[<prev] [next>] [thread-next>] [day] [month] [year] [list]

```
Date: Sat, 5 Jun 2021 02:55:10 +0200
From: Marek Marczykowski-Górecki <marmarek@...isiblethingslab.com>
To: oss-security@...ts.openwall.com
Subject: XScreenSaver 5.45: Disconnecting a video output can cause
 XScreenSaver to crash and unlock
```

Summary
========

XScreenSaver is the default screen locker in dom0. It tracks which video
outputs are connected to the system in order to blank them properly. In
some specific hardware configurations, disconnecting an output can cause
XScreenSaver to crash, leaving the screen unlocked.

The issue affects XScreenSaver 5.45 only.

Impact
=======

On hardware configurations with more than 10 video outputs that can be
disconnected, an attacker with physical access to a screen-locked system
may be able to unlock it by physically disconnecting one or more
outputs, bypassing standard screen lock authentication.

Details
========

On X11, screen locking and blanking is done by creating a window that
obscures the whole screen, which is a standard practice. In
XScreenSaver, each such window is assigned a specific property. When a
video output is disconnected, its corresponding blanking window is
destroyed, and its XScreenSaver-specific property is removed so that it
will not be used by `xscreensaver-command` anymore. This is handled by
the `update_screen_layout()` function in the `driver/screens.c` file:

```
     985 /* Synchronize the contents of si->ssi to the current state of the monitors.
     986    Doesn't change anything if nothing has changed; otherwise, alters and
     987    reuses existing saver_screen_info structs as much as possible.
     988    Returns True if anything changed.
     989  */
     990 Bool
     991 update_screen_layout (saver_info *si)
     992 {
     993   monitor **monitors = scan_monitors (si);
     994   int count = 0;
     995   int good_count = 0;
    ...
    1009   while (monitors[count])
    1010     {
    1011       if (monitors[count]->sanity == S_SANE)
    1012         good_count++;
    1013       count++;
    1014     }
    1015
    1016   if (si->ssi_count == 0)
    1017     {
    1018       si->ssi_count = 10;
    1019       si->screens = (saver_screen_info *)
    1020         calloc (sizeof(*si->screens), si->ssi_count);
    1021     }
    1022
    1023   if (si->ssi_count <= good_count)
    1024     {
    1025       si->ssi_count = good_count + 10;
    1026       si->screens = (saver_screen_info *)
    1027         realloc (si->screens, sizeof(*si->screens) * si->ssi_count);
    1028       memset (si->screens + si->nscreens, 0,
    1029               sizeof(*si->screens) * (si->ssi_count - si->nscreens));
    1030     }
    ...
    1092   for (; j < count; j++)
    1093     {
    1094       saver_screen_info *ssi = &si->screens[j];
    1095       if (!ssi->screensaver_window)
    1096         continue;
    1097       fprintf (stderr, "%s: %d: screen now unused, disabling.\n",
    1098                blurb(), j);
    1099       /* Undo store_saver_id() so that xscreensaver-command doesn't attempt
    1100          to communicate with us through this window. It might make more
    1101          sense to destroy the window, but I'm not 100% sure that there are
    1102          no outstanding grabs on it that have yet been transferred.
    1103        */
    1104       XDeleteProperty (si->dpy, ssi->screensaver_window,
    1105                        XA_SCREENSAVER_VERSION);
    1106     }
```

The initial portion of the function counts how many outputs are defined
(the `count` variable) and how many of them are connected (the
`good_count` variable). Then, the `si->screens` array is allocated or
re-allocated to fit information about connected outputs, with an extra
margin of 10 entries. However, the loop at the end iterates over the
array up to the total number of outputs, not just the ones that are
connected.

If there are 10 or fewer disconnected outputs, this works fine. However,
if there are more than 10, it will access the array beyond its end,
reading unrelated data from memory. It will interpret this data as an
XScreenSaver window ID. If that unrelated data happens to be non-zero
(which is very likely), then the condition at line 1095 will not skip
it, and the `XDeleteProperty` call will operate on that (most likely
invalid) window ID. This, in turn, will cause the XScreenSaver process
to crash, as that's what the error handler is programmed to do (the
`saver_ehandler()` function in the `driver/xscreensaver.c` file).

The error message will look like this:

```
    #############################################################################

    xscreensaver: 11:17:59: X Error!  PLEASE REPORT THIS BUG.
    xscreensaver: 11:17:59: screen 0/0: 0x2ae, 0x0, 0x6600001
    xscreensaver: 11:17:59: screen 0/1: 0x2ae, 0x0, 0x0

    #############################################################################

    X Error of failed request:  BadWindow (invalid Window parameter)
      Major opcode of failed request:  19 (X_DeleteProperty)
      Resource id in failed request:  0x188dba0
      Serial number of failed request:  4284
      Current serial number in output stream:  4286

    ######################################################################
```

The issue affects only XScreenSaver version 5.45. Versions 5.44 and
older, as well as 6.00, are not affected. The XScreenSaver author was
notified about this issue and decided not to publish an advisory, as the
issue does not affect the most recent version.

The Qubes Security Team has decided to address this issue in Qubes OS by

```
patching this specific bug rather than immediately upgrading to the 6.00
version. The reason is that XScreenSaver 6.00 is a major update with
major architectural changes. As such, it poses an increased risk of
introducing unrelated problems. However, this decision does not preclude
the possibility of updating to XScreenSaver 6.00 at some point in the
future, independently of this particular security patch.

Credits
========

The issue was reported by Mustafa Kuscu:
https://github.com/QubesOS/qubes-issues/issues/6595


This is mostly repost of Qubes Security Bulletin 068 (with minor edits),
as it may be relevant to other distributions:
https://github.com/QubesOS/qubes-secpack/blob/master/QSBs/qsb-068-2021.txt


-- 
Best Regards,
Marek Marczykowski-Górecki
Invisible Things Lab
```

Download attachment "signature.asc**" of type "**application/pgp-signature**" (489 bytes)**