



Securing Elixir Applications

ElixirConf.EU 2016 - Berlin

Bram Verburg
System Architect, Security Advocate
Cisco

“I am being paid to be paranoid”

Me, just now

Distributed Erlang

- beam @ ephemeral/TCP
- epmd @ 4369/TCP
- Short names: hostname
- Long names: FQDN, IP

```
$ ssh user@example.net epmd -names
epmd: up and running on port 4369 with data:
name phoenix at port 40831
$ ssh -N -L 4369:localhost:4369 \
  -L 40831:localhost:40831 user@example.net
```

```
$ ssh user@example.net epmd -names
epmd: up and running on port 4369 with data:
name phoenix at port 40831
$ ssh -N -L 4369:localhost:4369 \
  -L 40831:localhost:40831 user@example.net
```

```
$ iex --sname iex@localhost --remsh phoenix@localhost
Erlang/OTP 18 [erts-7.3] [source] [64-bit] [smp:8:8] [asy
```

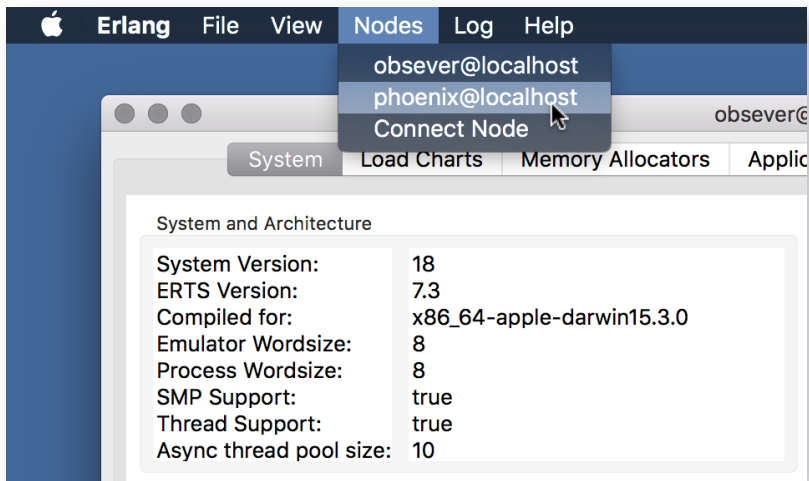
```
Interactive Elixir (1.2.3) - press Ctrl+C to exit (type h
iex(phoenix@localhost)1>
```

```
$ ssh user@example.net epmd -names
epmd: up and running on port 4369 with data:
name phoenix at port 40831
$ ssh -N -L 4369:localhost:4369 \
  -L 40831:localhost:40831 user@example.net

$ iex --sname iex@localhost --remsh phoenix@localhost
Erlang/OTP 18 [erts-7.3] [source] [64-bit] [smp:8:8] [asy

Interactive Elixir (1.2.3) - press Ctrl+C to exit (type h
iex(phoenix@localhost)1>

$ erl -sname observer@localhost -run observer
```




```
$ iex --sname test1@localhost  
[...]  
iex(test1@localhost)1>
```

```
$ epmd -names  
epmd: up and running on port 4369 with data:  
name test1 at port 51484
```

```
$ iex --sname test1@localhost  
[...]  
iex(test1@localhost)1>
```

```
$ epmd -names  
epmd: up and running on port 4369 with data:  
name test1 at port 51484
```

```
$ netstat -an | grep 4369  
tcp4    0    0  127.0.0.1.4369          127.0.0.1.51485      ESTABL  
tcp4    0    0  127.0.0.1.51485        127.0.0.1.4369      ESTABL  
tcp6    0    0  *.4369                  *.*                   LISTEN  
tcp4    0    0  *.4369                  *.*                   LISTEN  
$ netstat -an | grep 51484  
tcp4    0    0  *.51484                  *.*                   LISTEN  
$
```

```
$ ERL_EPMD_ADDRESS=127.0.0.1 iex --sname test1@localhost \  
  --erl "-kernel inet_dist_use_interface {127,0,0,1}"  
[...]  
iex(test1@localhost)1>
```

```
$ epmd -names
```

```
epmd: up and running on port 4369 with data:  
name test1 at port 51635
```

```
$ ERL_EPMD_ADDRESS=127.0.0.1 iex --sname test1@localhost \
  --erl "-kernel inet_dist_use_interface {127,0,0,1}"
[...]
```

iex(test1@localhost)1>

```
$ epmd -names
epmd: up and running on port 4369 with data:
name test1 at port 51635
```

```
$ netstat -an | grep 4369
tcp4    0  0  127.0.0.1.4369          127.0.0.1.51636      ESTABL
tcp4    0  0  127.0.0.1.51636        127.0.0.1.4369      ESTABL
tcp6    0  0  :::1.4369              *.*                  LISTEN
tcp4    0  0  127.0.0.1.4369          *.*                  LISTEN
$ netstat -an | grep 51635
tcp4    0  0  127.0.0.1.51635        *.*                  LISTEN
$
```

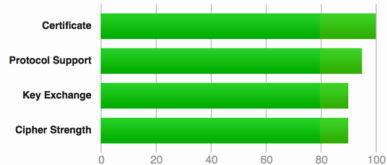
- Cookies provide limited security
 - Use *inet_tls_dist* for hardening
- Know when not to use it:
 - Over WAN connections
 - Between application tiers

TLS and :ssl

- **Server considerations:**
 - Does the application need a reverse proxy?
 - Will there be a load balancer?
 - Is TLS offload needed, for performance?
 - Do I trust Erlang/OTP's TLS implementation?

Summary

Overall Rating



Visit our [documentation page](#) for more information, configuration guides, and books. Known issues are documented [here](#).

HTTP Strict Transport Security (HSTS) with long duration deployed on this server. [MORE INFO »](#)

<https://www.ssllabs.com/ssltest/index.html>


```
config :app, App.Endpoint,  
  http: [port: 4000],  
  https: [port: 4001,  
    certfile: "priv/cert/cert.pem",  
    keyfile: "priv/cert/privkey.pem",  
    cacertfile: "priv/cert/chain.pem",  
    dhfile: "priv/cert/dhparams.pem",  
    secure_renegotiate: true,  
    honor_cipher_order: true,  
    ciphers: [  
      {:ecdhe_rsa, :aes_128_gcm, :null},  
      {:ecdhe_rsa, :aes_256_gcm, :null},  
      {:ecdhe_rsa, :aes_128_cbc, :sha256},  
      {:ecdhe_rsa, :aes_256_cbc, :sha384},  
      {:ecdhe_rsa, :aes_128_cbc, :sha},  
      {:ecdhe_rsa, :aes_256_cbc, :sha},
```

```
config :app, App.Endpoint,  
  http: [port: 4000],  
  https: [port: 4001,  
    certfile: "priv/cert/cert.pem",  
    keyfile: "priv/cert/privkey.pem",  
    cacertfile: "priv/cert/chain.pem",  
    dhfile: "priv/cert/dhparams.pem",  
    secure_renegotiate: true,  
    honor_cipher_order: true,  
    ciphers: [  
      {:ecdhe_rsa, :aes_128_gcm, :null},  
      {:ecdhe_rsa, :aes_256_gcm, :null},  
      {:ecdhe_rsa, :aes_128_cbc, :sha256},  
      {:ecdhe_rsa, :aes_256_cbc, :sha384},  
      {:ecdhe_rsa, :aes_128_cbc, :sha},  
      {:ecdhe_rsa, :aes_256_cbc, :sha},
```

```
config :app, App.Endpoint,  
  http: [port: 4000],  
  https: [port: 4001,  
    certfile: "priv/cert/cert.pem",  
    keyfile: "priv/cert/privkey.pem",  
    cacertfile: "priv/cert/chain.pem",  
    dhfile: "priv/cert/dhparams.pem",  
    secure_renegotiate: true,  
    honor_cipher_order: true,  
    ciphers: [  
      {:ecdhe_rsa, :aes_128_gcm, :null},  
      {:ecdhe_rsa, :aes_256_gcm, :null},  
      {:ecdhe_rsa, :aes_128_cbc, :sha256},  
      {:ecdhe_rsa, :aes_256_cbc, :sha384},  
      {:ecdhe_rsa, :aes_128_cbc, :sha},  
      {:ecdhe_rsa, :aes_256_cbc, :sha},
```

```
config :app, App.Endpoint,  
  http: [port: 4000],  
  https: [port: 4001,  
    certfile: "priv/cert/cert.pem",  
    keyfile: "priv/cert/privkey.pem",  
    cacertfile: "priv/cert/chain.pem",  
    dhfile: "priv/cert/dhparams.pem",  
    secure_renegotiate: true,  
    honor_cipher_order: true,  
    ciphers: [  
      {:ecdhe_rsa, :aes_128_gcm, :null},  
      {:ecdhe_rsa, :aes_256_gcm, :null},  
      {:ecdhe_rsa, :aes_128_cbc, :sha256},  
      {:ecdhe_rsa, :aes_256_cbc, :sha384},  
      {:ecdhe_rsa, :aes_128_cbc, :sha},  
      {:ecdhe_rsa, :aes_256_cbc, :sha},
```

```
config :app, App.Endpoint,  
  http: [port: 4000],  
  https: [port: 4001,  
    certfile: "priv/cert/cert.pem",  
    keyfile: "priv/cert/privkey.pem",  
    cacertfile: "priv/cert/chain.pem",  
    dhfile: "priv/cert/dhparams.pem",  
    secure_renegotiate: true,  
    honor_cipher_order: true,  
    ciphers: [  
      {:ecdhe_rsa, :aes_128_gcm, :null},  
      {:ecdhe_rsa, :aes_256_gcm, :null},  
      {:ecdhe_rsa, :aes_128_cbc, :sha256},  
      {:ecdhe_rsa, :aes_256_cbc, :sha384},  
      {:ecdhe_rsa, :aes_128_cbc, :sha},  
      {:ecdhe_rsa, :aes_256_cbc, :sha},
```

```
{:dhe_rsa, :aes_128_gcm, :null},  
{:dhe_rsa, :aes_256_gcm, :null},  
{:dhe_rsa, :aes_128_cbc, :sha256},  
{:dhe_rsa, :aes_256_cbc, :sha256},  
{:dhe_rsa, :aes_128_cbc, :sha},  
{:dhe_rsa, :aes_256_cbc, :sha},  
  
{:dhe_rsa, : "3des_edc_cbc", :sha},  
{:rsa, :aes_128_cbc, :sha},  
{:rsa, :aes_256_cbc, :sha},  
{:rsa, : "3des_edc_cbc", :sha},  
]  
],
```

```

{:dhe_rsa, :aes_128_gcm, :null},
{:dhe_rsa, :aes_256_gcm, :null},
{:dhe_rsa, :aes_128_cbc, :sha256},
{:dhe_rsa, :aes_256_cbc, :sha256},
{:dhe_rsa, :aes_128_cbc, :sha},
{:dhe_rsa, :aes_256_cbc, :sha},

{:dhe_rsa, : "3des_edc_cbc", :sha},
{:rsa, :aes_128_cbc, :sha},
{:rsa, :aes_256_cbc, :sha},
{:rsa, : "3des_edc_cbc", :sha},
]
],

```

- With Plug/Phoenix, use Plug.SSL
 - “HTTP Strict Transport Security”
 - Tell browser to only ever use HTTPS
 - Prevent downgrade attack
 - Prevent cookie hijacking


```
iex(1)> :inets.start && :ssl.start
:ok
iex(2)> :httpc.request 'https://self.voltone.net'
{:ok,
 {{'HTTP/1.1', 200, 'OK'},
  [{'date', 'Sat, 07 May 2016 07:20:18 GMT'},
   {'server', 'Cowboy'}],
  #...
```

```
iex(1)> :inets.start && :ssl.start
:ok
iex(2)> :httpc.request 'https://self.voltone.net'
{:ok,
 {{'HTTP/1.1', 200, 'OK'},
  [{'date', 'Sat, 07 May 2016 15:24:02 GMT'},
   {'server', 'Cowboy'}],
  #...
```



Safari can't verify the identity of the website "self.voltone.net".

The certificate for this website is invalid. You might be connecting to a website that is pretending to be "self.voltone.net", which could put your confidential information at risk. Would you like to connect to the website anyway?

☐ Always trust "self.voltone.net" when connecting to "self.voltone.net"



self.voltone.net



self.voltone.net

Root certificate authority

Expires: Monday, 4 May 2026 at 15:24:02 Israel Daylight Time

❗ This root certificate is not trusted

► Trust

► Details



Hide Certificate

Cancel

Continue

```
$ wget -q https://curl.haxx.se/ca/cacert.pem
$ iex --app inets --app ssl
iex(1)> :httpc.set_options(socket_opts: [
    verify: :verify_peer , cacertfile: 'cacert.pem'])
:ok
iex(2)>
```

```
$ wget -q https://curl.haxx.se/ca/cacert.pem
$ iex --app inets --app ssl
iex(1)> :httpc.set_options(socket_opts: [
    verify: :verify_peer, cacertfile: 'cacert.pem'])
:ok
iex(2)> :httpc.request 'https://self.voltone.net'

10:22:00.087 [error] SSL: :certify:
ssl_handshake.erl:1488:Fatal error: bad certificate
{:error,
 {:failed_connect,
  [{:to_address, {'self.voltone.net', 443}},
  # ...
```

```
iex(3)> :httpc.request 'https://api.voltone.net'  
{:ok,  
  {{'HTTP/1.1', 200, 'OK'},  
   [['date', 'Sat, 07 May 2016 07:32:31 GMT'],  
    ['server', 'Cowboy']],  
  # ...
```

```
iex(3)> :httpc.request 'https://api.voltone.net'  
{:ok,  
  {{'HTTP/1.1', 200, 'OK'},  
   [{'date', 'Sat, 07 May 2016 07:32:31 GMT'}],  
   {'server', 'Cowboy'}},  
  # ...
```



Safari can't verify the identity of the website "api.voltone.net".

The certificate for this website is invalid. You might be connecting to a website that is pretending to be "api.voltone.net", which could put your confidential information at risk. Would you like to connect to the website anyway?

☐ Always trust "voltone.net" when connecting to "api.voltone.net"



DST Root CA X3



Let's Encrypt Authority X3



voltone.net



voltone.net

Issued by: Let's Encrypt Authority X3

Expires: Thursday, 30 June 2016 at 09:22:00 Israel Daylight Time

✖ This certificate is not valid (host name mismatch)

► Trust

► Details



Hide Certificate

Cancel

Continue

hex Find packages Packages Documentation

hackney 1.6.0

simple HTTP client

Maintainers

Benoit Chesneau, Eduardo Gurgel

Links

[Github](#)
[Online documentation \(download\)](#)

License

Apache 2.0

46 023 downloads this version **4 017** downloads yesterday **22 301** downloads last 7 days

Versions

1.6.0 March 25, 2016
1.5.7 March 19, 2016
1.5.6 March 19, 2016
1.5.5 March 18, 2016
1.5.4 March 18, 2016
[Show All Versions](#)

Dependencies

certifi 0.4.0
idna 1.2.0
metrics 1.0.1
mimerl 1.0.2
ssl_verify_fun 1.1.0

Dependencies

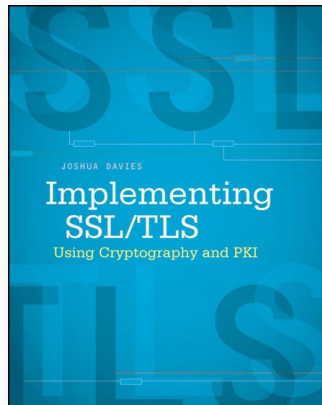
certifi 0.4.0
idna 1.2.0
metrics 1.0.1
mimerl 1.0.2
ssl_verify_fun 1.1.0

BULLETPROOF SSL AND TLS

Understanding and Deploying SSL/TLS and
PKI to Secure Servers and Web Applications



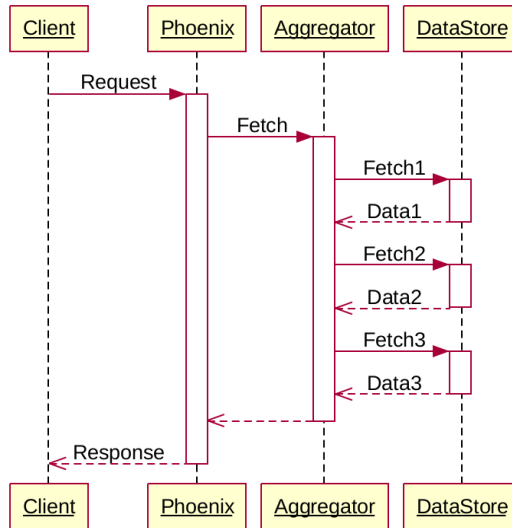
Ivan Ristić

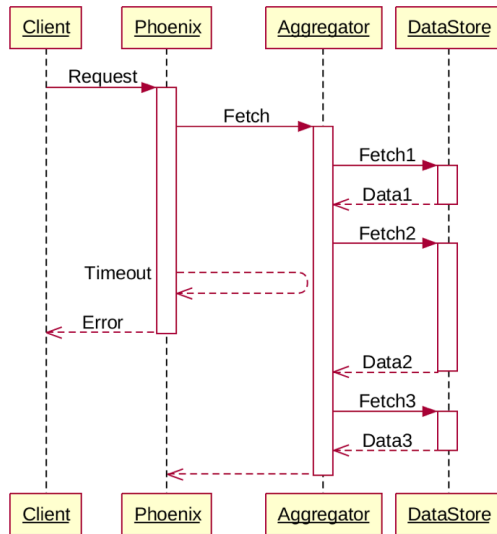


Denial of Service

- Atom table
 - Default size: 1.048.576 entries
 - No garbage collection
 - Overflow will crash the BEAM
- Mitigation
 - Use strings, or *String.to_existing_atom/1*

- **Process and Port limits**
 - Defaults: 262.144 and 65.536
 - Overflow causes error response
- **Mitigations**
 - Limit at the edge (e.g. :ranch)
 - Monitor and alert





- Fail fast
- Backpressure
- Compare Microservices:
 - Add circuit breakers and bulkheads

The Pragmatic
Programmers

Release It!

Design and Deploy
Production-Ready Software



Michael T. Nygard

O'REILLY

Building Microservices

DESIGNING FINE-GRAINED SYSTEMS



Sam Newman

Odds and Ends

- Phoenix response times
 - Low latency, low jitter
- More susceptible to timing attacks
 - For instance, username enumeration
- Beware of DoS attacks
 - E.g. Comeonin's *dummy_checkpw/0*

```
iex(1)> {:ok, {_, [{:abstract_code, {_, ast}}]}} =  
    String |>  
    :code.which |>  
    :beam_lib.chunks([:abstract_code])  
iex(2)> ast |>  
    :erl_syntax.form_list |>  
    :erl_prettypr.format |>  
    IO.puts  
//...  
to_atom(string@1) ->  
    erlang:binary_to_atom(string@1, utf8).  
  
to_integer(string@1, base@1) ->  
    erlang:binary_to_integer(string@1, base@1).  
:ok  
iex(3)>
```

- Compile without debug info:
 - `mix compile --no-debug-info`
- Strip `:abstract_code` chunk:
 - `:beam_lib.strip_files/1`
 - `:beam_lib.strip_release/1`
- Encrypt debug info

Thank you!

- IRC: voltone
- Twitter: @voltonez
- Questions? Or bonus section?

Bonus section: Phoenix as a TLS test server

```
config :app, App.Endpoint,  
  http: [port: 4000],  
  https: [  
    port: 4001,  
    # Default certificate file  
    certfile: "priv/cert/cert.pem",  
    keyfile: "priv/cert/privkey.pem",  
    cacertfile: "priv/cert/chain.pem",  
    sni_fun: &App.ssoptions/1  
  ],  
  # ...
```

```
config :app, App.Endpoint,  
  http: [port: 4000],  
  https: [  
    port: 4001,  
    # Default certificate file  
    certfile: "priv/cert/cert.pem",  
    keyfile: "priv/cert/privkey.pem",  
    cacertfile: "priv/cert/chain.pem",  
    sni_fun: &App.ssoptions/1  
  ],  
  # ...
```

```
def ssloptions('www.localtest.me'), do: []  
  
def ssloptions('expired.localtest.me'), do:  
  [certfile: 'priv/cert/expired.crt']
```



```
def ssloptions('www.localtest.me'), do: []

def ssloptions('expired.localtest.me'), do:
  [certfile: 'priv/cert/expired.crt']

def ssloptions('ssl3.localtest.me'), do:
  [versions: [:ssl3]]

def ssloptions('3des.localtest.me'), do:
  [
    ciphers:[
      {:ecdhe_rsa,  "3des_edc_cbc",  :sha},
      {:dhe_rsa,   "3des_edc_cbc",   :sha},
      {:rsa,       "3des_edc_cbc",    :sha}
    ]
  ]
```

```
$ openssl s_client -connect localhost:4001 \  
-CApath myrootca.pem \  
-servername expired.localtest.me
```

