Insecure Real-World Authentication Protocols (or Why Phishing is so Profitable)

Richard Clayton

University of Cambridge, Computer Laboratory, William Gates Building,
15 JJ Thomson Avenue, Cambridge CB3 0FD, United Kingdom

richard.clayton@cl.cam.ac.uk

Abstract. The users of online banking systems are currently at risk from "phishing" scams. Confidence tricksters persuade them to visit fraudulent websites and use their authentication credentials to steal from the victims' accounts. We analyse the authentication protocols used for online banking, find that they are entirely inadequate, and consider how to improve systems design so as to discourage attacks.

1 Introduction

"Phishing" is the use of email messages to entice customers of legitimate companies into the sharing of passwords or other credentials such as credit card numbers or PINs. The third party who has successfully conned the customer into revealing their details is then able to masquerade as the customer in order to steal money or services.

The earliest recorded use of the word "phishing" is in a Jan 2 1996 Usenet article by `drspamcake@aol.com` [2] and relates to the theft of America Online (AOL) passwords. However, the actual attacks are far older and, for example, the sending of instant messages, apparently from AOL staff, that asked for a password was so widespread that by 1995 the AOL software package contained a specific "report password solicitation" button [5]. A 1990 paper on a closely related attack, obtaining passwords from public terminals by altering the firmware [4], uses the spelling "fishing".

In recent years, the term phishing has come to be specifically associated with the operation of fake websites that purport to be a bank or an online system such as eBay (`www.ebay.com`) or PayPal (`www.paypal.com`). The customer is sent an email that claims to be from the bank and is invited to click on a link within it. This takes them to the fraudster's site where a superficially plausible web page is used to capture the customer's access credentials. The fraudster then uses these credentials to impersonate the customer.

Phishing emails and the associated websites are now almost indistinguishable from legitimate activity (not least because marketing departments continue to value the use of clickable links in their emails). MailFrontier have run a couple of online quizzes [6] and found that about 30% of respondents make at least

one mistake in categorising ten emails into legitimate or con-trick. So we should look to the authentication protocols and the overall system design to prevent phishing, rather than to user education or a change to the email standards.

In this paper we consider the inadequacy of current authentication protocols in section 2. Since no easy solution is apparent we consider how websites are authenticated in section 3. A real fix still being absent, in section 4 we consider how client certificates can also fail to deliver. We conclude that existing security primitives and security protocols fail to provide the tools needed to secure real-world online applications, though all is not lost because high-level system design changes may be sufficient to make online banking "secure enough".

2 Authentication Protocols

The standard protocol used for an online banking session with a bank B is for the Alice the user A, to supply a login name and a shared secret (password) S:

$$A \longrightarrow B : A, S$$

If the phisher, P, persuades Alice to visit his website then clearly he can masquerade as Alice:

$$A \longrightarrow P : A, S$$
$$P \longrightarrow B : A, S$$

In fact, because there is no freshness in this protocol, P can do this at any future time until Alice changes her password (or the Bank freezes the account).

This can be tackled by using a one-time password S_n which can never be reused. Example implementations would be a pad of single-use random numbers shared between Alice and the Bank, or a hardware token device such as RSA's SecurID [7]. This does not prevent the man-in-the-middle attack:

$$A \longrightarrow P : A, S_n$$
$$P \longrightarrow B : A, S_n$$

but P can only use the one-time password on the one occasion. The usage must also be done quickly. In the SecurID case, the password is inherently time-limited; but in both cases Alice is very likely to try and contact the bank again and as soon as she reaches the correct destination she will probably discover any attempted fraud before any money has been transferred.

The phisher can discourage Alice from making a new connection by acting as a man-in-the-middle for the entirety of her session. Most simply this would mean relaying all messages back and forth between Alice and the Bank except for a final "logoff"; thereafter P can perform extra transactions to his own benefit.

The bank can prevent extra transactions that are unknown to Alice (and force the phisher into providing a live service) by insisting on a fresh password

for every transaction, such as paying a bill, that occurs within the banking session. Essentially Alice is signing each transaction T_n by the password S_n.

Unfortunately, since there is no binding between T_n and S_n this still does not prevent the man-in-the-middle attack where P replaces T_n ("pay the gas bill") by a wicked ("send all A's money to P") transaction W_n.

$$A \longrightarrow P : T_n, S_n$$
$$P \longrightarrow B : W_n, S_n$$

Of course, if this was a purely computer protocol in the Needham-Shroeder tradition then one would be looking to establish a binding between the signature and that which was signed, viz we'd design messages such as:

$$A \longrightarrow B : \{A, B, \text{nonce}, T_n\}_{K_A^{-1}}$$

where the messages are cryptographically signed with Alice's private key K_A^{-1}. Since Alice will be unable to perform cryptography in her head, we've moved into a more complex area than we've considered so far. We'll return to cryptography below, but first we'll consider another notion: the use of secure channels.

2.1 Secure Channels

If a secure channel from the Bank to Alice is available then this can be used to prop up an otherwise insecure protocol. By a secure channel we mean for example delivering a "text message" to Alice's mobile phone, or sending an email to a previously agreed address. Although P has managed to persuade Alice to visit the wrong website, P is not omnipotent enough to interfere with the rest of Alice's activities.

Of course it would be best if all transactions were carried out over secure channels ! – but email and text messages are not as convenient as using a website, because they are slower and far less interactive.

As an example of "propping up", perhaps at the end of the session the Bank could send Alice an email containing a summary of transactions. Alice would then compare this with her records and any wicked transactions could be undone. Bank transfers currently take several days, so Alice can be given the time to do her checking before an irreversible event occurs. In protocol terms, using the symbolism $\overset{secure}{\longrightarrow}$ to mean a message from the Bank over the secure channel we would have:

$$\begin{aligned}
A &\longrightarrow B : A, S_0 \\
A &\longrightarrow B : T_1, S_1 \\
&\cdots \qquad \cdots \\
A &\longrightarrow B : T_n, S_n \\
B &\overset{secure}{\longrightarrow} A : \text{A}, T_1, \ldots, T_n
\end{aligned}$$

In practice this scheme would fail if P was able to borrow one of Alice's transaction keys to validate a change to her email address (viz to alter the secure channel) and then supply a forged email contained a spurious set of transactions. Similarly, the Bank is very likely to report upon transactions in human friendly terms such as "Payment to 'Gas Company' (a/c 01–02–03 1234567), £100" and so there is a risk that P can change the account number associated with the Gas Company and Alice will fail to see the fraud.

This suggests that there is a need for multiple levels of authentication; run-of-the-mill validation of transactions, and higher levels for operations such as changes to payees, inspection and alteration of out-of-band contact details. Further thought will show that the lower level of authorisation need not be onerous since it just prevents denial-of-service attacks (why would P go to considerable trouble to cause Alice to make specious payments to the Gas Company, who can be trusted to refund the excess?). However, the authenticators S_n will have significant value when a sensitive operation is occurring. The highest level of authentication (perhaps using old-fashioned pen and ink?) is needed for changes to the secure channel itself, which can be made even more robust by insisting that it is impossible to inspect the current setting.

However, the effect of all of this analysis and invention is merely to change the nature of phishing from "visit my website and type in an authenticator" to "visit my website and perform sensitive operations" which, since there are many plausible reasons for having to do sensitive operations, is unlikely to slow down the confidence tricksters significantly.

It is clear that we need proper cryptographic signing of Alice's messages, but this requires special software to achieve. Some cryptographic software ships as standard within web browsers. Maybe we can prevent phishing by having Alice realise that she has reached the wrong website?

3 The Standard Website Security Model

Secure Sockets Layer (SSL) was invented by Netscape Development Corporation in 1995 and version 3.0 [3] has become widely deployed. A closely related protocol called TLS (Transport Layer Security) was standardised by the IETF in 1999 [1]. By convention, websites using these protocols have URLs that commence `https` rather than `http`.

SSL/TLS is intended to provide a private connection, which cannot be eavesdropped, between two entities using symmetric encryption with a specially negotiated session key. The connection is reliable in that a message integrity check is used to ensure that messages have not been altered in transit. The entities can establish each other's identity by inspecting cryptographically signed packets. Certificates issued by a mutually trusted third party can be exchanged to demonstrate the authenticity of those signatures.

In practice, web sites have a certificate signed by a CA (Certificate Authority) such as Verisign. The consumer's browser is pre-loaded with the root certificate for the CA and can verify that the certificate presented by a website is owned

by that website. Unfortunately, customers seldom understand what has been verified by an `https` connection and believe that very different guarantees are in place.

If a fraudster obtains a certificate for `www.fakebank.com` then the certificate issued by Versign would correctly show that the site being connected to was indeed the promised `www.fakebank.com` and not an imposter. However, the consumer will believe that Verisign is promising that the site is wholesome and indeed that it is somehow related to the `www.truebank.com` that the customer thought they were visiting. This is entirely clear in protocol notation:

$$P \longrightarrow A : \{\ \texttt{site-P}\ \}_{K_{CA}^{-1}}$$

where A reads `site-P` as `site-B` and also believes the CA guarantees P's probity.

In the real world, if a fraudster obtains a certificate, sets up their site to resemble a major bank and then sends out a phishing email to attract the gullible then the emphasis placed on the value of using `https` is likely to make this a more successful con than if `http` had been used. The only reason this is not more common is the high cost of certificates and a slight risk that the CA might become aware of the fraudster's identity.

4 Client Certificates

As noted above, SSL is capable of securing both ends of a conversation. For this to occur the customer must install a "client certificate". The bank (who will not confuse `site-A` and `site-P`) can verify that there is no man-in-the-middle. In essence, the two parties A and B swap certificates signed by the CA, then use the Diffie-Hellman protocol to agree a communication key K_{Comm} and sign messages attesting to its value:

$$A \longrightarrow B : \{\ \texttt{site-A}, K_A\}_{K_{CA}}, \{K_{Comm}\}_{K_A^{-1}}$$
$$B \longrightarrow A : \{\ \texttt{site-B}, K_B\}_{K_{CA}}, \{K_{Comm}\}_{K_B^{-1}}$$

P is unable to forge either of these two messages and cannot interpose himself between A and B without causing the communication keys on the two links to differ – because those keys are constructed from secrets supplied by both ends of each link and A and B will not choose the same values.

The problem with client certificates is that it is expensive to operate the necessary certificate issuing system and it will prevent customers from using random machines (at the office, or on holiday) to do their banking. Therefore the use of client certificates has been restricted to high value systems such as share trading where the inconvenience of only being able to use a machine with the certificate installed is not significant.

However, if a certificated connection was only needed for the special transactions we identified earlier then this might be an acceptable solution. One could

visit a beachfront cybercafé to pay the gas bill, but could only set up a new payment destination from the living room at home.

It is also possible to use the secure channel mentioned above to replace the client certificate. The Bank can be verified by making an `https` connection, albeit with the risks already mentioned. The user must then compare the value of K_{Comm} on their machine with the value sent over the secure channel.

$$B \stackrel{secure}{\longrightarrow} A : K_{Comm}$$

Although in theory this works well, the main difficulty is that it hard to display the value of K_{Comm} within today's browsers and downloading a program to display it provides another opportunity for the phishers to con the users. Even if the value was displayed correctly, a phisher acting as a man-in-the-middle might be able to overlay that part of the screen with a misleading value[1], much as phishing sites today overlay "lock symbols" and URLs with their own graphics.

Of course, if phishers change their *modus operandi* into asking Alice to mail in her certificates ("we've discovered a security fault and need to replace them") then user certificates will become a significant liability because they do not usually require any password or other authorisation to be activated.

5 Conclusions

The types of authentication currently employed by online banking systems are, when analysed as security protocols, entirely ineffective. Some simple changes can make the phisher's task significantly harder – requiring them to run realtime man-in-the-middle attacks and forcing them to persuade customers to perform unnecessary sensitive operations. Although being in "the middle" and dynamically altering the traffic is conceptually simple, there are a number of things that banks could do to ensure that it is far from straightforward – and hence it might become difficult for the phisher to automate.

Cryptography fixes the problem at the protocol level, but in the real world there are significant limitations in how it can be effectively deployed.

However, from the banks' point of view, it may not be necessary to provide a perfect solution. If they can make phishing for bank details harder than capturing merchant accounts on Amazon or Tesco then the attackers will change their targets. As when two hunters are being chased by a bear, it is not necessary to outrun the bear, but merely to go faster than the other fellow.

References

1. Dierks, T. and Allen, C.: The TLS Protocol, Version 1.0, IETF, RFC2246, Jan 1999.

[1] P knows the value of K_{Comm} that will be sent via the secure channel, because P is using that key for their own connection to B.

2. DrSpamcake: Get on aol from off aol. alt.online-service.america-online, 2 Jan 1996. http://groups.google.com/groups?selm=4calah$eoh@newsbf02.news.aol.com
3. Freier, A. O., Karlton, P. and Kocher, P.C: The SSL Protocol Version 3.0. IETF Internet Draft <draft-freier-ssl-version3-02.txt>, 18 Nov 1996.
4. Harriman, D. D.: Password Fishing on Public Terminals. Computer Fraud and Security Bulletin, Elsevier Science Publishers, New York, Jan. 1990, pp. 12–14.
5. Lee, L.: AOL scam warning. bit.listserv.christia, 29 Sep 1995. http://groups.google.com/groups?selm=950929165422_112740484@mail02.mail.aol.com
6. MailFrontier Inc: MailFrontier to Unveil Phishing IQ Test II at Inbox East. Press Release, 11 Nov 2004. http://www.mailfrontier.com/press/press_phishtest2.html
7. RSA Security Inc: RSA SecurID Authentication. http://www.rsasecurity.com