Run-time Thread Injection
The Jugaad way

By Aseem Jakhar

$whoami

- Security and open source enthusiast.
- Founder – null The open security community.
- Organizer – nullcon security conference.
- Chief researcher – Payatu Technologies
- Speaker at various security | open source conferences
  - Defcon, Blackhat, Xcon, Gnunify, ISACA Blore, Cocon, Clubhack, Blore Cyber security Summit.

Disclaimer

- All views and ideas expressed in the presentation are mine and do not reflect that of my employer or null.

Disclaimer

- All views and ideas expressed in the presentation are mine and do not reflect that of my employer or null.
- For educational purpose only. Use it at your own risk.

Disclaimer

- All views and ideas expressed in the presentation are mine and do not reflect that of my employer or null.
- For educational purpose only. Use it at your own risk.
- If you use it in the field please don't send me a thanks email. I don't know you.

Agenda

- Motivation
- Code Injection
- Windows
- Linux
- ptrace() Primer
- Library Injection
- Jugaad
- Conclusion

Motivation

- Windows malware
- Ease of injection
- Stealthy

Agenda

- Motivation
- **Code Injection**
- Windows
- Linux
- ptrace() Primer
- Library Injection
- Jugaad
- Conclusion

Code Injection

- Injecting executable instructions/code.
- Altering the default flow of execution.
- RCE via Buffer overflow
- SQL Injection
- Cross site scripting
- XML Injection

Code Injection

- Injecting executable instructions/code.
- Altering the default flow of execution.
- RCE via Buffer overflow
- SQL Injection
- Cross site scripting
- XML Injection
- APIs

Agenda

- Motivation
- Code Injection
- **Windows**
- Linux
- ptrace() Primer
- Library Injection
- Jugaad
- Conclusion

Windows

- Allows code injection via a defined API.

Windows

- Allows code injection via a defined API.
- CreateRemoteThread and family.

Windows

- Allows code injection via a defined API.
- CreateRemoteThread and family.
- HANDLE WINAPI CreateRemoteThread(

  __in HANDLE **hProcess**,

  __in LPSECURITY_ATTRIBUTES lpThreadAttributes,

  __in SIZE_T **dwStackSize**,

  __in LPTHREAD_START_ROUTINE **lpStartAddress**,

  __in LPVOID lpParameter,

  __in DWORD dwCreationFlags,

  __out LPDWORD lpThreadId);

CreateRemoteThread

- hProcess – A handle to the process in which the thread is to be created.

- Source: http://msdn.microsoft.com/en-us/library/ms682437%28v=vs.85%29.aspx

CreateRemoteThread

- hProcess – A handle to the process in which the thread is to be created.
- dwStackSize – The initial size of the stack, in bytes.

- Source: http://msdn.microsoft.com/en-us/library/ms682437%28v=vs.85%29.aspx

CreateRemoteThread

- hProcess – A handle to the process in which the thread is to be created.
- dwStackSize – The initial size of the stack, in bytes.
- lpStartAddress – A pointer to the application-defined function to be executed by the thread and represents the starting address of the thread in the remote process. The **function must exist in the remote process**.

- Source: http://msdn.microsoft.com/en-us/library/ms682437%28v=vs.85%29.aspx

Agenda

- Motivation
- Code Injection
- Windows
- **Linux**
- ptrace() Primer
- Library Injection
- Jugaad
- Conclusion

Linux

- No well defined API/CreateRemoteThread equivalent.

Linux

- No well defined API/CreateRemoteThread equivalent.
- So how do we inject a thread/code?

Linux

- No well defined API/CreateRemoteThread equivalent.
- So how do we inject a thread/code?
- Wait a minute... Back to basics

Linux

- No well defined code injection support
- No CreateRemoteThread equivalent
- So how do we inject a thread/code?
- Wait a minute... Back to basics
- How does a debugger manipulate a debugee?

Linux

- No well defined code injection support
- No CreateRemoteThread equivalent
- So how do we inject a thread/code?
- Wait a minute... Back to basics
- How does a debugger manipulate a debugee?
- Awesomeness of ptrace()

Agenda

- Motivation
- Code Injection
- Windows
- Linux
- **ptrace() Primer**
- Library Injection
- Jugaad
- Conclusion

ptrace() primer

- Tracing API a.k.a Debugging.

ptrace() primer

- Tracing API a.k.a Debugging.
- Forced parenthood ;-)

ptrace() primer

- Tracing API a.k.a Debugging.
- Forced parenthood ;-)
- Gives caller the ability to read/write child process memory.

ptrace() primer

- Tracing API a.k.a Debugging.
- Forced parenthood ;-)
- Gives caller the ability to read/write child process memory.
- Gives caller the ability to change the flow of execution of child process

ptrace() primer

- Tracing API a.k.a Debugging.
- Forced parenthood ;-)
- Gives caller the ability to read/write child process memory.
- Gives caller the ability to change the flow of execution of child process.
- Gives caller the ability to start/stop the execution of the child process at will.

ptrace() primer

- Tracing API a.k.a Debugging.
- Forced parenthood ;-)
- Gives caller the ability to read/write child process memory.
- Gives caller the ability to change the flow of execution of child process.
- Gives caller the ability to start/stop the execution of the child process at will.
- Powerful API – Single function, multiple operations.

Prototype

- long ptrace(enum __ptrace_request *request*,

  pid_t *pid*,

  void **addr*,

  void **data*);
- request – The operation to be performed on the traced process.
- pid – The process identifier of the process being traced.
- addr and data – The values depend on the type of operation.

operations

- PTRACE_ATTACH - Attaches to the process specified in *pid.*
- PTRACE_CONT - Restarts the stopped child process.
- PTRACE_DETACH - Restarts the stopped child as for PTRACE_CONT, but first detaches from the process.

operations

- PTRACE_ATTACH - Attaches to the process specified in *pid.*
- PTRACE_CONT - Restarts the stopped child process.
- PTRACE_DETACH - Restarts the stopped child as for PTRACE_CONT, but first detaches from the process.
- PTRACE_PEEKTEXT - Reads a word at the location *addr* in the child's memory.
- PTRACE_POKETEXT - Copies the word *data* to location *addr* in the child's memory.

operations

- PTRACE_ATTACH - Attaches to the process specified in *pid.*
- PTRACE_CONT - Restarts the stopped child process.
- PTRACE_DETACH - Restarts the stopped child as for PTRACE_CONT, but first detaches from the process.
- PTRACE_PEEKTEXT - Reads a word at the location *addr* in the child's memory.
- PTRACE_POKETEXT - Copies the word *data* to location *addr* in the child's memory.
- PTRACE_GETREGS - Copies the child's general purpose to location *data* in the parent.
- PTRACE_SETREGS - Copies the child's general purpose or floating-point registers, respectively, from location data in the parent.

Control

- Getting the control back after executing specific instructions.

Control

- Getting the control back after executing specific instructions.
- Breakpoints Yeah!!!

Control

- Getting the control back after executing specific instructions.
- Breakpoints Yeah!!!
- Int3 instruction (0xcc)

Agenda

- Motivation
- Code Injection
- Windows
- Linux
- ptrace() Primer
- **Library Injection**
- Jugaad
- Conclusion

Library Injection

- Injecting shared objects into running processes.

Library Injection

- Injecting shared objects into running processes.
- Open source tool – injectSo.

Library Injection

- Injecting shared objects into running processes.
- Open source tool – injectSo.
- Read/write fds, intercept IO, functions. Basically do anything within the context of the victim process.

Library Injection

- Injecting shared objects into running processes.
- Open source tool – injectSo.
- Read/write fds, intercept IO, functions. Basically do anything within the context of the victim process.
- But wait.. Whats that in /proc...?

Proc maps

- Memory map after injection
- cat /proc/<pid>/maps

  00d74000-00f63000 r-xp 00000000 08:01 8698 /home/victim/evil.so

Demo

- InjectSO dummy shared object
- Executes shared object init() function which printf()s “Yo from init”
- Shared object visible in proc maps

Agenda

- Motivation
- Code Injection
- Windows
- Linux
- ptrace() Primer
- Library Injection
- **Jugaad**
- Conclusion

Jugaad

- What it is
  - Jugaad – Hindi word, means work-around/hack.
  - Open source thread injection kit.
  - In-memory injection. Stealthy.
  - Customizable payload.

Jugaad

- What it is
  - Jugaad – Hindi word, means work-around/hack.
  - Open source thread injection kit.
  - In-memory injection. Stealthy.
  - Customizable payload.
- What it is not
  - Exploit/Vuln/Zero day.
  - Privilege escalation.

Jugaad

- Memory Allocation and Execution
- Threadification
- Payload (Evil code)
- libjugaad API

Memory | Execution

- Backup predefined memory location – PEEKTEXT

Memory | Execution

- Backup predefined memory location – PEEKTEXT
- Backup registers – GETREGS

Memory | Execution

- Backup predefined memory location – PEEKTEXT
- Backup registers – GETREGS
- Overwrite with mmap2 shellcode (int3 appended) – POKETEXT

Memory | Execution

- Backup predefined memory location – PEEKTEXT
- Backup registers – GETREGS
- Overwrite with mmap2 shellcode (int3 appended) – POKETEXT
- Set EIP to point to the overwritten memory location – SETREGS

Memory | Execution

- Backup predefined memory location – PEEKTEXT
- Backup registers – GETREGS
- Overwrite with mmap2 shellcode (int3 appended) – POKETEXT
- Set EIP to point to the overwritten memory location – SETREGS
- Execute the shellcode – CONT

Memory | Execution

- Backup predefined memory location – PEEKTEXT
- Backup registers – GETREGS
- Overwrite with mmap2 shellcode (int3 appended) – POKETEXT
- Set EIP to point to the overwritten memory location – SETREGS
- Execute the code – CONT
- Get the control back – wait() (for int3 execution)

Memory | Execution

- Backup predefined memory location – PEEKTEXT
- Backup registers – GETREGS
- Overwrite with mmap2 shellcode (int3 appended) – POKETEXT
- Set EIP to point to the overwritten memory location – SETREGS
- Execute the code – CONT
- Get the control back – wait() (for int3 execution)
- Address of newly allocated memory - GETREGS

Stub mmap2 shellcode

```
"\x31\xdb"                  // xor %ebx,%ebx # Zero out ebx
"\xb9\x10\x27\x00\x00" // mov $0x2710,%ecx # memory size 10000 bytes
"\xba\x07\x00\x00\x00" // mov $0x7,%edx # page permissions R|W|E = 7
"\xbe\x22\x00\x00\x00" // mov $0x22,%esi #flags MAP_PRIVATE|
                            // MAP_ANONYMOUS
"\x31\xff"                  // xor %edi,%edi # Zero out edi
"\x31\xed"                  // xor %ebp,%ebp # Zero out ebp
"\xb8\xc0\x00\x00\x00" // mov $0xc0,%eax # mmap2 sys call no. 192
"\xcd\x80"                  // int $0x80  # s/w interrupt
"\xcc";                     // int3       # breakpoint interrupt
```

Memory | Execution

- Process independent memory allocated.
- Can be used to store malicious code and data.

Jugaad

- Memory Allocation and Execution
- **Threadification**
- Payload (Evil code)
- libjugaad API

Threadification

- Clone system call wrapper.
- int clone(int (***fn**)(void *),

  void ***child_stack**,

  int **flags**, void *arg, ...

  /* pid_t *ptid, struct user_desc *tls, pid_t *ctid */);
- fn – Function application to execute.
- child_stack – location of the stack used by the child process. Stack bottom (highest memory) address.
- flags – specify what is shared between the calling process and the child process.

Threadification

- Store clone shellcode in hidden memory.
- Execute the shellcode.
- Get the control back in the main thread
- The injected child thread starts execution independently of the ptrace() caller

Jugaad

- Memory Allocation and Execution
- Threadification
- **Payload (Evil code)**
- libjugaad API

Payload

- Custom payload.

Payload

- Custom payload.
- Thread aware.

Payload

- Custom payload.
- Thread aware.
- The payload is injected as a combined threading payload for relative addressing and jumping to thread code from the clone code.

Payload

- Custom payload.
- Thread aware.
- The payload is injected as a combined threading payload for relative addressing and jumping to thread code from the clone code.
- Kind of a sandwich shellcode.
- **[**CLONE_HEAD**] [**PAYLOAD**] [**CLONE_TAIL**]**
- CLONE_HEAD – clone syscall.
- PAYLOAD – The evil code.
- CLONE_TAIL – exit syscall.

Jugaad

- Memory Allocation and Execution
- Threadification
- Payload (Evil code)
- **libjugaad API**

Libjugaad API

- int create_remote_thread(pid_t pid,

  size_t stack_size,

  unsigned char * tpayload,

  size_t tpsize);

- The function takes care of injecting the code inside remote process unlike windows CreateRemoteThread

Libjugaad API

- For the Experimental Dudes:

```
int create_remote_thread_ex(pid_t pid,
                            int stack_size,
                            unsigned char * tpayload,
                            size_t tpsize,
                            int thread_flags,
                            int mmap_prot,
                            int mmap_flags,
                            void * bkpaddr);
```

Demo

- Payload
  - Standard TCP listener on port 4444
- Victim process
  - Firefox

Agenda

- Motivation
- Code Injection
- Windows
- Linux
- ptrace() Primer
- Library Injection
- Jugaad
- **Conclusion**

Conclusion

- Stealthy CreateRemoteThread now possible on Linux.
- Simple debugging functionality can be abused for injection purposes.
- Injecting library is not that stealthy, shared object name in maps file.
- Disable ptrace functionality in your Linux boxes

Future

- Add 64 bit support
- Add Anti-forensics
- Add AV bypass
- Add kernel and user mode rootkit suport
- A one stop shop for understanding malware techniques.

Source code

- http://null.co.in/2011/07/03/project-jugaad-2/
- https://github.com/aseemjakhar/jugaad
- If you would like to contribute to the project
  - Fork it in git
  - Add some cool stuff
  - Send me an email

References

- CreateRemoteThread: http://msdn.microsoft.com/en-us/library/ms682437(v=vs.85).aspx
- Needle (By skape): http://www.hick.org/code/skape/papers/needle.txt
- Linux ELF Format: http://asm.sourceforge.net/articles/startup.html
- memgrep: http://www.hick.org/code/skape/memgrep/
- Playing with ptrace Part 1 (By Pradeep Padala): http://www.linuxjournal.com/article/6100
- Playing with ptrace Part 2 (By Pradeep Padala): http://www.linuxjournal.com/article/6210
- InjectSo (By Shawn Clowes): http://www.secureReality.com.au/archives/injectso-0.2.1.tar.gz

Thanks

- QA?
- Contact: null at null.co.in