Pidgin

CVE-2014-3694

Context

Plug-ins: GnuTLS SSL/TLS and OpenSSL SSL/TLS plugin. It occurs during the handling of X.509 certificates from SSL servers.

Problem

Both of libpurple's bundled SSL/TLS plugins (one for GnuTLS and one for NSS) failed to check that the Basic Constraints extension allowed intermediate certificates to act as CAs. This allowed anyone with any valid certificate to create a fake certificate for any arbitrary domain and Pidgin would trust it.

Solution

Both bundled plugins were changed to check the Basic Constraints extension on all intermediate CA certificates.

Codes

"Not checking for proper certificate validation", "Spoofing"

CVE-2013-6483

Context

XMPP protocol plugin (when handling spoofed replies)

Problem

The XMPP protocol plugin failed to ensure that iq (Instant Messaging Intelligence Quotient) replies came from the person they were sent to. A remote user could send a spoofed iq reply and attempt to guess the iq id. This could allow an attacker to inject fake data or trigger a null pointer dereference.

Solution

Keep track of the 'to' when sending an iq stanza and make sure replies for a given stanza ID come from the same address it was sent to.

Codes

"Application crash", "Spoofing", "Incorrect origin check", "Inject fake data"

CVE-2013-0271

Context

MXit protocol plugin in libpurple (when handling invalid file paths)

Problem

The MXit protocol plugin saves an image to local disk using a filename that could potentially be partially specified by the IM server or by a remote user, which could be used to create malicious files or overwrite files of the user.

Solution

Escape values that come from the network before using them in filenames.

Codes

"File path traversal", "Overwrite files"

CVE-2012-6152

Context

Yahoo! protocol plugin in libpurple (during validation of incoming strings)

Problem

Many places in the Yahoo! protocol plugin assumed incoming strings were UTF-8 and failed to transcode from non-UTF-8 encodings. This can lead to a crash when receiving strings that aren't UTF-8.

Solution

Depending on the context, either validate that a string is UTF-8 or transcode the string from the appropriate encoding to UTF-8.

Codes

"Application crash", "Unsanitized data", "Assuming encoding of incoming data is UTF-8"

CVE-2012-1178

Context

A flaw was found in the way the Pidgin MSN protocol plug-in processed text that was not encoded in UTF-8. A remote attacker could use this flaw to crash Pidgin by sending a specially-crafted MSN message.

Problem

In some situations the MSN server sends text that isn't UTF-8 encoded, and Pidgin fails to verify the text's encoding. In some cases this can lead to a crash when attempting to display the text.

Solution

Verify that incoming text is UTF-8, and sanitize if it's not.

Codes

"Application crash", "Sanitizing data", "Assuming encoding of incoming data is UTF-8"

CVE-2011-4603

Context

SILC protocol plugin in libpurple during validation of incoming messages.

Problem

When receiving various incoming messages, the SILC protocol plugin failed to validate that a piece of text was UTF-8. In some cases invalid UTF-8 data would lead to a crash. This vulnerability is similar to CVE-2011-3594, but occurs in a different piece of code and was fixed at a later date.

Solution

Validate incoming strings as UTF-8 before using them as such.

Codes

"Application crash", "Perform security check on unsanitized data", "Assuming encoding of incoming data is UTF-8"

CVE-2011-4601

Context

Oscar protocol plugin in libpurple during validation of incoming messages.

Problem

When receiving various messages related to requesting or receiving authorization for adding a buddy to a buddy list, the oscar protocol plugin **failed to validate that a piece of text was UTF-8**. In some cases invalid UTF-8 data would lead to a **crash**.

Solution

Validate incoming strings as UTF-8 before using them as such.

Codes

"Application crash", "Sanitizing data", "Assuming encoding of incoming data is UTF-8"

CVE-2011-3594

Context

SILC protocol plug-in when **handling non-UTF8 strings** using the glib2.

Problem

When receiving various incoming messages, the SILC protocol plugin **failed to validate that a piece of text was UTF-8**. In some cases invalid UTF-8 data would lead to a **crash.** A flaw was reported [1] in libpurple's SILC protocol plugin, and all software which uses SILC via libpurple. The g_markup_escape_text() function, when called on strings that **have not been verified as valid UTF-8**, will read past the end of the string and eventually **segfault** for certain sequences in some versions of Glib2. The behaviour of this function was undefined, and because it depends on the particular version of Glib2 in use, it is unknown what the complete ramifications of the flaw is, however it has been verified that an untrusted user could **remotely crash a libpurple** client via specially crafted SILC messages. The crash is **caused by passing "user-controlled" non-UTF8 string** to the g_markup_escape_text function. A non-utf8 string passed to g_markup_escape_text causes the same string to be passed along to append_escaped_text in gmarkup.c append_escaped_text is supposed to parse this text and uses g_utf8_next_char to read the entire input string (assuming that it is utf8 of-course). g_utf8_next_char returns invalid pointers which causes "while (p != end)" loop in append_escaped_text to never exit. This ultimately causes OOB read and eventual **client crash**.

Solution

Validate incoming strings as UTF-8 before using them as such.

Codes

"Application crash", "Assuming encoding of incoming data is UTF-8"

CVE-2011-2943

Context

IRC protocol plug-in when **processing invalid nicknames**

Problem

A NULL pointer dereference flaw was found in the way IRC protocol plug-in of the Pidgin multiprotocol instant messaging client processed certain nick names, when list set of users (/who command) was issued upon user session startup and connecting user has had certain **encoding configuration** setup. A remote attacker could use a specially-crafted string as their nickname to cause the Pidgin client on the side of the victim (connecting user) **to crash**.

Certain characters in the nicknames of IRC users can trigger a null pointer dereference in the IRC protocol plugin's handling of responses to WHO requests. This can cause a crash on some operating systems. Clients based on libpurple 2.8.0 through 2.9.0 are affected.

Solution

Change libpurple to **validate the data it receives from the server** before attempting to use it.

Codes

"Application crash", "Check object is not null", "Perform security check on unsanitized data"

CVE-2010-3088

Context

Pidgin-knotify plug-in flaw when **processing incoming messages**

Problem

pidgin-knotify is a pidgin plugin that displays received messages and other notices from pidgin as KDE notifications. It uses system() to invoke ktdialog and **passes the unescaped messages as command line arguments**. An attacker could use this to **inject arbitrary commands** by sending a prepared message via any protocol supported by pidgin to the victim.

Reproducible: Always

Steps to Reproduce:

1. Install and enable pidgin-knotify

2. Receive a message like ';touch /tmp/vulnerable;'

3. Confirm that /tmp/vulnerable exists

Actual Results: /tmp/vulnerable exists

Expected Results: The touch command should not be run.

The vulnerable system() call is located in src/pidgin-knotify.c, line 71-74: command = g_strdup_printf("kdialog --title '%s' --passivepopup '%s' %d", title, body, timeout); [...] result = system(command);

Solution

Instead of using system(), functions of the **exec family should be used**, e.g. **execve with a sanitized environment**. If a dbus interface for showing notifications in KDE exists, it could be used as well. I've written a patch some time ago to remove system() and instead use dbus, and upstream has given me access to the repository so I was planning to release a new version with that when RL shit happened and all my free time went to hell

Codes

"Arbitrary code execution", "Unsanitized data"

CVE-2010-0013

Context

MSN protocol plugin in libpurple when **processing emoticon messages**

Problem

Directory traversal vulnerability in slp.c in the MSN protocol plugin in libpurple in Pidgin 2.6.4 and Adium 1.3.8 allows remote attackers to **read arbitrary files** via a .. (dot dot) in an application/x-msnmsgrp2p MSN emoticon (aka custom smiley) request, a related issue to CVE-2004-0122.

NOTE: it could be argued that this is resultant from a vulnerability in which an emoticon download request is processed even without a preceding text/x-mms-emoticon message that announced availability of the emoticon.

Solution

Remove ~/.purple/custom_smiley/ directory if it exists. The directory is not created by default and is created when first custom smiley is defined.

Codes

"File path traversal", "Data leakage"