Jugaad
Linux Thread Injection Kit

By Aseem Jakhar

$whoami

- A Big Hello from India.
- Founder – null The open security community.
- Organizer – nullcon security conference.
- Chief researcher – Payatu Labs http://www.payatu.com
- Speaker at various security conferences
    - Blackhat, Xcon, Gnunify, ISACA Blore, Cocon, Clubhack, Blore Cyber security Summit.

null

- Registered Non-Profit organization.
- The largest security community in India.
- Focus – security research, knowledge sharing.
- 6 chapters in India.
- Monthly meets in all chapters.
- Security awareness camps.
- nullcon – The Favorite go-to destination for hackers and security professionals in the Indian sub-continent.

Agenda

- What is Jugaad
- What Jugaad is not
- Code Injection
- Windows
- Linux
- ptrace() Primer
- Library Injection
- Jugaad
- Conclusion

What is Jugaad

- Jugaad – Hindi word, means work-around/hack.
- Code injection technique.
- Threading capability.
- Customized payload.
- Jugaad in it's true sense.

Agenda

- What is Jugaad
- **What Jugaad is not**
- Code Injection
- Windows
- Linux
- ptrace() Primer
- Library Injection
- Jugaad
- Conclusion

What Jugaad is not

- Zero day.
- Vulnerability.
- Privilege escalation technique.

Agenda

- What is Jugaad
- What Jugaad is not
- **Code Injection**
- Windows
- Linux
- ptrace() Primer
- Library Injection
- Jugaad
- Conclusion

Code Injection

- Injecting executable instructions/code.
- Altering the default flow of execution.
- Buffer overflow
- SQLi
- XSS
- XML
- APIs

n|u

nullCON

Agenda

- What is Jugaad
- What Jugaad is not
- Code Injection
- **Windows**
- Linux
- ptrace() Primer
- Library Injection
- Jugaad
- Conclusion

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);
```

Windows

- 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

- What is Jugaad
- What Jugaad is not
- Code Injection
- Windows
- **Linux**
- ptrace() Primer
- Library Injection
- Jugaad
- Conclusion

Linux

- No remote code injection API.
- No CreateRemoteThread equivalent.
- How do we inject code into remote process?
- Wait a minute... what does gdb do?
- Awesomeness of ptrace().

Agenda

- What is Jugaad
- What Jugaad is not
- Code Injection
- Windows
- Linux
- **ptrace() Primer**
- Library Injection
- Jugaad
- Conclusion

n|u

ptrace() primer

- Tracing API a.k.a Debugging.
- Powerful API – Single function, multiple operations.
- 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.

ptrace() primer

- request parameter.
- 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() primer

- 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.

ptrace() primer

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

Agenda

- What is Jugaad
- What Jugaad is not
- Code Injection
- Windows
- Linux
- ptrace() Primer
- **Library Injection**
- Jugaad
- Conclusion

Library Injection

- Injecting shared libraries into running processes.
- Open source tool – injectSo.
- Awesome!!!
- Read/write fds, intercept IO, functions.
- But wait... Whats that in /proc ?

Library Injection

- cat /proc/1234/maps

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

Library Injection

Demo

Agenda

- What is Jugaad
- What Jugaad is not
- Code Injection
- Windows
- Linux
- ptrace() Primer
- Library Injection
- **Jugaad**
- Conclusion

Jugaad

- Thread injection kit.
- In-memory injection.
- Stealthier.
- No more library traces in maps.
- Awesomeness??
- Custom Payload.

Jugaad

- Memory Allocation and Execution
- Threadification
- Payload (Evil code)
- Implementation
- Demo

Jugaad

- **Memory Allocation and Execution**
- Threadification
- Payload (Evil code)
- Implementation details
- Demo

Jugaad

- Backup memory location and registers.
- Overwrite with shellcode.
- Set EIP to point to the overwritten memory location.
- Execute the code.
- Upon executing int3 instruction we get the control back.

Jugaad

- Allocate memory using mmap2 system call.
- void *mmap(void *addr,

  size_t **length**,

  int **prot**,

  int **flags**,

  int fd,

  off_t offset);
- length – Length of the mapping.
- prot – Desired memory protection of the mapping.
- flags – mapping specific flags.

Jugaad

- Sample 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
```

Jugaad

- Memory Allocation and Execution
- **Threadification**
- Payload (Evil code)
- Implementation
- Demo

Jugaad

- 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.

Jugaad

- Execute clone shellcode.
- Get the control back from the remote process in main thread by int3 instruction.
- The injected thread starts execution and becomes independent of the ptrace caller and the traced process main thread.

Jugaad

- Memory Allocation and Execution
- Threadification
- **Payload (Evil code)**
- Implementation
- Demo

Jugaad

- 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.
- [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)
- **Implementation**
- Demo

Jugaad

- **Shellcode**
- mmap2, clone, exit, evil code.
- Shellcode Stubs for mmap2 and clone.
- Actual shellcode generated on the fly based on caller requirements.

Jugaad

- struct shellcode {unsigned char * payload, size_t psize};
- struct shellcode * shellcode_mmap2(size_t length,

```
                                    int prot,
                                    int flags);
```

- struct shellcode * shellcode_thread(unsigned char * tpayload,

```
                                    size_t tpsize,
                                    void * child_stack,
                                    int flags);
```

Jugaad

- libjugaad API
- int create_remote_thread(pid_t pid,

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

Jugaad

- Memory Allocation and Execution
- Threadification
- Payload (Evil code)
- Implementation
- **Demo**

Agenda

- What is Jugaad
- What Jugaad is not
- Code Injection
- Windows
- Linux
- Ptrace() Primer
- Library Injection
- Jugaad
- **Conclusion**

Conclusion

- Stealthy CreateRemoteThread now possible.
- 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 via SELinux/apparmor.

Project details

- http://null.co.in/section/projects
- Version 1 contains 32 bit support.
- Next release will include 64 bit support, library injection (possibly without the trace in maps file).

Contribution

- null local Chapters.
- null projects.
- null Jobs – http://jobs.nullcon.net
- nullcon security conference – http://nullcon.net
- Mailing list – http://groups.google.com/group/null-co-in

Thanks

- ? & .
- Contact
  - aseemjakhar |at| gmail.com
  - null |at| null.co.in
- I'll be around if you feel like contributing to null or if you have any queries.