

Gitano



<http://www.gitano.org.uk/>

Gitano



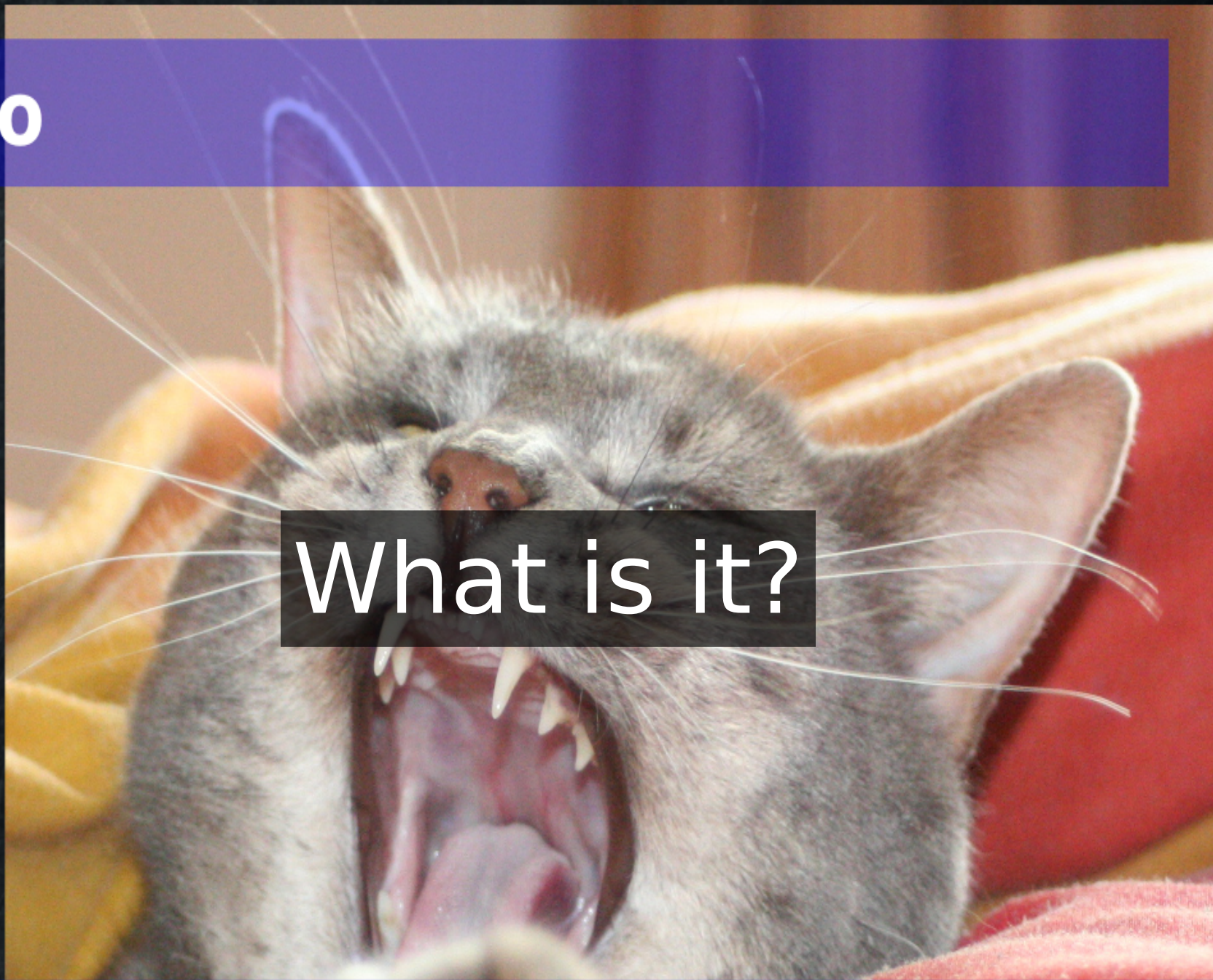
Gitano - A Git service written in Lua

Daniel Silverstone <dsilvers@digital-scurf.org>

<http://www.gitano.org.uk/>

Burple about who you are
Explain - no evil token parser stuff

Gitano



What is it?

<http://www.gitano.org.uk/>

Git Server

Written in Lua

Configured in Git where plausible

To do this I needed to write a bunch of libs



Gitano

Technology choices
(or things was too lazy to write)

<http://www.gitano.org.uk/>

Git - odd to list, but I mean configuration is in git
rulesets are in git, etc.etc.etc.

Lua - I like Lua, it's easy to prototype and write stuff

Luxio

libgit2/luagit2 - Way faster than invoking git commands
although Gitano **can** operate without them

cggit rather than gitweb - much faster, caches, prettier

Gitano



Gall - Git Abstraction Layer (in) Lua

<http://www.gitano.org.uk/>

Git abstraction - obviously necessary. Uses Luxio's subprocess to run git commandline and luagit2/libgit2 via LuaNativeObjects by Robert G. Jakabosky to work in process.

Gitano



```
r = gall.repository.new("/path/to/gitano-admin.git")
c = r:get(r.HEAD).content
t = gall.tree.flatten(c.tree.content)
b = t['site.conf']
print(b.obj.content)
```

<http://www.gitano.org.uk/>

Gitano



Lace - Lua Access Control Engine

<http://www.gitano.org.uk/>

Lace - Access control lists. No syntax I came up with in Lua was neat enough for non-Lua programmers to accept. Turing complete ACLs are also a fairly bad idea. Show example, show that Lace comes with instructions

Gitano



```
define success equals want_to_pass yes  
allow "Ok" success
```

<http://www.gitano.org.uk/>

Simple example showing definition, match type, and arguments.
list of defined predicates on the allow line must all pass

Gitano

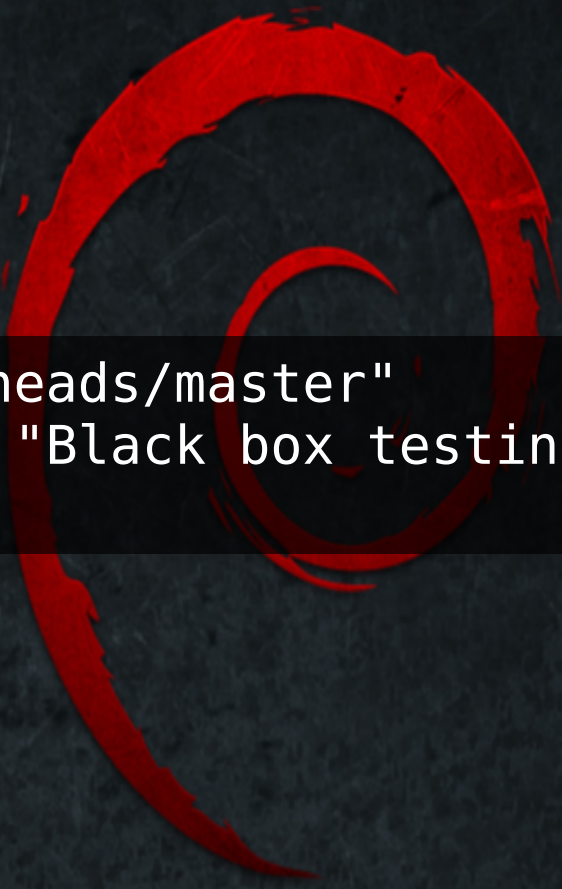


Clod - Configuration Language Organised (by) Dots

<http://www.gitano.org.uk/>

Designed to keep track of ordering of entries (and spaces)
Currently doesn't track comments (because that's super
hard) Humans and the library tend to edit files in similar
ways meaning diffs are sane

Gitano



```
project.head "refs/heads/master"  
project.description "Black box testing of Unix programs"  
project.owner "liw"
```

<http://www.gitano.org.uk/>

Three simple string entries as might be found in a repository configuration in Gitano.

Gitano




```
description "Gitano Instance Administrators"  
members["*"] "dsilvers"
```

<http://www.gitano.org.uk/>

Clod also supports lists which remain ordered. This is an example group file in a Gitano repository

Gitano



Supple - Sandbox [(for) Untrusted Procedure Partitioning (in) Lua] Engine

<http://www.gitano.org.uk/>

Supple allows me to run hooks provided by project owners safely without risking them gaining access to the server in any unusual way.

Hooks are run as Lua code with a limited set of functions and only the data relevant to the event they're hooking (along with a read-only repository object they can use to interrogate other things a bit)

Gitano



To limit the attack surface...

The "untrusted" code runs in a (limited) Lua sandbox.

That sandbox is soft-limited in terms of VM opcodes and memory.

The sandbox is monitored and IO marshalled externally.

<http://www.gitano.org.uk/>

Your "untrusted" code is run inside a Lua sandbox which has only a limited set of Lua's functionality exposed to it.

That sandbox is soft-limited (optionally) in terms of VM opcodes and memory allocated by Lua

The sandbox is run inside a monitoring Lua VM instance which is responsible for carefully marshalling calls etc into and out of the sandbox. All your comms go via this monitor.

Gitano

Just in case...

The monitor is a Lua VM anyway, and it's all inside a separate process.
The sandbox process is in an ephemeral chroot.

<http://www.gitano.org.uk/>

The monitor is, itself, a Lua VM anyway, inside a process which is separate from the process you're doing untrusted work on behalf of.

The sandbox process is created using a rootly helper so that it's put into an isolation state consisting of a directory which is owned by root which is set as your root via the chroot call, but which is also rmdir'd so it's ephemeral. Your process drops privileges back to the calling UID so it cannot do anything inside its CWD anyway.

Gitano



And if that's not enough...

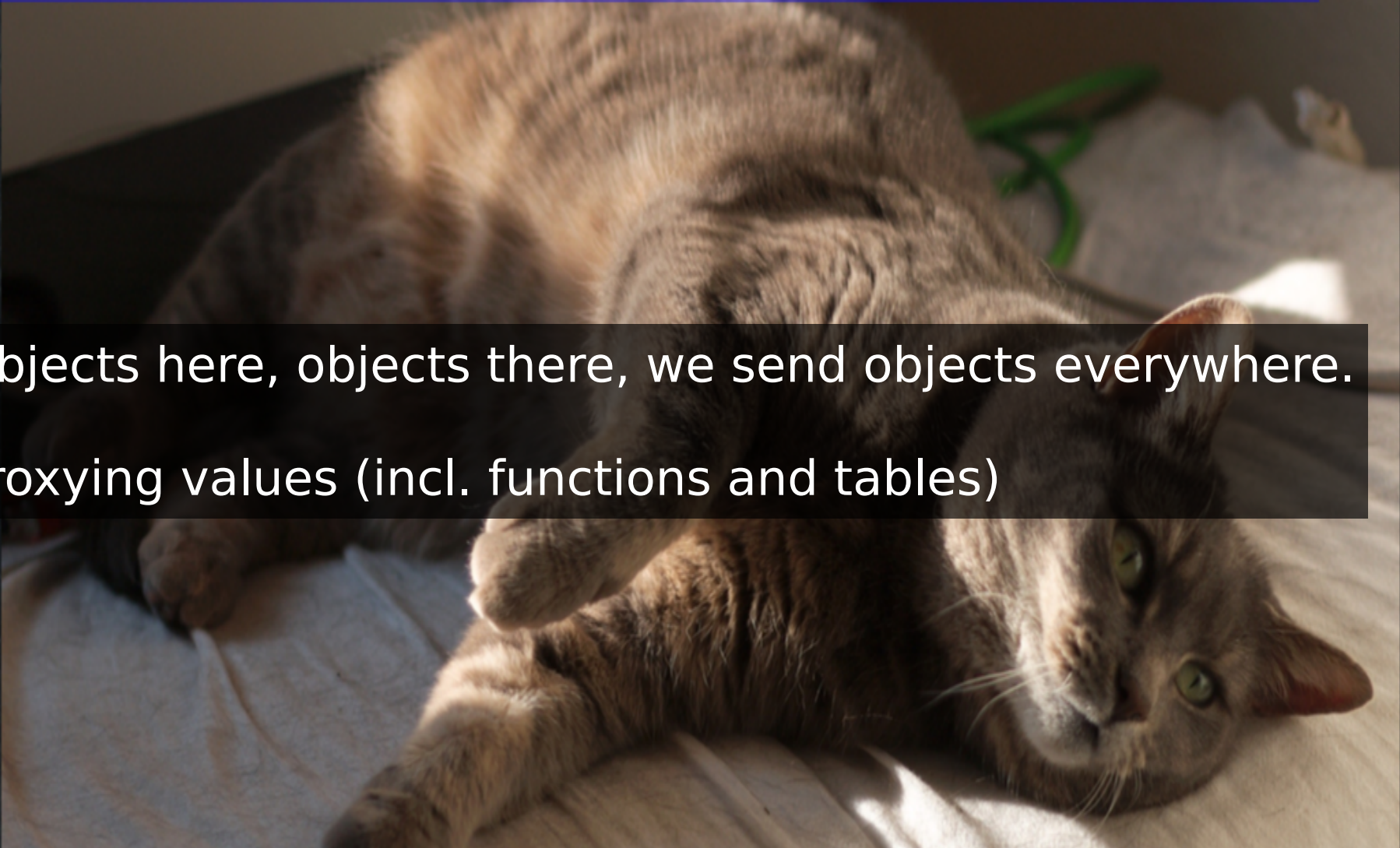
Solid rlimits in terms of memory and open FDs

And on Linux, memory is pre-allocated and we enter seccomp mode 1.

<http://www.gitano.org.uk/>

On top of that, the sandbox has some pretty solid rlimits set in terms of max CPU usage, max VM size, max FDs open, and max size of any file it writes. As such, it can't create > 0 byte files in the directory it doesn't have access to, and could only do that if it closed the FD to the host process which is its only communications avenue. Then, if you're on Linux, we go one step further and pre-allocate enough memory for the interpreter to not hit the rlimit and then enter seccomp mode 1 which limits the syscalls permissible to read, write, _exit and sigreturn so even if you could have circumvented any/all of the limits above, you now can't make syscalls to take advantage of them. If that's not sandbox enough, please tell me how to improve matters further.

Gitano



Objects here, objects there, we send objects everywhere.
Proxying values (incl. functions and tables)

<http://www.gitano.org.uk/>

Talk about how Supple proxies values across the link so that the sandboxed code can have do whatever it likes and it looks and feels like it's running in the host.
e.g. `next()` works, calling things works etc.

Gitano

```
local repo, ref, oldsha, newsha = ...

local branch = ref:match("^refs/heads/(.+)$")
if branch == "master" then
    log.state("Looking at commit history on: " .. branch)

    local commit = repo:get(newsha)

    while commit.sha ~= oldsha do
        commit = commit.content
        local parents = commit.parents
        if #parents < 2 then
            error("Detected non-merge-commit during parent walk, at " .. commit.sha)
        end
        commit = parents[1]
    end

    log.state("Commits between old and new sha seem to all be merge commits")
else
    log.state("Skipping commit history check on: " .. ref)
end
```

<http://www.gitano.org.uk/>

Non-trivial example which shows an update hook for Gitano to prevent non-merge commits when pushing to master

Gitano

```
local repo, ref, oldsha, newsha = ...

local branch = ref:match("^refs/heads/(.+)$")
if branch == "master" then
  log.state("Looking at commit history on: " .. branch)

  local commit = repo:get(newsha)

  while commit.sha ~= oldsha do
    commit = commit.content
    local parents = commit.parents
    if #parents < 2 then
      error("Detected non-merge-commit during parent walk, at " .. commit.sha)
    end
    commit = parents[1]
  end

  log.state("Commits between old and new sha seem to all be merge commits")
else
  log.state("Skipping commit history check on: " .. ref)
end
```

<http://www.gitano.org.uk/>

Green is input arguments for the hook, passed in by the host app

Red is an example of a module table passed over from the host

Gitano

```
local repo, ref, oldsha, newsha = ...

local branch = ref:match("^refs/heads/(.+)")
if branch == "master" then
    log.state("Looking at commit history on: " .. branch)

    local commit = repo:get(newsha)

    while commit.sha ~= oldsha do
        commit = commit.content
        local parents = commit.parents
        if #parents < 2 then
            error("Detected non-merge-commit during parent walk, at " .. commit.sha)
        end
        commit = parents[1]
    end

    log.state("Commits between old and new sha seem to all be merge commits")
else
    log.state("Skipping commit history check on: " .. ref)
end
```

<http://www.gitano.org.uk/>

Here, blue are simple table lookups which propagate across the link to the host

And yellow are function invocations which call functions in the host

Gitano

```
local repo, ref, oldsha, newsha = ...

local branch = ref:match("^refs/heads/(.+)$")
if branch == "master" then
    log.state("Looking at commit history on: " .. branch)

    local commit = repo:get(newsha)

    while commit.sha ~= oldsha do
        commit = commit.content
        local parents = commit.parents
        if #parents < 2 then
            error("Detected non-merge-commit during parent walk, at " .. commit.sha)
        end
        commit = parents[1]
    end

    log.state("Commits between old and new sha seem to all be merge commits")
else
    log.state("Skipping commit history check on: " .. ref)
end
```

<http://www.gitano.org.uk/>

Purple is a call to error

Explain how errors propagate back and forth across the link, obeying pcalls and trying to gather stack info for both sides. Also note that the host can extract extra debugging about the entire supple transaction.

Gitano



Real users of Gitano

- git.gitano.org.uk, git.liw.fi
- git.netsurf-browser.org, richard.maw.name/git
- Codethink and Baserock

<http://www.gitano.org.uk/>

Equally horrifyingly, people use this crap what I wrote.
But, it's not enough (sound stern)

Gitano



Future plans

Lots of ideas for future content, see the Trello for some of the things I have planned.

- <https://trello.com/b/l4ld6iiC/gitano>
- (Link is on www.gitano.org.uk)

<http://www.gitano.org.uk/>

Future - explain how background task stuff looks plausible using the nanomsg stuff recently talked about on list. Explain how currently I'm testing Gitano using a testing tool written in Python, but want to write a Lua equivalent of it.

Gitano



Mailing list: gitano-dev@gitano.org.uk

IRC Channel: #gitano on Freenode

Website: <http://www.gitano.org.uk/>

Any questions?

<http://www.gitano.org.uk/>

Intrusive cat says "Enough with the talkings"

Gitano



Mailing list: gitano-dev@gitano.org.uk

IRC Channel: #gitano on Freenode

Website: <http://www.gitano.org.uk/>

Thank you for listening

<http://www.gitano.org.uk/>