

[NT] Security Side Effects of Word Fields

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2002-08/0106.html>

From: support@securiteam.com

Date: 08/27/02

From: support@securiteam.com

To: list@securiteam.com

Date: Tue, 27 Aug 2002 21:12:14 +0200 (CEST)

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.

Security Side Effects of Word Fields

SUMMARY

Alex Gantman has stumbled onto a couple potential security issues in Microsoft Word. In both cases, the adversary (mis)uses fields to perpetrate the attack. It is important to note that fields are not macros and, as far as Alex knows, cannot be disabled by the user. Alex is going to provide a basic description along with a proof-of-concept demo. Alex is certain that someone with free time and imagination can expand on these principles, possibly applying them to other products.

DETAILS

Following tradition Alex will use Alice and Bob as the two parties involved. Alice will be the adversary.

1) Document collaboration spyware

Attack Basics:

Alice sends Bob a Word document for revisions. After Bob edits, saves, and sends it back via email to Alice. The file being sent back will also include contents of another file(s) from Bob's computer that Alice has specified a priori. To achieve this, Alice embeds the INCLUDETEXT field into the document. The field results in inclusion of a specified file into the current document. Of course, Alice must be careful include it in such a way that it does not become apparent to Bob. Alice can do all the usual

things like hidden text, small white font, etc. Alternatively (and in Alex's opinion cleaner, she can embed the INCLUDETEXT field within a dummy IF field that always returns an empty string. In this case, the only way Bob can notice the included file is if he goes browsing through field codes.

Attack Improvements:

The disadvantage of the basic attack is that Alice must rely on Bob to update the INCLUDETEXT field to import the file. If the document is large and contains tables of contents, figures, etc. then Bob is very likely to update all the fields. However, Alice would like to make sure that the field is updated regardless of whether Bob does it manually or not. Automatic updates can be forced if a DATE field is embedded into the INCLUDETEXT and it is the last date field in the document.

Proof of concept:

Inserting the following field structure into the footer of the last page will steal the contents of c:\a.txt on the target's computer. Keep in mind the plain curly braces below must actually be replaced with Word field braces (you can use the menus to insert fields one by one).

```
{ IF { INCLUDETEXT { IF { DATE } = { DATE } "c:\a.txt" "c:\\a.txt" } \* MERGEFORMAT } = "" "" \* MERGEFORMAT }
```

Countermeasures:

The only thing you can do now is decide how paranoid you want to be. If you must edit and send out a Word file with unknown origins, you may want to manually go through the fields. It would be nice to be able to force user confirmation (via a dialog box) for all includes. Alternatively, one could write a scanner. Of course, an optional standalone checker will never be used by those most at risk.

2) Oblivious signing

Attack Basics:

Alice and Bob want to sign a contract saying that Alice will pay Bob \$100. Alice types it up as a Word document and both digitally sign it. In a few days Bob comes to Alice to collect his money. To his surprise, Alice presents him with a Word document that states he owes her \$100. Alice also has a valid signature from Bob for the new document. In fact, it is the exact same signature as for the contract Bob remembers signing and, to Bob's great amazement, the two Word documents are actually identical in hex. What Alice did was insert an IF field that branched on an external input such as date or filename. Thus, even though the sign contents remained the same, the displayed contents changed because they were partially dependent on unsigned inputs. The basic point is that very few users know the actual contents of their Word documents and it should be obvious that one should never sign what one cannot read. Of course, Bob could contest the contract in court. An expert witness (that is actually an expert) could easily demonstrate that there are unsigned inputs and therefore it is not clear which version was actually signed. Thus, Bob can get out of the fraudulent contract. However, the same logic will hold for Alice and she gets away without paying Bob \$100 she signed for. Thus, an

Securiteam: [NT] Security Side Effects of Word Fields

adversary can build in a free escape clause. Note that I am just speculating about all the legal aspects.

Proof of concept:

Inserting the following field structure at the tail of the document will cause "Hello" to be displayed if the filename is "a.doc" and "Bye" otherwise.

```
{ IF { FILENAME \* MERGEFORMAT { DATE } } = "a.doc" "Hello" "Bye" \* MERGEFORMAT }
```

Countermeasures:

Do not sign dynamic documents (i.e. know what you are signing).

ADDITIONAL INFORMATION

The information has been provided by <mailto:agantman@qualcomm.com> Alex Gantman.

=====

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: "[NT] Microsoft Internet Explorer Legacy Text Control Buffer Overflow"
 - **Messages sorted by:** [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#) [\[attachment \]](#)