In Offense of Erlang

Through the *Magic* of lols

Lab Mouse Security

Who am I?

- Well known security researcher
- DARPA grant winner
- Wrote the GSMA IoT Security Standard
- Black Hat, HITB, etc speaker
- Only public Erlang exploit developer

Lab Mouse Security

Thank You

Fred Hebert @ Heroku
Geoff Cant @ Heroku
Daed Latrope @ Heroku
Kenneth Lundin @ Ericsson
The entire Ericsson VM team

Lab Mouse Security

Why Does This Matter?

- Erlang runs GPRS, 3G, LTE
- Switching infrastructure
- Banking infrastructure
- Web infrastructure
- And soon? IoT infrastructure

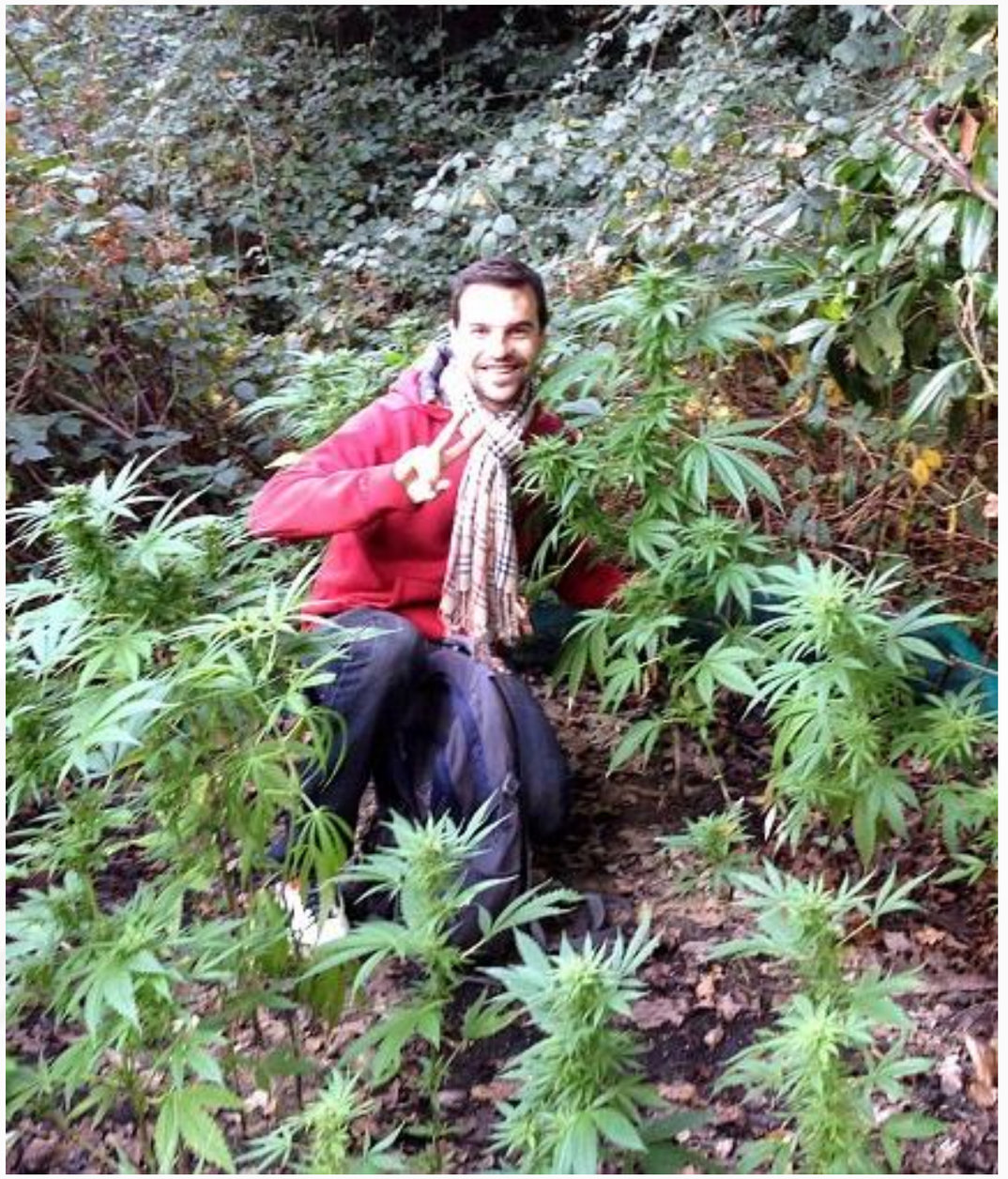
Lab Mouse Security

Erlang Security Model

Lab Mouse Security

Sort of a Secret Garden

- If the adversary can execute code, they can take over the node
- Compromising one node means compromising all nodes
- Node protection mechanisms must be rigidly managed



Lab Mouse Security

Traditional Risks Nearly Impossible

- All variables are write once -- sort of
- No pointers or direct memory access
- VM is register and heap based; no true stack
- All user written code is call-back based, reducing code complexity
- Pattern matching helps process data faster and isolates valid use cases from anomalies
- "Let it crash" mantra reduces overhead required to prevent crashes

BEAM = Root

- Loading code into an Erlang node is equivalent to having root
- Remember, Erlang is essentially an operating system of its own
- But there are no privileges beyond node access
- Many ways to execute custom code
- No way to verify code on update (md5 sums are versioning only)

All Nodes are One

- In the docs, "cookies" are indeed defined as security tokens
- If a node is accessible via the network, the only thing standing between you is the cookie (by default)
- Monotonic timers are used to auto-generate cookies; this is guessable based on host uptime (nmap); less entropy than previously thought
- Any Erlang thread can push code to any node in the mesh

The Exploit

Cookies and Names are Atoms

- Atoms live forever
- Connecting to an instance means submitting a node name
- Node name is converted to an Atom by the VM
- Atoms have a hard limit

Timing and Atoms

- The VM has multiple operations to convert a string
- Existing Atoms cannot be overwritten
- New atoms require memory allocation etc
- Timing delta is substantial
- Can detect existing Atoms this way

Submitting Atoms

- Joining a node requires submitting a name
- Name is converted to an atom
- Check delta here for existing atoms; stealth
- If the name exists, it's probably the cookie
- Reconnect using the cookie

Cookie generation is Predictable

- Cookies are combined from multiple integral types
- Bit entropy is very low
- Most cloud systems have guessable base integers
- Deltas are within 1MB
- That's a familiar number!

Stealth!

- Node connection isn't logged if the cookie isn't submitted!
- Atom generation isn't disclosed to supervisor!
- Only notification is a successful join

Lab Mouse Security

Putting it Together

- Check system type
- Find windows for initial cookie entropy
- Create 1MB window
- Launch Node -> Atom guessing attack
- Find existing atom -> Cooke!
- Reconnect with cookie

Lab Mouse Security

Vulnerable Systems

- Current Erlang versions are exploitable
- All Erlang versions are vulnerable
- Most cloud servers can be fingerprinted
- RabbitMQ and friends recommend vulnerable method of defining a cookie

Lab Mouse Security

Vulnerable Scope

- Robert Graham used massscan for me in 2016
- He found almost a quarter million open services
- Mostly RabbitMQ
- Cloud services likely vulnerable

The Assist

Lab Mouse Security

Admin Assassination

- Target specific nodes based on type
- Find administrative nodes or logger nodes
- Continually crash them
- Lols!!!

Lab Mouse Security

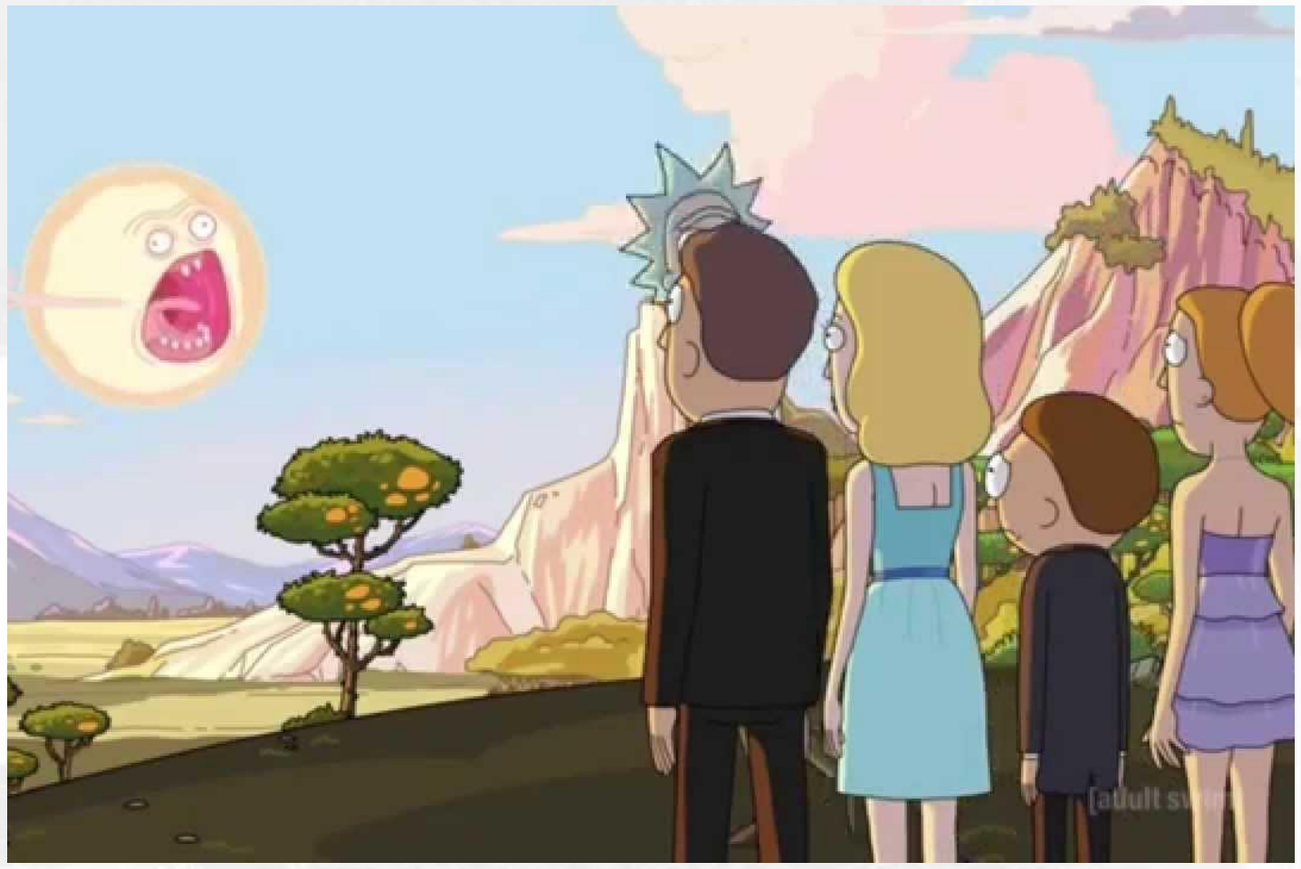
Core Files

- When Erlang dies it drops core
- Not like UNIX core, more like iOS dump
- Will show crash reason and all memory
- Auto delete it with EICAR test string!!!

Lab Mouse Security

Summary

- Erlang is a beautiful language
- Fast to develop, fast to deploy
- High security potential
- Yet, many areas of risk not previously known nor addressed
- Still requires auditing and security lifecycle integration



Lab Mouse Security

Questions?

Answers guaranteed.
(Guarantee void in Tennessee)

donb@securitymouse.com
www.securitymouse.com