make its secret credentials unique for each application. In this case, through combining the hardware credentials with the application credentials, a unique credential pair can be handled with less probability of compromising the account. Credential pairs created in this fashion can be handled by the third party credential providers, with less risk of being compromised, since they are never exposed to the outside world, and do not immediately tip off the attacker as to the next human target to try (except the aggregate server where the credential may point to).

Explosion of enmeshed accounts that share their user interfaces between messages of different trust levels has redefined authentication failure; compromise is now largely a matter of lateral movement to find the weakest link. Defenders may complain that credential compromise attacks are "stupid", but as any attacker knows, it's not "stupid" if it works. Intelligible credentials are a boon to the attacker whether they are used in a single-factor or a multiple-factor scheme–it may be the time to get rid of them.

References

[1] "2014 Verizon Data Breach Investigations Report", www.verizonenterprise.com/DBIR/2014/

[2] Matt Blaze, "Toward a Broader View of Security Protocols", In proceedings of SPW12 (Cambridge, 2004), www.crypto.com/papers/humancambridgepreproc.pdf

[3] Dan Geer, "Monoculture on the Back of the Envelope", USENIX ;login:, December 2005, www.usenix.org/legacy/publications/login/2005-12/openpdfs/geer.pdf

[4] Collin Mulliner, Ravishankar Borgaonkar, Patrick Stewin, and Jean-Pierre Seifert, "SMS-Based One-Time Passwords: Attacks and Defense", In proceedings of DIMWA 2013, LNCS 7967, http://www.mulliner.org/collin/academic/publications/mulliner_dimva2013.pdf

[5] Nitay Artenstein and Idan Revivo, "Man in the Binder: He Who Controls IPC Controls the Droid", BlackHat Europe, 2014, https://www.blackhat.com/docs/eu-14/materials/eu-14-Artenstein-Man-In-The-Binder-He-Who-Controls-IPC-Controls-The-Droid.pdf

[6] Greg Mori and Jitendra Malik, "Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA", In proceedings of the IEEE Computer Society conference in Computer Vision and Pattern Recognition, 2003, www.cs.sfu.ca/~mori/research/papers/mori_cvpr03.pdf

[7] Elie Bursztein, Jonathan Aigrain, Angelika Moscicki, and John C. Mitchell, "The End is Nigh: Generic Solving of Text-based CAPTCHAs", In proceedings of the USENIX Workshop on Offensive Technologies, 2014 www.usenix.org/system/files/conference/woot14/woot14-bursztein.pdf

[8] Tobias Fiebig, Jan Krissler, and Ronny Hänsch, "Security Impact of High Resolution Smartphone Cameras", In proceedings of the USENIX Workshop on Offensive Technologies, 2014, www.usenix.org/system/files/conference/woot14/woot14-fiebig.pdf

[9] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, Joy Zhang, "ACCessory: Password Inference using Accelerometers on Smartphones", In proceedings of the ACM HotMobile, 2012, www.hotmobile.org/2012/papers/HotMobile12-final42.pdf

[10] Zishuang (Eileen) Ye and Sean W. Smith, "Trusted Paths for Browsers", In proceedings of the USENIX Security Symposium, 2002, https://www.usenix.org/legacy/event/sec02/full_papers/ye/ye.pdf

[11] Zishuang (Eileen) Ye, Sean W. Smith, and Denise Anthony, "Trusted Paths for Browsers", ACM Transactions in Information Systems Security, vol 8, pp. 153-186