Building a security appliance based on FreeBSD
Mariusz Zaborski

m.zaborski@fudosecurity.com

oshogbo@FreeBSD.org

https://oshogbo.vexillium.org

@oshogbovx
Data encryption
Data encryption

Storage
Data Encryption
Data Encryption

- GBDE
- GELI
- native ZFS encryption
GBDE - Geom Based Disk Encryption

- FreeBSD 5.0
- AES-CBC 128bits
- Different key for each write
  - CPU overhead
  - disk space overhead
GELI

● Many cryptographic algorithms
  ○ AES-XTS
  ○ AES-CBC
  ○ Blowfish-CBC
  ○ Camellia-CBC
  ○ 3DES-CBC
● Integrity verification (HMAC)
● Don’t have such overheads like GDBE
● One-time key
Keeping encryption key

Appliance:
- Use memstick
- Need only during boot
- Initialize during first boot

VM:
- Use passphrase
- Use no encryption
Storage
Storage

- ZFS
- UFS
ZFS

- checksums
- snapshots
- compression
- RAIDZ
ZFS - checksum

- fletcher2
- fletcher4
- sha256
- sha512
- skein

```
if (id < 0 ||
   id > channels_alloc)
```

```
if (id < 0 ||
   id >= channels_alloc)
```

```
jle 30
```

```
jl 30
```

```
39 45 08 7c 1a 8b 45
```

```
39 45 08 7e 1a 8b 45
```

```
01111100
```

```
01111110
```
ZFS - compression

- GZIP
- lz4
- ZSTD

# zfs list -o name,compression,compressratio

<table>
<thead>
<tr>
<th>NAME</th>
<th>COMPRESS</th>
<th>RATIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>data/data/local/dumps</td>
<td>lz4</td>
<td>16.20x</td>
</tr>
<tr>
<td>data/tmp</td>
<td>lz4</td>
<td>1.00x</td>
</tr>
<tr>
<td>data/var/crash</td>
<td>lz4</td>
<td>11.17x</td>
</tr>
</tbody>
</table>
ZFS - compression

- GZIP
- lz4
- ZSTD

```
# zfs list -o name,compression,compressratio

<table>
<thead>
<tr>
<th>NAME</th>
<th>COMPRESS</th>
<th>RATIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>data/data/local/dumps</td>
<td>lz4</td>
<td>16.20x</td>
</tr>
<tr>
<td>data/tmp</td>
<td>lz4</td>
<td>1.00x</td>
</tr>
<tr>
<td>data/var/crash</td>
<td>lz4</td>
<td>11.17x</td>
</tr>
</tbody>
</table>
```

Problem: What if customer want to backup the data?
ZFS - snapshots

A **snapshot** is a read-only copy of a file system or volume. Snapshots can be created almost instantly, and they initially consume no additional disk space within the pool. However, as data within the active dataset changes, the snapshot consumes disk space by continuing to reference the old data, thus preventing the disk space from being freed.

https://docs.oracle.com/cd/E23824_01/html/821-1448/gbciq.html
Snapshots - cluster multi-master

Master0  \hspace{1cm} \text{continuous replication} \hspace{1cm} Master1

New Data  \hspace{2cm} New Data
Snapshots - cluster multi-master

Master0 → continuous replication → Master1

New Data → access to all data → New Data
## Snapshots - cluster multi-master

```
# zfs list

<table>
<thead>
<tr>
<th>NAME</th>
<th>USED</th>
<th>AVAIL</th>
<th>REFER</th>
<th>MOUNTPOINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>135G</td>
<td>7.93T</td>
<td>192K</td>
<td>/</td>
</tr>
<tr>
<td>data/data</td>
<td>135G</td>
<td>7.93T</td>
<td>2.82M</td>
<td>/data</td>
</tr>
<tr>
<td>data/data/12345678/dumps</td>
<td>192K</td>
<td>7.93T</td>
<td>192K</td>
<td>/data/12345678/dumps</td>
</tr>
<tr>
<td>data/data/local/dumps</td>
<td>7.27G</td>
<td>7.93T</td>
<td>7.27G</td>
<td>/data/local/dumps</td>
</tr>
</tbody>
</table>
```
Snapshots - cluster multi-master

```bash
# zfs list -t snapshot

<table>
<thead>
<tr>
<th>NAME</th>
<th>USED</th>
<th>AVAIL</th>
<th>REFER</th>
<th>MOUNTPOINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>data/data/12345678/dumps@20180130051939</td>
<td>0</td>
<td>-</td>
<td>192k</td>
<td>-</td>
</tr>
<tr>
<td>data/data/local/dumps@20180130051934</td>
<td>0</td>
<td>-</td>
<td>7.27G</td>
<td>-</td>
</tr>
<tr>
<td>data/data/local/dumps@20180130052038</td>
<td>0</td>
<td>-</td>
<td>192k</td>
<td>-</td>
</tr>
</tbody>
</table>
```
ZFS sending & receiving snapshots

Before r317414:
- ZFS decompress FS to send
- manual compress FS to reduce latency
- sending over SSH
- manual decompress FS received over SSH
- ZFS compress FS which was received

After r317414:
- ZFS FS send over SSH
- ZFS FS receive over SSH
Downside of using ZFS snapshots

- Data loss after rollback
- Can’t rollback ZFS changes
- Snapshots can take a lot of space on cluster multi-master
Downside of using ZFS snapshots

- Data loss after rollback
- Can’t rollback ZFS changes
- Snapshots can take a lot of space on cluster multi-master
Downsides of ZFS

- Not enough RAM to import pool
- No full disk encryption
- If something will go very wrong we still want to be able to do something
- What about factory reset?
Read only file system - UFS

- GELI&ZFS for customer data
- Contains read-only operating system
- Data are not encrypted
- If something goes wrong we can still boot from it
- Try to reflect some ZFS promises
Read only file system - UFS

```
# gpart show -l ada0
=>
   40  11721045101  ada0  GPT  (5.5T)
     40    128     1 boot0  (64K)
  168    8388608     2 system0-0  [bootme] (4.0G)
  8388776  8388608     3 system1-0 (4.0G)
 16777384  8388608     4 system2-0 (4.0G)
 25165992 16572416     5 swap0  (7.9G)
 41738408 11679306727     6 data0 (5.4T)
```
RAIDZ2

data  data  data  data  data  data  parity  parity
### Reflect RAIDZ2 with UFS

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>mirror/system0</td>
<td>COMPLETE</td>
<td>gpt/system0-0 (ACTIVE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gpt/system0-1 (ACTIVE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gpt/system0-2 (ACTIVE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gpt/system0-3 (ACTIVE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gpt/system0-4 (ACTIVE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gpt/system0-5 (ACTIVE)</td>
</tr>
</tbody>
</table>
## Reflect RAIDZ2 with SWAP

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>mirror/swap0</td>
<td>COMPLETE</td>
<td>gpt/swap1 (ACTIVE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gpt/swap2 (ACTIVE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gpt/swap0 (ACTIVE)</td>
</tr>
<tr>
<td>mirror/swap1</td>
<td>COMPLETE</td>
<td>gpt/swap3 (ACTIVE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gpt/swap4 (ACTIVE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gpt/swap5 (ACTIVE)</td>
</tr>
</tbody>
</table>
### Upgrade steps

<table>
<thead>
<tr>
<th>Sector</th>
<th>EBP</th>
<th>Size</th>
<th>Offset</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>45</td>
<td>46</td>
<td>49</td>
<td>20</td>
</tr>
<tr>
<td>50</td>
<td>51</td>
<td>52</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td>320</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>1</td>
</tr>
<tr>
<td>420</td>
<td>FF</td>
<td>FF</td>
<td>01</td>
<td>00</td>
</tr>
<tr>
<td>240</td>
<td>FF</td>
<td>FF</td>
<td>01</td>
<td>00</td>
</tr>
<tr>
<td>340</td>
<td>FF</td>
<td>FF</td>
<td>01</td>
<td>00</td>
</tr>
<tr>
<td>250</td>
<td>FF</td>
<td>FF</td>
<td>01</td>
<td>00</td>
</tr>
<tr>
<td>350</td>
<td>FF</td>
<td>FF</td>
<td>01</td>
<td>00</td>
</tr>
</tbody>
</table>

### GPT HEADER

- **signature**: e3c9e316-0b5c-4db8-817d-92dd00215ae
- **attributes**: 0
- **attributes**: Microsoft reserved partition

### GPT PARTITION ENTRY ARRAY

- **partition type guid**: ebd00a2a-29e5-4433-87c0-6b9e6726699c7
- **attributes**: 0
- **attributes**: Basic data partition

---

**FUDO SECURITY**
**Upgrade steps** - Boot from system0

```
# gpart show -l ada0
=> 40 11721045101 ada0 GPT (5.5T)
      128 1 boot0 (64K)
   168 8388608 2 system0-0 [bootme] (4.0G)
  8388776 8388608 3 system1-0 (4.0G)
16777384 8388608 4 system2-0 (4.0G)
25165992 16572416 5 swap0 (7.9G)
41738408 11679306727 6 data0 (5.4T)
```
Upgrade steps - override system1 and set bootonce

```bash
# gpart show -l ada0
=> 40 11721045101 ada0 GPT (5.5T)
    40 128 1 boot0 (64K)
   168 8388608 2 system0-0 [bootme] (4.0G)
  8388776 8388608 3 system1-0 [bootonce, bootme] (4.0G)
16777384 8388608 4 system2-0 (4.0G)
25165992 16572416 5 swap0 (7.9G)
41738408 11679306727 6 data0 (5.4T)
```
Upgrade steps - reboot

1. Boot [ENTER]
2. [Escape to loader prompt]
3. Rboot

Options:
4. [ACPI Support]: Enabled
5. Boot Safe [Mode]: NO
6. Boot [Single User]: NO
7. Boot [Verbose]: NO
Upgrade steps - bootloader removes bootme

```bash
# gpart show -l ada0

=>  40  11721045101  ada0  GPT  (5.5T)
     40     128         1  boot0  (64K)
   168   8388608        2  system0-0 [bootme] (4.0G)
  8388776  8388608       3  system1-0 [bootonce] (4.0G)
16777384  8388608       4  system2-0 (4.0G)
25165992  16572416      5  swap0  (7.9G)
41738408 11679306727    6  data0  (5.4T)
```
Upgrade steps

- Create zfs snapshot
- Upgrade error accrued
- Reboot machine
**Upgrade steps** - boot from partition with bootme

```
# gpart show -l ada0
=>  40 11721045101  ada0  GPT  (5.5T)
     40    128   1  boot0  (64K)
     168  8388608  2  system0-0  [bootme] (4.0G)
     8388776  8388608  3  system1-0  [bootonce] (4.0G)
    16777384  8388608  4  system2-0  (4.0G)
    25165992 16572416  5  swap0  (7.9G)
    41738408 11679306727  6  data0  (5.4T)
```
Upgrade steps - rollback

```
# zfs rollback -R data@upgrade

# gpart show -l ada0

  =>  40  11721045101  ada0  GPT  (5.5T)
      40        128  1  boot0  (64K)
      168      8388608  2  system0-0  [bootme]  (4.0G)
     8388776  8388608  3  system1-0  [bootfailed]  (4.0G)
    16777384  8388608  4  system2-0  (4.0G)
   25165992   16572416  5  swap0  (7.9G)
  41738408  11679306727  6  data0  (5.4T)
```
Upgrade steps - upgrade succeeded

```bash
# gpart show -l ada0
=>  40  11721045101  ada0  GPT  (5.5T)
     40          128          1  boot0  (64K)
168      8388608         2  system0-0  [bootme] (4.0G)
 8388776   8388608         3  system1-0  [bootonce] (4.0G)
16777384   8388608         4  system2-0  (4.0G)
25165992  16572416         5  swap0  (7.9G)
41738408  11679306727       6  data0  (5.4T)
```
Upgrade steps - upgrade succeeded

# gpart show -l ada0

=> 40 11721045101 ada0 GPT (5.5T)
    40  128  1 boot0  (64K)
    168  8388608  2 system0-0  (4.0G)
    8388776  8388608  3 system1-0 [bootme] (4.0G)
   16777384  8388608  4 system2-0  (4.0G)
   25165992 16572416  5 swap0  (7.9G)
   41738408 11679306727  6 data0  (5.4T)
Hot plug

```plaintext
notify 20 {
  match "system" "DEVFS";
  match "type" "CREATE";
  match "cdev" "^ada[0-9]+$";
  action "/usr/local/bin/newdisk $cdev";
};
```
External Storage
External storage

- NFS
- iscsi
- SAN over FC
External storage - NFS

- NFS
- iscsi
- SAN over FC

- No encryption
- No authorization
- Is it corporate solution?
- Able to mount on multiple machines
External storage - iscsi

- NFS
- iscsi
- SAN over FC

- Encryption
- Authorization
- Is it corporate solution?
- Not able to mount on multiple machines
External storage - SAN over FC

- NFS
- iscsi
- SAN over FC
- Encryption
- Authorization
- It is a corporate solution
- Not able to mount on multiple machines
External storage - SAN over FC

- NFS
- iscsi
- SAN over FC with GELI

- Encryption
- Authorization
- It is a corporate solution
- Not able to mount on multiple machines
Redundancy

- Use at least two FC cards
- Combine multiple luns with gmultipath
Remote access
Access the box

- Through SSH
- We don’t want to customize our builds per client
- In theory we can have an key per client
- SSH keys
  - Hard to hijack
  - What if your engineer change the job?
  - We have to be in customer network
Access the box - exotic

- IPMI
- Some video conference (like webex)
- No SSH keys
  - So maybe password after all?
  - But password is easy to hijack
  - What if yours enginner change the job?
Implementing S/Key (whlkey)

S/KEY password generation

1. Initial secret $W$  
   $H$ function

2. Password 1 $H(W)$  
   $H$ function

3. Password 2 $H(H(W))$

4. Password $n$ $H^n(W)$

   - This password is stored on the server

S/KEY authentication

1. The user has
   - Password $n$ $H^n(W)$
   - Password $n-1$ $H^{n-1}(W)$
   - Password $n-2$ $H^{n-2}(W)$
   - ...  
   - Password 2 $H(H(W))$

2. The server knows
   - Password $n$ $H^n(W)$ reference
   - Password $n-1$ $H^{n-1}(W)$ reference
   - Password 1 $H(W)$ reference

The initial secret must be discarded!
Implementing S/Key (whlkey)

● We configure it as:
  ○ 50 keys per day
  ○ The key length is 16 chars
  ○ Key is rotated every day

● Unified password:
  ○ O == 0, I == l, etc.

● The secret is stored in some trusted machine

● The engineer can only get keys for this week
Process security
Basic problem

- You can’t build everything from scratch
- You can’t audit everything
- Who you really trust?
Basic problem

- You can’t build everything from scratch
- You can’t audit everything
- Who you really trust?

Security stops where the trust begins
Privileged separation

- Reduce TCB
- Simple communication
Privileged process

● Have access to:
  ○ DB
  ○ Storage
  ○ Network

● Authenticate unprivileged process

● Extend capabilities of unprivileged process
Unprivileged process

- Have access to storage by single FD
- Have access to network by single/two FD
- Implements complicate logic
- Is sending a simple commands asking privileged process
- Limited RAM
- Limited CPU time
Privileged separation with FreeBSD
Capsicum

- tight sandboxing (cap_enter(2))
- capability rights (cap_rights_limit(2))
Libnv

- nvlist_create
- nvlist_add_${type}
- nvlist_get_${type}
- nvlist_take_${type}
- nvlist_move_${type}
- nvlist_send
- nvlist_recv
- nvlist_destroy

Types:
- string
- number
- bool
- nvlist
- descriptor
- binary
- array
Privileged separation - is it hard?

OpenSSL

OpenSSH

FreeRDP

freetype

tesseract

libNTLM

FreeTDS

libX11

leptonica
Privileged separation - network daemon

- Privileged process is waiting for connection
Privileged separation - network daemon

- Privileged process is waiting for connection
- New connection from client
Privileged separation - network daemon

- Privileged process is waiting for connection
- New connection from client
- Fork and create unprivileged process
Privileged separation - network daemon

- Privileged process is waiting for connection
- New connection from client
- Fork and create unprivileged process
- Client is authenticating
Privileged separation - network daemon

- New connection from client
- Fork and create unprivileged process
- Client is authenticating
- Privileged process is raising unprivileged process limits
Privileged separation - network daemon

- Fork and create unprivileged process
- Client is authenticating
- Privileged process is raising unprivileged process limits
- Creating connection to the server
Privileged separation - network daemon

- Client is authenticating
- Privileged process is raising unprivileged process limits
- Creating connection to the server
- Pass connection to unprivileged process
Privileged separation - network daemon

- Client is authenticating
- Privileged process is raising unprivileged process limits
- Creating connection to the server
- Pass connection to unprivileged process
Privileged separation - network daemon

- Privileged process is raising unprivileged process limits
- Creating connection to the server
- Pass connection to unprivileged process
- Create a dump file
Privileged separation - network daemon

- Privileged process is raising unprivileged process limits
- Creating connection to the server
- Pass connection to unprivileged process
- Create a dump file
- Pass dump file
Privileged separation - network daemon

- Privileged process is raising unprivileged process limits
- Creating connection to the server
- Pass connection to unprivileged process
- Create a dump file
- Pass dump file
Other methods

- Jails
- CloudABI
Thank you!

Mariusz Zaborski

- m.zaborski@fudosecurity.com
- oshogbo@FreeBSD.org
- https://oshogbo.vexillium.org
- @oshogbovx