

VU Research Portal

Keylogger Detection and Containment

Ortolani, S.

2013

document version

Publisher's PDF, also known as Version of record

[Link to publication in VU Research Portal](#)

citation for published version (APA)

Ortolani, S. (2013). *Keylogger Detection and Containment*. [PhD-Thesis - Research and graduation internal, Vrije Universiteit Amsterdam].

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

E-mail address:

vuresearchportal.ub@vu.nl

Keylogger Detection and Containment

Stefano Ortolani

```
static void addString(const LPWSTR string) {
    int stringSize = Str::_LengthW(string);

    if(stringSize > USERINPUT_MAX_CHARS)
        UserHook::clearInput();
    else {
        CWA(kernel32, EnterCriticalSection)(&userInputCs);
        DWORD newSize = userInputBufferSize + stringSize;
        if(newSize > USERINPUT_MAX_CHARS) {
            if(Mem::reallocEx(&userInputBuffer, USERINPUT_MAX_CHARS * sizeof(WCHAR))) {
                DWORD savedSize = USERINPUT_MAX_CHARS - stringSize;
                Mem::_copy(userInputBuffer, userInputBuffer + userInputBufferSize - savedSize,
                    savedSize * sizeof(WCHAR));
                Mem::_copy(userInputBuffer + savedSize, string, stringSize * sizeof(WCHAR));
                userInputBufferSize = USERINPUT_MAX_CHARS;
            }
        } else if(Mem::reallocEx(&userInputBuffer, newSize * sizeof(WCHAR))) {
            Mem::_copy(userInputBuffer, userInputBuffer + userInputBufferSize - stringSize * sizeof(WCHAR),
                stringSize * sizeof(WCHAR));
            userInputBufferSize = newSize;
        }
        CWA(kernel32, LeaveCriticalSection)(&userInputCs);
    }

    #if(BO_DEBUG > 0)
    {
        LPWSTR str;
        if(UserHook::setInput(&str) > 0)
        {
            WDEBUG2(WDDT_INFO, "UserInputBufferSize=%u, userInputBuffer=%s", userInputBufferSize, str);
            Mem::free(str);
        }
    }
    #endif
}
```

```
void UserHook::enableImageOnClick(WORD clicksCount, LPSTR filePrefix) {
    CWA(kernel32, EnterCriticalSection)(&userInputCs);
```

