ASPR-MOZ-1: Binary Planting in Mozilla Thunderbird (Import Outlook Express Address Book / wab32.dll)

General

Title:	Binary Planting in Mozilla Thunderbird (Import Outlook Express Address Book / wab32.dll)
Update ID:	ASPR-MOZ-1
Notification date:	6/23/2011
Contributors:	Simon Raner
Report type:	vendor ASPR

Summary

There is a Binary Planting vulnerability in Mozilla Thunderbird 3.1.11, allowing a remote, possibly Internet-based attacker to deploy malicious dynamic-link library (DLL) to a user's Windows machine and have it launched in the context of that user. In particular, Mozilla's `Thunderbird.exe` tries to load `%CommonProgramFiles%\System\wab32.dll` when a user tries to import an Outlook Express address book on Windows 7, but failing to find it in the search path provides an opportunity for the attacker to plant a malicious DLL in the current working directory.

ⓘ Problem type:	**Actual security problem**
ⓘ Discoverability:	**MEDIUM**
ⓘ Severity:	**VERY HIGH**

Affected Components

- Mozilla Thunderbird 3.1.10
- Mozilla Thunderbird 3.1.11

Other versions than the current one(s) or specified one(s) may also be vulnerable. Current versions are vulnerable on the following operating systems (no tests were done on others):

- Windows 7: YES
- Windows XP: NO

Demonstrations

Demonstration 1

1. Prepare a freshly installed and fully updated Windows 7 computer.
2. Install Mozilla Thunderbird on the system (we tested versions 3.1.10 and 3.1.11).
3. Create an empty folder `c:\temp` (if it doesn't already exist).
4. Place the attached `test.eml` in `c:\temp`.
5. Create a subfolder `c:\temp\%CommonProgramFiles%\System\` (use actual percent characters!) and place the attached `wab32.dll` in it.
6. In Windows Explorer, open `c:\temp`, then double-click the `test.eml` file. Mozilla Thunderbird gets launched.
7. In Thunderbird's Menu Bar, click on "Tools" and select "Import...".
8. In the displayed "Settings" window, select "Address Books" and click "Next" button, then select "Outlook Express" and click "Next" button.
9. As you do this, `c:\temp\%CommonProgramFiles%\System\wab32.dll` is loaded, which pops up a "HACKED" dialog.

Analysis

When launched by double-clicking an `.eml` file, Thunderbird is started with the current working directory (CWD) set to the folder where this file resides. If user tries to import an Address Book from Outlook Express on Windows 7 (this may not even make sense on Windows 7, but Thunderbird does provide an option to do so), Thunderbird makes an unsafe call to `LoadLibrary("%CommonProgramFiles%\System\wab32.dll")`. This DLL is not found on the system and is thus searched for in the search path and eventually in CWD.

The offending DLL path (`%CommonProgramFiles%\System\wab32.dll`) is obtained from the registry key `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WAB\DLLPath` and directly loaded with `LoadLibrary()` function by the Thunderbird code. This path is different on Windows XP and Windows 7:

- Windows XP: `C:\Program Files\Common Files\System\wab32.dll`
- Windows 7: `%CommonProgramFiles%\System\wab32.dll`

The `LoadLibrary()` function doesn't automatically resolve environment variables such as `%CommonProgramFiles%` but rather considers them as literal folder names. Since the `DLLPath` registry key only contains an environment variable on Windows 7 but not on Windows XP this vulnerability only affects Windows 7. (Note that we haven't tested other Windows operating systems.)

The attacker can mark the malicious DLL and the subfolders it is in as hidden so that per the default Windows Explorer settings, the user will not see it and is thus less likely to become suspicious.

Note that this also works when the `.eml` file and the malicious DLL are hosted on a network share in LAN (suitable for an internal attacker with access to a corporate network), and furthermore, even when stored on an Internet-based WebDAV server (suitable for a completely access-less attacker); most corporate firewalls are unlikely to stop outbound WebDAV (HTTP) traffic if they allow outbound HTTP.

More information on binary planting can be found here:

- The official web site of the binary planting vulnerability research
  http://www.binaryplanting.com
- ACROS Security blog, covering many aspects of binary planting
  http://blog.acrossecurity.com
- Online Binary Planting Exposure Test
  http://www.binaryplanting.com/test.htm
- A public list of some vulnerable applications
  http://secunia.com/advisories/windows_insecure_library_loading/

Error categories: **Other**

ⓘ Discoverability: **MEDIUM** Since this vulnerability does not reside in a frequently-used functionality, it can remain undetected for a long time.

Mitigating factors: -

Attack Scenarios

Attack Scenario 1: Deploying a malicious DLL to Windows workstations from the Internet

An external attacker sets up an anonymous WebDAV server somewhere on the Internet and places a number of interesting-sounding `.eml` files on it. Alongside these files, she places a hidden malicious `wab32.dll` in a hidden subfolder (see the above demonstration).

She then sends an e-mail to multiple users containing a hyperlink to a shared folder on the server, enticing the users to visit the share and open the `.eml` files. Upon user's clicking on the link, the content of the remote share is displayed in Windows Explorer. When a user double-clicks any one of the `.eml` files, Thunderbird gets launched and sets the current working directory to the remote share location. Now the tricky part for the attacker is to get the user to try to import an Address Book from Outlook Express.

One possible way to do that is by mimicking an error message inside the displayed remote `.eml` file, informing the user that the content of this e-mail can not be displayed due to an error with displaying the address of the sender, which will be resolved by the user importing an Address Book from Outlook Express. The error message also provides exact instructions for this. By doing so, the user unwittingly causes a malicious DLL to be "downloaded" and immediately executed on his computer.

ⓘ Impact:	VERY HIGH	Executing arbitrary malicious code on user's computer
ⓘ Required access: (likelihood)	VERY HIGH	The attacker needs no access to the user's environment
ⓘ Config. depend.:	MEDIUM	The exploitability of this vulnerability depends on (1) the Web Client service running on the user's computer (this service is running by default), (2) the network firewall allowing outbound WebDAV traffic (almost all do) and (3) the user's workstation is not actively protected by Microsoft's "CWDIllegalInDllSearch" countermeasure (unlikely). In addition, the user must be on a Windows 7 system.
ⓘ Simplicity:	LOW	This attack requires a heightened level of social engineering.
ⓘ Cost:	VERY LOW	This attack costs nothing
ⓘ Detectability:	VERY LOW	The attacker can make this attack virtually undetectable for the attacked user. Her malicious code can forward function calls to the original DLL, retaining the original functionality of the product.
ⓘ Traceability:	VERY LOW	It is very difficult to determine who placed some files on a network share. The attacker can also use network proxies and IP spoofing to hide her origin.
ⓘ Severity:	**VERY HIGH (4.1)**	

Recommendations

There are a number of options for fixing this issue or limiting its exploitability, assuming that the affected functionality is required at all:

1. Before calling `LoadLibrary()`, call `ExpandEnvironmentStrings()` to expand environment variables like `%CommonProgramFiles%` (more information are available at http://msdn.microsoft.com/en-us/library/ms724884%28v=vs.85%29.aspx).
2. Changing the current working directory to a safe location (e.g., to Thunderbird's home folder) immediately before the vulnerable call, and reverting it back immediately after it would fix this particular issue.
3. Calling `SetDllDirectory("")`, while generally a suitable solution for DLL-related binary planting problems, is severely limited by a known Windows bug which sometimes causes environment variables in the PATH unresolved (for more information, see http://blog.acrossecurity.com/2010/10/breaking-setdlldirectory-protection.html).
4. Up-to-date developer recommendations are available at http://www.binaryplanting.com/guidelinesDevelopers.htm

Binary planting issues have been found in almost every Windows application: our research (http://www.binaryplanting.com) has found 9 out of 10 applications vulnerable, frequently with multiple binary planting bugs.

Distribution

- Entered into Mozilla's Bugzilla on 6/23/2011

Contact

ACROS d.o.o.
Makedonska ulica 113
SI - 2000 Maribor, Slovenia

e-mail: security@acrossecurity.com
web: http://www.acrossecurity.com
phone: +386 2 3000 280
fax: +386 2 2000 282

PGP Key: http://www.acrossecurity.com/pgpkey.asc
PGP Fingerprint: FE9E 0CFB CE41 36B0 4720 C4F1 38A3 F7DD

ACROS Security Advisories: http://www.acrossecurity.com/advisories.htm
ACROS Security Papers: http://www.acrossecurity.com/papers.htm

Disclaimer

that use of these names means specific hosts, companies or individuals are vulnerable to any attacks nor does it mean that they consent to being used in any vulnerability tests. The use of information in this report is entirely at user's risk.

Glossary

Security Problem Type specifies whether the reported security issue currently presents a real risk to the product's users ("Actual Security Problem") or merely has the potential to evolve into such a risk in the future ("Potential Security Problem"). An example of the former is a buffer overflow vulnerability that can be remotely exploited for executing malicious code on the user's computer. An example of the latter is the same buffer overflow vulnerability which is only present in the debug build of the product: the vulnerable code might get copied to release parts of the code in the future, or a debug build could be mistakenly dispatched to the users.

Discoverability defines the likelihood that an independent third party will discover the vulnerability in the foreseeable future using publicly obtainable information about the product (including its source code in case of an open-source product). Following the worst-case principle, it is assumed that whoever should discover this vulnerability, would either announce it to the public (e.g. on security-related mailing lists) before a fix was available, thus causing damage to vendor's reputation and putting users at risk, or enable its covert exploitation, again causing damage to vendor's customers and reputation. Discoverability is quantified between 1 (very low) and 5 (very high), where 1 means "very difficult to discover" and 5 means "trivial to discover."

Impact is the "worst-case scenario" assessment of the damage the attacker could cause by exploiting the vulnerability, regardless of what special conditions would have to be met in order for this scenario to become possible. Impact is quantified between 1 (very low impact) and 5 (very high impact).

Required access refers to the access the attacker would need to obtain in order to be able to exploit a particular vulnerability. It defines the "likelihood" of attacker actually obtaining such access, and is quantified between 1 (very low) and 5 (very high) where 1 means "very few attackers could gain such access" and 5 means "every motivated attacker could gain such access".

Configuration dependence is an assessment of the likelihood that a specific configuration described in the attack scenario would be used in a real-world production system. For example, if the attack scenario requires the usernames to be of a specific, unlikely form (e.g., containing question marks), or that two servers must have IP addresses with matching last octet (which is a 1:255 chance in average), the configuration dependence would be very high. On the other hand, if the attack scenario assumes a default configuration or a configuration proposed by the product documentation, the configuration dependence would be very low. Configuration dependence is quantified between 1 (very low) and 5 (very high), and relates closely to the "Number of users affected" metric used by some other vulnerability assessment methods.

Simplicity defines how likely it would be for an attacker to actually carry out a successful attack by exploiting the vulnerability, given the required access to the target system. This attribute also includes the difficulty of crafting any special exploit tools, using special hardware or other equipment and performing social engineering on users. Simplicity is quantified between 1 (very low) and 5 (very high), where 1 means "very difficult to exploit" and 5 means "very easy to exploit."

Cost simply describes the estimated cost of a particular attack for the attacker (per the attack scenario) - in money and other resources. A very high cost, for example, would be assigned to an attack that required cracking a DES key - although this has been proven to be possible, it still requires special costly equipment. An attack that requires sending a billion requests to a web server over a telephone line would be assigned a high cost due to communication expenses. However, most of the attacks are usually quite inexpensive, thus having "very low" or "low" cost values. Cost is quantified between 1 (very low) and 5 (very high).

Traceability defines how likely a security-trained system administrator would be able to trace the origin of the attack, once it has been detected, and therefore also defines the attacker's exposure to system owner's legal or administrative retribution. Traceability is higher when the attack leaves traces of attacker's IP address, her username, targeted information, etc. on the attacked system. Following the worst-case assumption, any external logging facilities (e.g. firewall appliances, logging routers etc.) and non-default local logging facilities are not considered in the assessment. Traceability is quantified between 1 (very low) and 5 (very high), where 1 means "very difficult to trace" and 5 means "very easy to trace."

Severity is calculated with formula: **Imp** * **AVG** (**Acc** , (6 - **Conf**) , **Simp** , (6 - **Cost**) , (6 - **Det**) , (6 - **Trc**))/5, where **Imp** = Impact, **Acc** = Required access, **Conf** = Configuration dependence, **Simp** = Simplicity, **Cost** = Cost, **Det** = Detectability, **Trc** = Traceability and **AVG** = average of arguments. Rationale behind this formula is the following: (1) Impact affects the severity proportionally: if impact were 0, the severity would also be 0 regardless of other factors; (2) Required Access, Configuration dependence, Simplicity, Cost, Detectability and Traceability are "softer" attributes, more prone to subjective opinions. Therefore their average is calculated, normalized and used as a proportional factor in severity formula. Note that Configuration dependence, Cost, Detectability and Traceability need to be inverted (subtracted from 6) in order to correctly contribute to severity. According to the possible values of each attribute used in the formula the resulting severity can have a value between 0 and 5. We round this value to one decimal place to provide meaningful differentiation between severities of the discovered vulnerabilities. Note that the resulting severity is merely an orientation source: the assessments of input values are based on our team's experience, knowledge and expectations and would almost certainly be somewhat different were they evaluated by someone else.