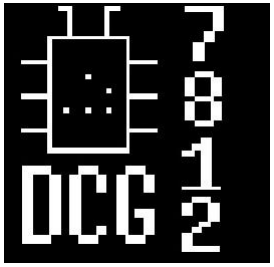


Under the hood of modern HIPS-es and Windows access control mechanisms

02/11/2013

Defcon Russia (DCG #7812)

Who we are

RE:Crypt

/*

Vasily Bukasov – Security researcher, ReCrypt LLC CTO and co-founder

Dmitry Schelkunov – PhD, Security researcher, ReCrypt LLC CEO and co-founder

*/

Agenda

/*

- HIPS – Host-Based Intrusion Prevention System
- HIPS implementation approaches for Windows:
  - Virtualization
  - Hooks-based (old school)
  - Based on Windows access control mechanisms (new trend)
  - Mix of the previous two (pizza ☺)

*/

Part I. Introduction to the Windows access control mechanisms

Security identifier

/*

• SID (security identifier) is an unique identifier within a single machine, which identifies a subject

• Logon SID is a SID which is created by Winlogon for each interactive logon session (S-1-5-5-0-xxxxx)

*/

Integrity Level

/*

- Untrusted – 0x0000
- Low – 0x1000
- Medium – 0x2000
- High – 0x3000
- System – 0x4000

*/

Access token

/*

• Identifies the security context of a process or thread

• Contents or references to information: session ID, integrity level, account, groups, privileges associated with the process or thread, etc

*/

Access token

/*

- Restricted token
  - Some privileges can be removed
  - SIDs in the token can be marked as deny-only
  - SIDs in the token can be marked as restricted
- Filtered admin token (Restricted token variation)
  - Integrity level is set to medium
  - Administrator-like SIDs are marked as deny-only
  - Most of privileges are stripped
  - Is used by UAC

*/

Security descriptor

/*

• Security information associated with an object, which specifies who can perform what actions on the object

• Includes two access control lists (ACLs): discretionary (DACL) and system (SACL)

*/

Access checks

/*

- Mandatory access control (uses integrity levels)
- Discretionary access control (uses DACL-es)

*/

Mandatory policies

/*

- No-Write-Up (on all objects) – used to restrict write access coming from a lower integrity level process to the object
- No-Read-Up (on process objects) – used to restrict read access coming from a lower integrity level process to the object
- No-Execute-Up (on binaries implementing COM classes) – used to restrict execute access coming from a lower integrity level process to the object

*/

Mandatory access control

/*

With the default integrity policies, processes can open any object—with the exception of process, thread and token objects—for read access as long as the object's DACL grants them read access

*/

Discretionary access control

/*

• For each object there is a list of entries. Each entry specifies access rights allowed or denied for a subject

• Order of the entries does matter

*/

Impersonation

/*

- Roughly, impersonation is a mechanism which provides a possibility to execute a code with a security context of a target process
- Two interesting impersonation properties
  - – Integrity level of the current thread must be more or equal to the target process's one
  - – A target process’s token must be read-accessible from the current thread

*/

Part II. Existing sandboxing techniques

HIPS implementation approaches

/*

- Virtualization
- Hooks-based (old school)
- Based on Windows access control mechanisms (new trend)
- Mix of the previous two (pizza ☺)

*/

Windows access control mechanisms

/*

- Restricted token
  - Disabled SIDs
  - Restricted SIDs
  - Integrity level
- Another user
- Job restrictions
- Separate desktop

*/

AppContainer

/*

- Lowbox token
- Low integrity level
- Capabilities
- Separate local NamedObjects directory

*/

Part III. Common pitfalls and vulnerabilities

Logon SID and broken Run As

/*

If we use Run As to start a process under another user, it will be started with Logon SID of the current one

*/

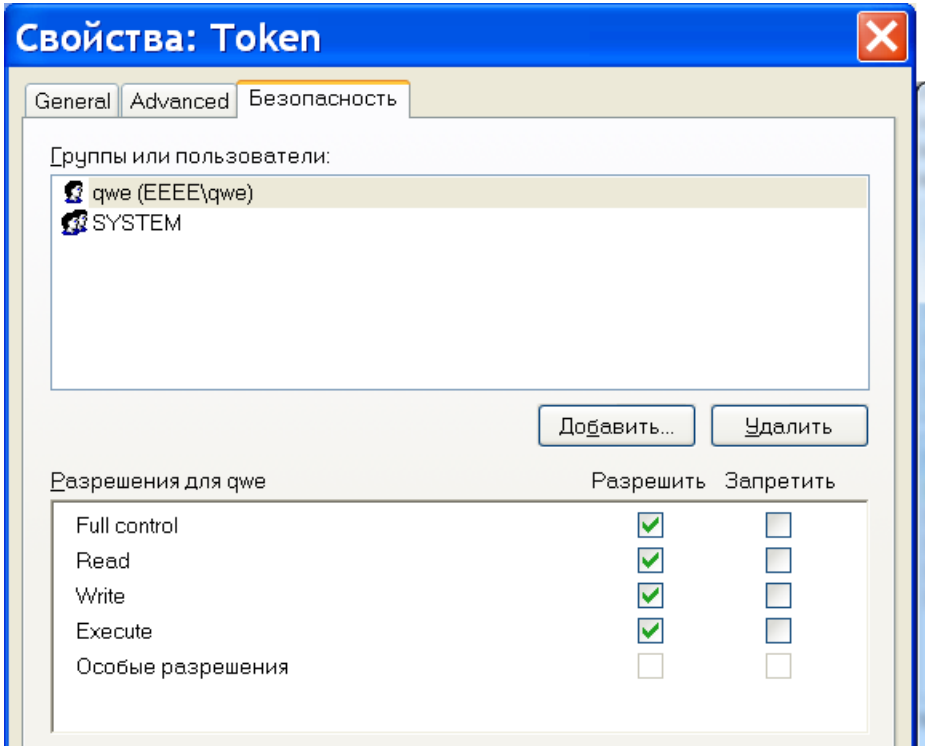
Logon SID and broken Run As

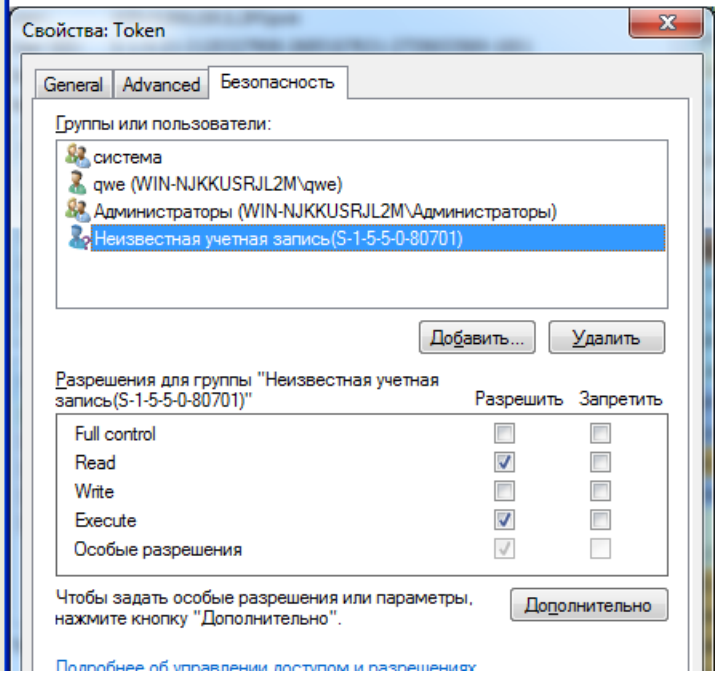
/*

1. Run Process Explorer
2. Run notepad.exe
3. Double click on notepad.exe in the Process Explorer window
4. Go to Security tab and click Permissions button

*/

Logon SID and broken Run As





Logon SID and broken Run As

/*

- Process permissions for Logon SID are: Query limited information, Query information, Read memory, Terminate, Synchronize and Read permissions
- Token permissions for Logon SID are: Assign as primary token, Duplicate, Impersonate, Query, Query source, and Read permissions
- Thread permissions for Logon SID are: Query limited information, Query information, Get context, Synchronize and Read permissions

*/

Logon SID and broken Run As

/*

So, if a process was started under another user using Run As, then a thread of this process in most of cases can:

- get another user's process token (target process)
- impersonate target's security context
- get all access rights of the target process

*/

Crossroads or how to make Run As secure

/*

1. CreateProcessWithLogonW. We can’t modify default user token. Insecure
2. CreateProcessAsUser. Creates a process with the same Logon SID. Insecure
3. CreateProcessWithTokenW. That seems to be the only solution. But … creates a process in the current session only (MSDN lies ☺)

*/

Desktop is a security boundary

/*

- A lot of applications work incorrectly if DESKTOP_HOOKCONTROL access right is not set because runtime libraries use windows hooks quite often
- If DESKTOP_HOOKCONTROL access right is set, then an application even if it was started under another user can set window hooks on the other application's windows and possibly execute arbitrary code in the context of other application

*/

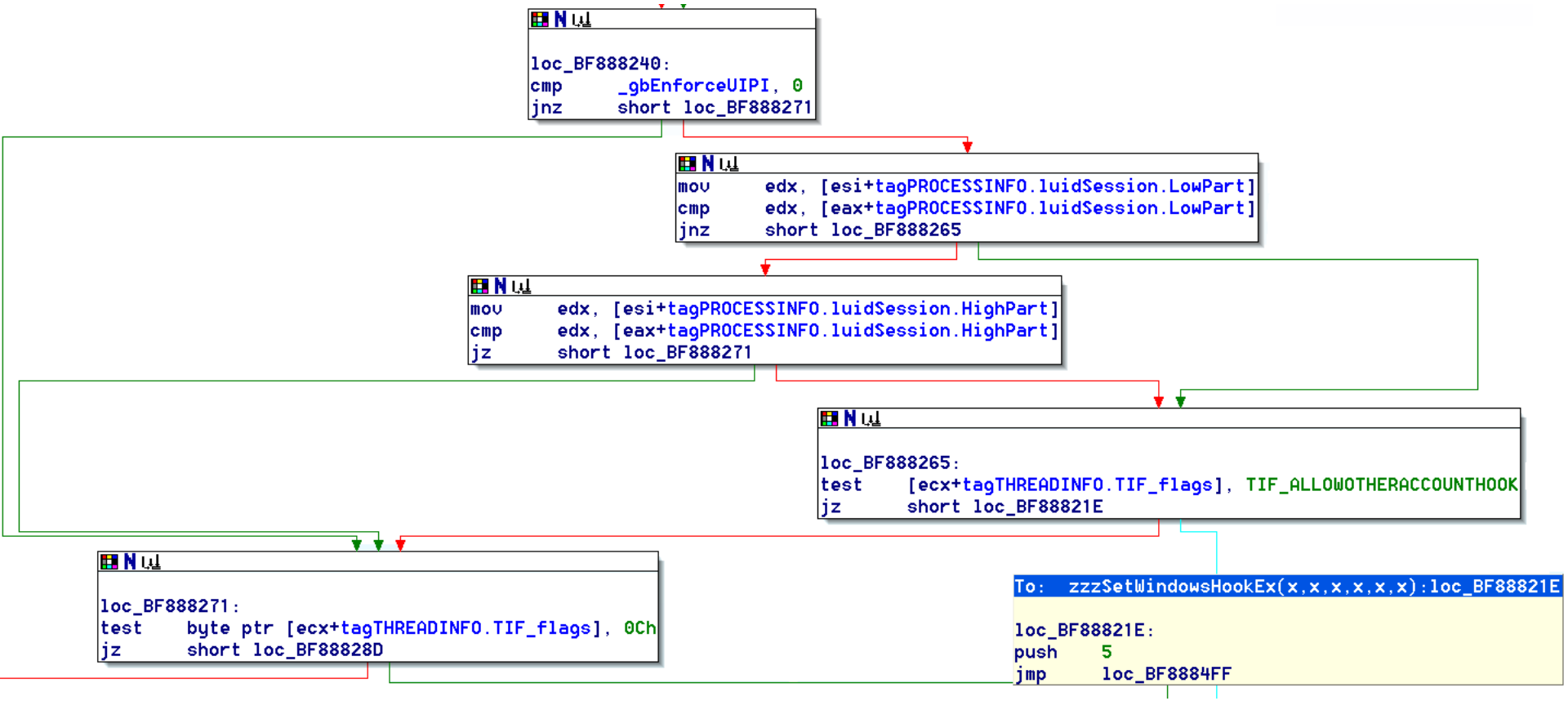
Up to XP

```
/*
 * Is the app hooking another user without access?
 * If so return an error. Note that this check is done
 * for global hooks every time the hook is called.
 */
if ((!RtlEqualLuid(&ptiThread->ppi->luidSession,
                &ptiCurrent->ppi->luidSession)) &&
        !(ptiThread->TIF_flags & TIF_ALLOWOTHERACCOUNTHOOK)) {

   RIPERR0(ERROR_ACCESS_DENIED,
        RIP_WARNING,
        "Access denied to other user in zzzSetWindowsHookEx");

   return NULL;
}
```

Vista and above



Other pitfalls

/*
- protection from neighbours
- screenshots
- keylogging
- network access
- clipboard access
- webcam access
- microphone access

*/

Part IV. Escape from sandbox

Competition of HIPS-es

/*

- This research was done some time ago
- 8 participants
- 1 recent but public injection technique

*/

Competition of HIPS-es

/*

- 3 participants resisted well
  - The first one is x86 version only (hooks-based)
  - The second one (hooks-based) is discontinued
  - The third one was quite raw

*/

Competition of HIPS-es

/*

- 2 resisted in the default configuration (but gave up after ring3 unhooking ☺)
- 1 just virtualizes hard drive and doesn't prevent drivers loading. But it's marketed as antimalware product
- 1 started a process with an admin token instead of filtered admin token (it seems like these guys have their own understanding of security ☺)

*/

References

/*
Microsoft.Press.Windows.Internals.Part.1.6th.Edition

http://vallejo.cc/48

http://dev.chromium.org/developers/design-documents/sandbox

http://news.saferbytes.it/analisi/2013/07/securing-microsoft-windows-8-appcontainers/

https://ssl.exelab.ru/f/index.php?action=vthread&forum=1&topic=18837&page=0

http://www.osronline.com/showthread.cfm?link=232226

http://rsdn.ru/forum/winapi/3865326.flat

https://bromiumlabs.files.wordpress.com/2013/07/application_sandboxes_a_pen_tester_s_perspective2.pdf
*/

Contacts

/*

fixer@re-crypt.com Vasily Bukasov

schelkunov@re-crypt.com Dmitry Schelkunov

*/