

Authentication Systems: Security Analysis



Secondary Authentication

- People *will* forget passwords
- *That happens more if you require password rotation, a practice that is now disfavored*
- People *will* lose tokens or break their phones
- They still have to log in. How?

Secondary Authentication

- All real services need some way to recover from these errors
- How this is done varies, depending on the environment, the value of the service, and the cost of lack of access
- And: in many situations, you have to guard against corrupt insiders

- You can't afford much customer care
- (Gmail has **1.8 billion** users)
- You *must* have a secure(-enough) automated recovery procedure
- You could send an email—but what if it's the email password that's lost?
- Perhaps a text message—but that carries **privacy risks**
- Other choices?

Email-based Secondary Authentication

- For low-value sites, the most common solution is to send an email
- This means that email accounts are exceedingly precious
- If you have only one service for which you can use strong authentication, it should be your email account
- Fortunately, Google is a strong proponent of FIDO2

Knowledge-Based Authentication

- Have a help desk (or web server) ask questions that (in theory) only the real person will know
- Large commercial data brokers know an amazing amount about people!
- Of course, attackers have been known to tap into those databases, too
- (Who are the attackers?)
- In a corporate setting, there may be values not in these databases, e.g., an employee ID number
- (What is the Card Number on the back of your CU ID card? But what if it's your card that's lost? In some places, you use your employee ID card as a second factor)

Out-of-Band Authentication

- Require someone to come in in person
- Have their supervisor vouch for them
- Send physical mail—some banks do this
- Use ANI (automatic number identification)—*not* the same as CallerID, which is easily spoofed

- A lot depends on your threat model—some attacks on secondary authentication require more resources or access
- (Can a former intimate partner intercept physical mail?)
- Many of the knowledge-based systems require answers with limited entropy, often far less than the primary authentication systems

Analyzing Password Security Practices

What's the First Question?

What's the First Question?

- What are we trying to protect, and against whom?

What's the First Question?

- What are we trying to protect, and against whom?
- That's always the first question! So: what are we trying to protect?

What's the First Question?

- What are we trying to protect, and against whom?
- That's always the first question! So: what are we trying to protect?
- We have to protect the plaintext password—possession of it gives access to this site and (probably) many others
- We have to protect the ability to log in, even without knowledge of the password
- We want maximal protection against all enemies
 - Why “all enemies”?

What's the First Question?

- What are we trying to protect, and against whom?
- That's always the first question! So: what are we trying to protect?
- We have to protect the plaintext password—possession of it gives access to this site and (probably) many others
- We have to protect the ability to log in, even without knowledge of the password
- We want maximal protection against all enemies
 - Why “all enemies”?
 - Passwords are used in almost all contexts, even if supplemented by 2FA

- Never store passwords in plaintext—what if the machine with the password store is hacked?
- Use hashed passwords, not encrypted passwords. (Why?)
- Implication: no password recovery, only password reset
- What's next?

Second Step

- Assume that the machine holding the hashed passwords is hacked—now what?

Second Step

- Assume that the machine holding the hashed passwords is hacked—now what?
- Harden the hash against password-guessing
- This is as far as Morris and Thompson went—is there more?

- In 1979, computers were expensive; passwords were generally entered over dial-up phone lines
- Governments could do modem taps; few others could
- Today, passwords are entered over the Internet—much easier to tap
- Conclusion: we *must* use encryption
- More?

Fourth Step

- Passwords today are mostly for web services, but web servers are fragile
- Conclusion: store the hashed passwords on a login server that is (somehow!) less vulnerable to being hacked
- Ideally, encrypt the password from the user to the login server, so that even the web server can't see it (but this is rarely done)
- More?

The User Perspective

- The above is a server-centric perspective
- Users have to worry about many sites, not just one or two
- In Morris and Thompson's day, very few people had more than one login—today, many people have hundreds of logins
- User perspective: some sites *will* be hacked—how do they protect their logins on other sites?

The User Perspective

- The above is a server-centric perspective
- Users have to worry about many sites, not just one or two
- In Morris and Thompson's day, very few people had more than one login—today, many people have hundreds of logins
- User perspective: some sites *will* be hacked—how do they protect their logins on other sites?
- Answer: you *must* use a separate password for each site
- Thought exercise: does it work to have separate classes of password, for sites of different sensitivity?

Why Didn't Morris and Thompson Go Further?

Why Didn't Morris and Thompson Go Further?

- Technology of the time!
- The external attack surface for a computer of that era was typically very small: the login service
- 👉 Networking was in its infancy
 - Users had *no* local compute capacity and little or no local storage
 - A separate login computer would have cost (at a minimum) tens of thousands of dollars; it and the network would have been additional availability failure points
 - What do you do with a stolen password file? Compute time was expensive and not that readily available

The worst mistake in technology is to give yesterday's answers to today's questions. The second worst mistake is to ignore yesterday's answers. . .

Definition: Attack Surface

Definition: Attack Surface

“The set of points on the boundary of a system, a system element, or an environment where an attacker can try to enter, cause an effect on, or extract data from, that system, system element, or environment.”

NIST Computer Security Resource Center

Implicit Constraints


- Some of the constraints Morris and Thompson faced were implicit
- Their threat model wasn't as clearly spelled out as we would do today
- Getting that right is crucial, especially for security for long-lasting systems
- Do you have an ongoing process for monitoring changes in technology and threat model?

Password Managers

Behold, the fool saith, “Put not all thine eggs in the one basket”—which is but a manner of saying, “Scatter your money and your attention;” but the wise man saith, “Put all your eggs in the one basket and—watch that basket.”

Puddn’head Wilson
—Mark Twain

Synchronization Issues

- Most people have more than one device—how do they synchronize encrypted password manager “vaults” between them?
- Inconvenient but (probably) secure mechanism: manual copying, e.g., via USB flash drive
- More common: cloud storage
-  A proprietary solution or a generic cloud service like Dropbox or Apple’s iCloud?
- Generic services are harder for users to set up, but proprietary solutions become interesting targets for attackers
- What are the tradeoffs?

- Easier for service operator to spot unusual attempts to grab someone's vault
- Can even mandate (or at least support) 2FA to gain access
- But—what if the service is hacked and someone steals all of the vaults and then runs password-guessing attacks?
- (1Password uses a random AES key on the user's devices to do the encryption—but when you add a new device to your account, you have to manually copy over this key)

Generic Cloud Solutions

- More setup for the users
- More opportunity for improper sharing
- More code, to support several different cloud storage services
- No easy central control or monitoring of vault access, or of the login requirements for the generic service
- No easy way to detect improper vault access

One the User's Machine

- A manager that's integrated with the browser can fill in fields, which is very convenient
- 👉 It can also make sure that the password is only used with the proper site
- When you change your password, it's easier for the manager to detect this and store the update
- But—browsers have a large attack surface; too-close integration exposes the *decrypted* vault to malware in the browser
- A separate program runs less risk that way, but is much less convenient to use

Tradeoffs

- It is impossible for people to remember many strong, different passwords
- A password manager does that for you; most will even generate random passwords on request
- Current NIST guidance discourages password rotation and strength requirements; instead, it stresses password length and non-reuse. Why?

- It is impossible for people to remember many strong, different passwords
- A password manager does that for you; most will even generate random passwords on request
- Current NIST guidance discourages password rotation and strength requirements; instead, it stresses password length and non-reuse. Why?
- Contrast: a 12-character password using all 94 printable ASCII characters:

$$94^{12} \approx 4.7 \cdot 10^{23} \text{ possibilities}$$

versus 18 lower-case characters:

$$26^{18} \approx 2.5 \cdot 10^{25}$$

- And using lots of special characters is very hard on a phone

Analysis of Other Authentication Mechanisms

What Are the Failure Modes?

Let's look at the failure modes for different authentication methods and different aspects of the system.

- Passwords
- Hardware time-based one-time passwords (TOTP)
- Software time-based one-time passwords (TOTP)
- Hardware or software tokens
- Text messages (SMS)
- FIDO2
- Server
- Link
- Browser
- User
- Device or service (if any)

(We've already discussed them extensively...)

Server Exposure of hashed, encrypted, or plaintext passwords

Link Eavesdropping

Browser Malware in browser; keystroke loggers

User Many...

Service Attacks on password managers

Server Exposure of key database (Remember: $H_k(T)$ is displayed)

Link Eavesdropping

Browser Malware in browser or keystroke loggers plus realtime OTP relay

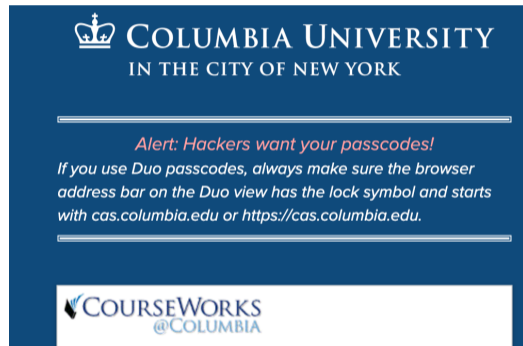
User Loss of device; phishing with realtime OTP relay

Service Extraction of secret from device—is it tamper-resistant?

- Lure the user to a bogus site via phishing
- Have them enter the current value (or approve the login)
- Relay that in realtime to the actual site you want to penetrate
- Will users actually properly follow the cautions? (Do you? Did you ever notice that caution?)



Note the URL in that warning—we'll come back to it much later in the term



Server Exposure of key database

Link Eavesdropping

Browser Malware in browser or keystroke loggers plus realtime OTP relay

User Phishing with realtime OTP relay

Service Extraction of secret from service or OS

Server Possible exposure of key database

Link Eavesdropping

Browser Malware in browser

User Phishing with realtime OTP relay

Service Extraction of secret from device or OS

Text Messages

Server N/A

Link Eavesdropping, including on phone network

Browser Malware in browser; keystroke loggers

User Phishing with realtime OTP relay

Service SIM-jacking; SS7 attacks; enemy-controlled phone system

Server N/A

Link N/A

Browser Malware in browser; silent malware

User Activation for wrong site

Service Extract key from device or OS

Conclusions

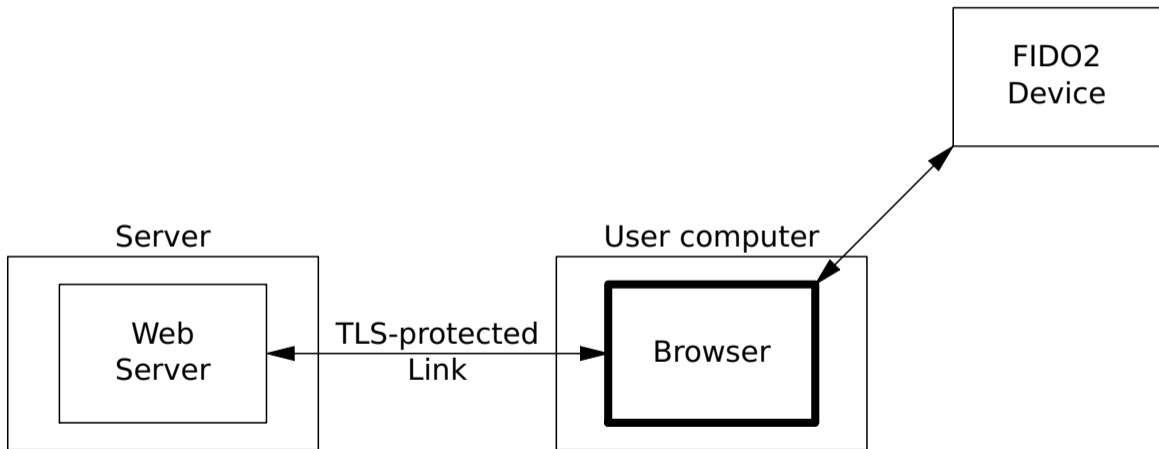
- Most technologies are vulnerable to eavesdropping—but TLS (or equivalent) is near-universal. (FIDO2 has special provisions to strengthen TLS under certain circumstances.)
- TOTP and tokens are at *more* risk after server compromise than (properly stored) passwords—but since the secret is different for different sites, the attacker's benefit is much less
- FIDO2 seems the best—but there are risks. What are they?

The Risks of FIDO2

Let's look again at the “Details of FIDO2” slide. . .

- At registration time, the device receives **from the browser** “a hash of the origin (combination of protocol, hostname and port)”
- The user has to “activate” the device via some manual action
- The device sends the site a “key handle” along with the key. The key handle is just a pointer to the key pair, and is tied to that origin
- To authenticate, the **browser sends** the device the request origin and the random challenge; after (optional) activation, the device signs the whole thing and sends it back
- (There are more details—see the reading!)

Communication Path to the FIDO2 Device



The Browser is Crucial!

- When a user authorizes authentication, to which site is that happening?
- The *browser* displays it—but is the browser trustworthy?
- What if some malware is silently lurking until it detects an authentication request, and gets its request in first?
- If the “no activation” option is chosen, a compromised browser can log into your bank at any time

The Imperva Red Team recently disclosed a vulnerability, dubbed [CVE-2022-3656](#), affecting over 2.5 billion users of Google Chrome and Chromium-based browsers. This vulnerability allowed for the theft of sensitive files, such as crypto wallets and cloud provider credentials.

Mozilla Releases Security Updates for Firefox

Original release date: January 18, 2023



Mozilla has released security updates to address vulnerabilities in Firefox ESR and Firefox. An attacker could exploit some of these vulnerabilities to take control of an affected system.

Browsers!

- The security of FIDO2 *authentication*—the real-world process, not the protocol or device!—depends crucially on the security of the browser
- But browsers have a large attack surface
- Sandboxing the browser will help protect against theft of secrets from the OS, but will *not* protect against bogus authentication under the control of a subverted browser

“As long as U2F devices can be accessed directly from user space on the client OS, it is possible for malware to create a keypair using a fake origin and exercise the U2F device. The U2F device will not be able to distinguish ‘good’ client software from ‘bad’ client software. . . It is not in scope to protect against this situation.”

Conclusions

- No method of authentication is perfect
- A FIDO2 device with a display would be stronger—*if* people actually looked at it
- And: would that increase the attack surface of the device?
- FIDO2 still seems best, but we really need to harden browsers

Questions?



(Red-bellied woodpecker, Riverside Park, December 20, 2021)