Never Lose a Syslog Message

Alexander Bluhm

bluhm@openbsd.org

September 24, 2017

Agenda

1 Motivation

2 Starting Position

3 Local Improvements

4 Remote Logging

5 Conclusion

Why reliable logging?

- system analysis
- attacker tries to prevent log
- required by common criteria

What can go wrong?

- UDP for remote logs
- UNIX datagram for local logs
- file descriptors
- chroot environment
- timestamps and time zones

Agenda

1. Motivation
2. Starting Position
3. Local Improvements
4. Remote Logging
5. Conclusion

Traditional Message Flow

program

```
syslog(LOG_ERR, "message %d", 7)
```

libc

priority, timestamp, sprintf, send

kernel

```
/dev/log
```

syslogd

recv, log file, send UDP

Priority, Facility, Level, Severity, Options

```
openlog("ftpd", LOG_PID|LOG_CONS, LOG_FTP)
syslog(LOG_INFO, "%s logged in", user)

#define LOG_FTP (11<<3) /* ftp daemon */
#define LOG_INFO 6 /* informational */

<94>Sep 24 09:35:00 ftpd[4711]: bluhm
logged in
```

Agenda

1 Motivation

2 Starting Position

3 Local Improvements

4 Remote Logging

5 Conclusion

/dev/log

Problems with `/dev/log` UNIX socket

- needs file descriptor
- use `LOG_NDELAY`
- reconnect after `SIGHUP` syslogd
- needs UNIX socket in chroot
- needs `pledge("unix")`
- LOG_CONS is even worse

sendsyslog

New system call sendsyslog(2)

```
int
sendsyslog(const void *msg, size_t len,
int flags)
```

```
sendsyslog("<94>Sep 24 09:36:23
ftpd[4711]: bluhm logged in", 47,
LOG_CONS)
```

Using sendsyslog

Syslogd does

- create socketpair
- register one end with `ioctl(LIOCSFD)`
- receive form other end

Kernel does

- send to syslogd's socketpair
- write to console if necessary
- ktrace if activated
- count errors

Error Handling

```
void
syslog(int prio, const char *msg, ...)
```

- libc cannot return error
- program cannot log error

Kernel sendsyslog can do it

- count failures when sending to syslogd
- write message to syslog when it works again

```
sendsyslog: dropped 2 messages, error 57
```

Libc Timestamp

Timestamp from syslog(3)

- needs `/etc/localtime` in every chroot
- no year
- no time zone
- no indication of daylight saving time
- insufficient precision
- does not work for kernel messages

```
Sep 24 09:37:42
```

Syslogd Timestamp

Timestamp added by syslogd

- timestamp is optional in received message
- syslogd adds it if missing
- libc does not generate it
- syslogd -Z generates ISO format in UTC
- use millisecond precision

```
2017-09-24T07:38:59.333Z
```

Logging without Libc

System call sendsyslog allows logging

- from signal handler
- at memcpy overlap
- from stack protector handler
- from ld.so dynamic linker

dmesg Overflow

Detect dmesg overflow in log file

- ring buffer with kernel logs
- syslogd reads from `/dev/klog`
- messages may overwrite
- special kernel message at gap

```
<4>klog: dropped 1243 bytes, message
buffer full
```

Agenda

1 Motivation

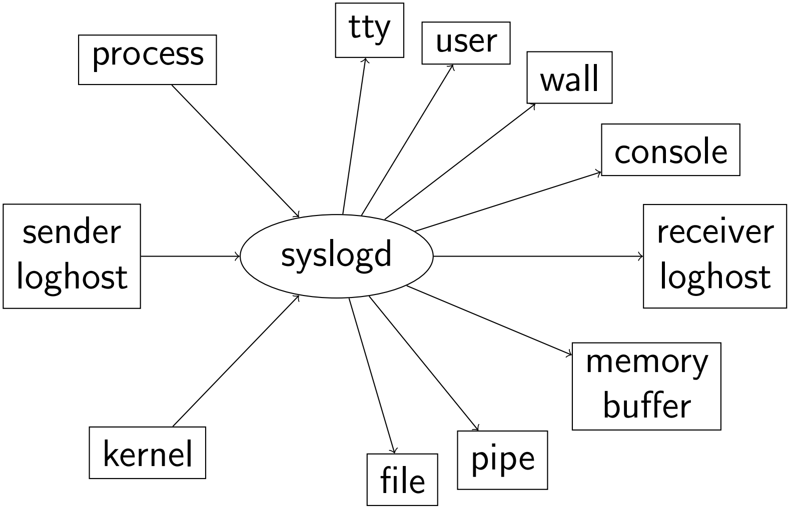
2 Starting Position

3 Local Improvements

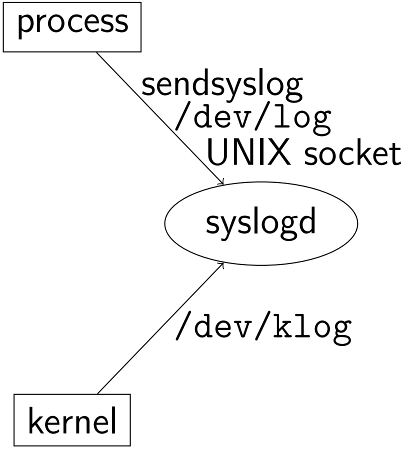
4 Remote Logging

5 Conclusion

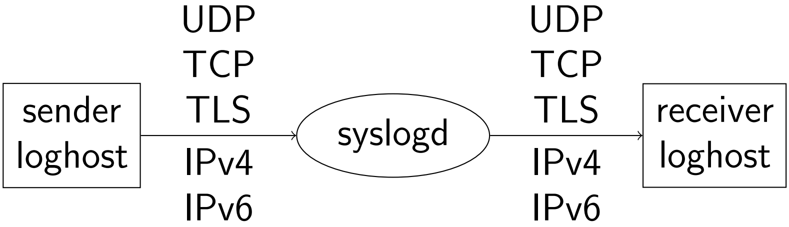
Possibilities



Local Methods



Remote Methods



UDP Format

- single UDP packet
- max 1180 bytes

```
<94>Sep 24 10:07:13 80.154.94.47
ftpd[4711]: bluhm logged in
```

TCP Format

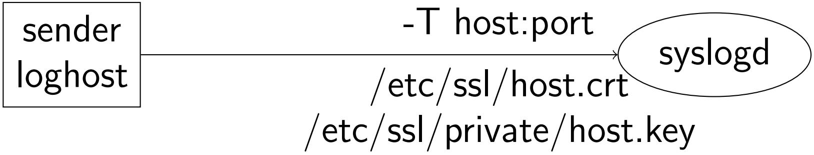
- no proper RFC 6587
- new line delimiter
- or NUL delimiter
- or octet counting

```
60 <94>Sep 24 10:08:52 80.154.94.47
ftpd[4711]: bluhm logged in
```

TLS Format

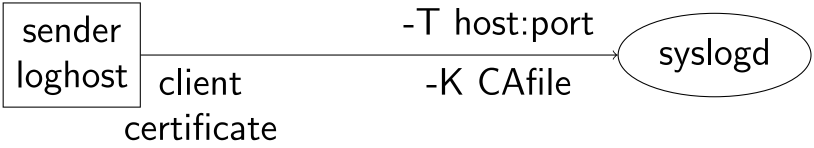
- octet counting
- must support 2048 bytes
- should support 8192 bytes
- libevent and libtls

Provide Server Certificate



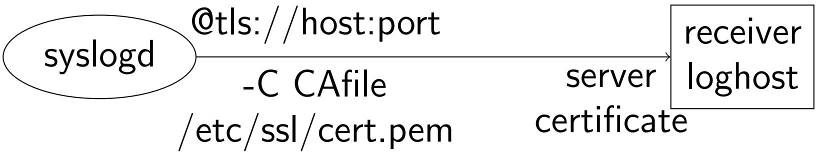
- syslogd must provide server certificate
- sender can identify syslogd
- attacker cannot see messages

Validate Client Certificate



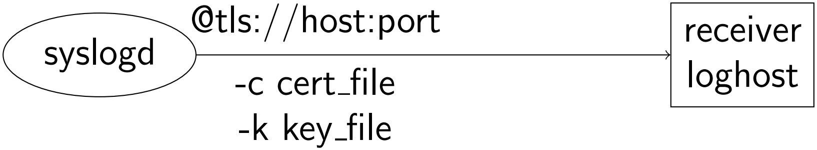
- sender may provide client certificate
- syslogd can identify sender
- attacker cannot inject messages

Validate Server Certificate



- syslogd must know server CA
- hostname must be in server certificate
- syslogd can identify receiver
- attacker cannot see messages
- turn off with -V

Provide Client Certificate



- syslogd may provide client certificate
- receiver can identify syslogd
- attacker cannot inject messages

TCP/TLS Errors

- debug incoming connections
- log connection errors
- count dropped messages
- suppress "last message repeated"

```
syslogd[17361]: dropped 2 messages to
remote loghost
```

Agenda

1 Motivation

2 Starting Position

3 Local Improvements

4 Remote Logging

5 Conclusion

OpenBSD Message Flow

program

```
syslog(LOG_ERR, "message %d", 7)
```

libc

priority, sprintf, *syscall*

kernel

sendsyslog, *error handling*

syslogd

recv, *timestamp*, log file, send *TLS*

Run and Log Reliably

- no fatal errors
- count dropped messages
- TCP and TLS transport
- libevent
- safe signal handlers
- file descriptor exhaustion
- privsep with re-exec
- pledge child and parent

Tests

- 180 regression tests
- for almost everything
- config, start, log, stop, check
- stderr, client, server, file, pipe, console, user, ktrace, fstat

TODO

- initialization errors to file
- continue after file system full
- log memory buffer overflow
- move format from RFC 3164 to 5424
- fix bug found by mpi@openbsd
- ivadasz@dragonfly likes kernel timestamps

Questions

?