RAGNARNET – BREVET OFFICIEL

Documentation technique consolidée des modules du système RagnarNet.

FICHIER : update.lua

1. Objectif du fichier :
Ce fichier est le programme d'installation initial de RagnarNet. Il télécharge et installe les autres fichiers nécessaires.

2. Fonctionnement général :
Le fichier est analysé pour détecter ses fonctions principales, boucles, et appels externes.

3. Code source :

```
-- update.lua : Installation / MAJ complète RagnarNet (met à jour manifest & ver
sion)

local function println(c, msg)
  if term and colors and c then term.setTextColor(c) end
  print(msg)
  if term and colors then term.setTextColor(colors.white) end
end

-- Télécharge via pastebin
local function download(id, dest)
  if fs.exists(dest) then fs.delete(dest) end
  return shell.run("pastebin get " .. id .. " " .. dest)
end

local function readAll(p)
  if not fs.exists(p) or fs.isDir(p) then return "" end
  local f = fs.open(p, "r"); local s = f.readAll() or ""; f.close(); return s
end

-- BXOR portable + FNV1a
local function BXOR(a, b)
  if bit and bit.bxor then return bit.bxor(a, b) end
  if bit32 and bit32.bxor then return bit32.bxor(a, b) end
  local r, v = 0, 1
  while a > 0 or b > 0 do
    local A, B = a % 2, b % 2
    if (A + B) % 2 == 1 then r = r + v end
    a = math.floor(a / 2); b = math.floor(b / 2); v = v * 2
  end
  return r
end
local function fnv1a(s)
  local h = 2166136261
  for i = 1, #s do
    h = BXOR(h, s:byte(i))
    h = (h * 16777619) % 4294967296
  end
  return tostring(h)
end
local function fileHash(path) return fnv1a(readAll(path)) end
local function extractCodeVer(txt) return txt:match('CODE_VER%s*=%s*"%s*([^"]-)%
s*"') end
```

```
-- IDs OFFICIELS (on n'utilise plus ceux de config.lua)
local files = {
  { id = "m7wpD8wF", name = "startup.lua" },
  { id = "DWHJU4bC", name = "ui.lua" },
  { id = "jK7srvyY", name = "config.lua" },
  { id = "gNHAVd7D", name = "update.lua" },
}

println(colors.cyan, "=== RagnarNet Installer ===")

-- 1) Téléchargements
for _, f in ipairs(files) do
  println(colors.lightBlue, "Telechargement de "..f.name.." ...")
  local ok = download(f.id, f.name)
  if not ok then println(colors.red, "Echec de telechargement: "..f.name); retur
n end
end

-- 2) Heuristique anti-sabotage pour le startup
local sTxt = readAll("startup.lua")
local ok_ver  = sTxt:match('local%s+CODE_VER%s*=%s*"7%.1%.0"')
local ok_db   = sTxt:match('usersDB%s*=%s*"users%.db"')
if not (ok_ver and ok_db) then
  println(colors.red, "Startup invalide (signature heuristique). Annulation.")
  return
end

-- 3) Manifest & expected version
local cfg = {}
local ver  = extractCodeVer(sTxt) or "7.1.0"

cfg.expectedStartupVersion = ver
cfg.autoSeal = true
cfg.tamperAction, cfg.outdatedAction = "error","error"
cfg.askUpdateAtBoot = true
cfg.key="RAGNAR123456789KEYULTRA2025"; cfg.protocol="ragnarnet"
cfg.adminUser="ragnar"; cfg.adminCode="2013.2013"
cfg.spamLimit=5; cfg.maxMessageLength=200; cfg.spamResetTime=300
cfg.pepper="RAG-PEPPER-2025"; cfg.pwdHashRounds=512
cfg.updateURL_startup="m7wpD8wF"; cfg.updateURL_ui="DWHJU4bC"
cfg.updateURL_config="jK7srvyY"; cfg.updateURL_update="gNHAVd7D"
cfg.errorCodeTamper=163; cfg.errorCodeOutdated=279
cfg.manifest = {
  ["startup.lua"] = fileHash("startup.lua"),
  ["ui.lua"]      = fileHash("ui.lua"),
  ["update.lua"]  = fileHash("update.lua"),
}
local function writeConfigTable(tbl)
  local ser = textutils.serialize(tbl)
  local f = fs.open("config.lua", "w"); f.write("return " .. ser); f.close()
end
writeConfigTable(cfg)

println(colors.lime, "Installation terminee. Redemarrage dans 3 secondes...")
sleep(3)
os.reboot()
```

FICHIER : ui.lua

1. Objectif du fichier :
Contient les fonctions liées à l'affichage et à l'interface utilisateur dans le terminal (texte, boîtes, menus...).

2. Fonctionnement général :
Le fichier est analysé pour détecter ses fonctions principales, boucles, et appels externes.

3. Code source :

```
-- ui.lua : RagnarNet UI (adapté 7.2.x)
-- Exporte: ui.drawUI(username, isAdmin, w, h, version)
--          ui.showMessages(messages, uiHeight, blacklist, adminUser)

local ui = {}

-- Couleurs sûres (fallback si écran non couleur)
local HAS_COLOR = term.isColor and term.isColor()
local C_BG      = HAS_COLOR and colors.lightGray or colors.white
local C_TITLE   = HAS_COLOR and colors.blue      or colors.black
local C_TEXT    = HAS_COLOR and colors.white     or colors.black
local C_MUTE    = HAS_COLOR and colors.gray      or colors.black
local C_WARN    = HAS_COLOR and colors.orange    or colors.black
local C_BAD     = HAS_COLOR and colors.red       or colors.black

local function fill(x1,y1,x2,y2,c)
  paintutils.drawFilledBox(x1,y1,x2,y2,c)
end

local function titleBar(w, title)
  paintutils.drawLine(1, 1, w, 1, C_TITLE)
  term.setCursorPos(2,1)
  term.setTextColor(C_TEXT)
  term.write(title or "")
end

local function footBar(w, h, text)
  paintutils.drawLine(1, h-2, w, h-2, C_MUTE)
  term.setCursorPos(2, h-1)
  term.setTextColor(C_MUTE)
  term.write(text or "")
end

local function trunc(s, maxw)
  if #s <= maxw then return s end
  return s:sub(1, math.max(0, maxw-1)) .. "?"
end

-- Public: dessine l'UI statique (cadre, barres, bouton arrêter)
function ui.drawUI(username, isAdmin, w, h, version)
  term.setBackgroundColor(colors.black)
  term.clear()
  fill(1,1,w,h,C_BG)

  local head = " RagnarNet UI "
  if version then head = head .. "v"..tostring(version).." " end
  titleBar(w, head)

  -- bande inférieure (zone d'aide)
  local who = "Connecté en tant que " .. (username or "?") .. (isAdmin and " [AD
```

```
MIN]" or "")
  footBar(w, h, who)

  -- bouton arrêt (click: y==1, x>=w-12)
  term.setCursorPos(math.max(1, w-12), 1)
  term.setTextColor(C_BAD)
  term.write("[ARRETER]")
end

-- Public: affiche la liste des messages dans la zone centrale
-- messages = { {id=1, from="u", text="...", admin=false}, ... }
function ui.showMessages(messages, uiHeight, blacklist, adminUser)
  local w, h = term.getSize()
  -- on nettoie la zone centrale (lignes 2..h-3)
  for y = 2, (h-3) do
    term.setCursorPos(2, y)
    term.clearLine()
  end

  local maxWidth = math.max(1, w - 4) -- marge à gauche/droite
  local start = math.max(1, #messages - uiHeight + 1)

  for i = start, #messages do
    local m = messages[i]
    local line = (i - start) + 2 -- commence sous la barre titre

    -- couleur par type
    if m and m.admin then
      term.setTextColor(C_WARN)
    elseif m and m.from and blacklist and blacklist[m.from] then
      term.setTextColor(C_BAD)
    else
      term.setTextColor(C_TEXT)
    end

    local id   = tostring(m.id or i)
    local from = tostring(m.from or "?")
    local txt  = tostring(m.text or "")

    local raw = "["..id.."] "..from..": "..txt
    local out = trunc(raw, maxWidth)

    term.setCursorPos(2, line)
    term.write(out)
  end
end

return ui
```

FICHIER : startup.lua

1. Objectif du fichier :
C'est le point de départ du système RagnarNet. Il initialise l'environnement, lance l'interface et applique la configuration.

2. Fonctionnement général :
Le fichier est analysé pour détecter ses fonctions principales, boucles, et appels externes.

3. Code source :

```
-- startup.lua : RagnarNet OS principal v7.1.0 (restauré + durci)

-----------------------------------------------------------------
-- >>> SECURE PREAMBLE (sans hash, anti-suppression d'appel) <<<
-----------------------------------------------------------------
do
  -- 1) Fichiers essentiels présents + non vides + marqueurs de structure
  local essentiels = {
    { "ui.lua",     "return", "drawUI" },     -- doit être un module qui 'return
' un tableau + avoir drawUI
    { "config.lua", "return", "{"      },     -- doit 'return {'
    { "users.db",   nil,      nil      },     -- peut être vide au 1er boot, mai
s doit exister
  }

  local function readAll(p)
    if not fs or not fs.exists or not fs.exists(p) or fs.isDir(p) then return ""
 end
    local f = fs.open(p, "r"); local s = f.readAll() or ""; f.close(); return s
  end

  for _, spec in ipairs(essentiels) do
    local path, m1, m2 = spec[1], spec[2], spec[3]
    if not fs or not fs.exists or not fs.exists(path) then
      error("[SECURITE] Fichier essentiel manquant : "..tostring(path))
    end
    local data = readAll(path)
    if #data == 0 and path ~= "users.db" then
      error("[SECURITE] Fichier essentiel vide : "..tostring(path))
    end
    if m1 and not data:find(m1, 1, true) then
      error("[SECURITE] Structure invalide dans "..path.." (marqueur "..m1.." ab
sent)")
    end
    if m2 and not data:find(m2, 1, true) then
      error("[SECURITE] Structure invalide dans "..path.." (marqueur "..m2.." ab
sent)")
    end
  end
end
-- >>> FIN PREAMBLE <<<

local CODE_VER = "7.1.0"
local cfg = require("config")

-------------------------------
-- Garde-fous config forcés
-------------------------------
cfg.spamLimit       = math.max(1,  math.min(50,   tonumber(cfg.spamLimit or 5))
```

```
)
cfg.maxMessageLength = math.max(10, math.min(500,  tonumber(cfg.maxMessageLength
 or 200)))
cfg.pwdHashRounds    = math.max(128,math.min(4096, tonumber(cfg.pwdHashRounds or
 512)))
if cfg.strict_mode == false then cfg.strict_mode = true end
-- Empêcher un contournement via actions trop ?douces?
cfg.tamperAction   = (cfg.tamperAction   == "halt" or cfg.tamperAction   == "err
or") and cfg.tamperAction   or "error"
cfg.outdatedAction = (cfg.outdatedAction == "halt" or cfg.outdatedAction == "err
or") and cfg.outdatedAction or "error"

----------------
-- Utils
----------------
local function BXOR(a, b)
  if bit and bit.bxor then return bit.bxor(a, b) end
  if bit32 and bit32.bxor then return bit32.bxor(a, b) end
  local r, v = 0, 1
  while a > 0 or b > 0 do
    local A, B = a % 2, b % 2
    if (A + B) % 2 == 1 then r = r + v end
    a = math.floor(a / 2); b = math.floor(b / 2); v = v * 2
  end
  return r
end

local function readAll(p)
  if not fs.exists(p) or fs.isDir(p) then return "" end
  local f = fs.open(p, "r"); local s = f.readAll() or ""; f.close(); return s
end

local function fnv1a(s)
  local h = 2166136261
  for i = 1, #s do
    h = BXOR(h, s:byte(i))
    h = (h * 16777619) % 4294967296
  end
  return tostring(h)
end
local function fileHash(path) return fnv1a(readAll(path)) end

local function writeConfigTable(tbl)
  local ser = textutils.serialize(tbl)
  local f = fs.open("config.lua", "w"); f.write("return " .. ser); f.close()
end

------------------------
-- Erreurs propres
------------------------
local function showErrorAndExit(code, reason)
  term.setBackgroundColor(colors.black)
  term.setTextColor(colors.red)
  term.clear()
  term.setCursorPos(2,2)
  print("[ERREUR " .. tostring(code) .. "] RagnarNet")
  term.setTextColor(colors.white)
  print(reason or "Erreur de securite")
  print("\nLe programme s'arrete. Lance 'update' ou utilise la disquette de reco
```

```
very.")
  sleep(2.5)
  error("ERR_"..tostring(code), 0)
end

local function handleBreach(kind, reason, action)
  action = action or "error"
  local code = 199
  if kind == "tamper"   then code = (cfg.errorCodeTamper or 163) end
  if kind == "outdated" then code = (cfg.errorCodeOutdated or 279) end
  if action == "error" then showErrorAndExit(code, reason)
  elseif action == "halt" then error(reason or "Security halt", 0)
  else showErrorAndExit(199, reason or "Security error") end
end

-------------------------------------------------
-- Intégrité (avec anti-reseal et auto-run)
-------------------------------------------------
local function integrityCheck()
  -- Anti-reseal : un seul ?seal? autorisé à l?installation
  local sealedFlag = ".sealed"
  local firstBoot = not fs.exists(sealedFlag)

  -- 1) Version attendue (si définie dans config)
  if cfg.expectedStartupVersion and cfg.expectedStartupVersion ~= CODE_VER then
    handleBreach("outdated",
      "Version trop ancienne: "..tostring(CODE_VER).." (attendue "..tostring(cfg
.expectedStartupVersion)..")",
      cfg.outdatedAction or cfg.tamperAction or "error")
  end

  -- 2) Cibles d?intégrité ? on n?inclut jamais config.lua
  local targets = { "startup.lua", "ui.lua", "update.lua" }

  -- 3) Manifest
  if not cfg.manifest then
    if cfg.autoSeal and firstBoot then
      local newcfg = {}; for k,v in pairs(cfg) do newcfg[k] = v end
      newcfg.manifest = {}
      for _, p in ipairs(targets) do newcfg.manifest[p] = fileHash(p) end
      writeConfigTable(newcfg)
      local f = fs.open(sealedFlag, "w"); f.write("ok"); f.close()
      term.setTextColor(colors.lime); print("[Integrite] Scellage initial OK (ma
nifest ecrit)."); term.setTextColor(colors.white)
    else
      -- Si manifest absent mais pas ?vraie? installation : tentative de reseal
-> breach
      local msg = firstBoot and "Manifest absent et autoSeal=false (installation
 corrompue)."
                              or  "Manifest absent (tentative de re-scellage int
erdite)."
      handleBreach("tamper", msg, cfg.tamperAction or "error")
    end
  else
    for _, p in ipairs(targets) do
      local exp, act = cfg.manifest[p], fileHash(p)
      if not exp or exp ~= act then
        handleBreach("tamper", "Integrite rompue sur: "..p, cfg.tamperAction or
"error")
```

```
      end
    end
  end
end

-- AUTO-RUN : vérifie l?intégrité même si l?appel plus bas est supprimé
do
  local ok, err = pcall(integrityCheck)
  if not ok then
    handleBreach("tamper", "Echec verif integrite: "..tostring(err), cfg.tamperA
ction or "error")
  end
end

-------------------------------------------------
-- MAJ au démarrage (prompt unique)
-------------------------------------------------
local _askedUpdateOnce = false
local function askUpdateAtBoot()
  if _askedUpdateOnce then return end
  _askedUpdateOnce = true
  if cfg.askUpdateAtBoot == false then return end
  if not fs.exists("update.lua") then return end
  term.setTextColor(colors.cyan); print("\nFaire la mise a jour maintenant ? (o/
n)"); term.setTextColor(colors.white)
  local a = read()
  if a and a:lower() == "o" then
    local dat = readAll("update.lua")
    if dat == "" then showErrorAndExit(503, "update.lua invalide ou vide") end
    shell.run("update.lua")
  end
end

-- ====== Boot: intégrité -> prompt MAJ -> charge UI ======
-- Même si quelqu?un supprime la ligne suivante, l?intégrité a déjà été vérifiée
 (auto-run ci-dessus).
integrityCheck()
askUpdateAtBoot()

---------------------------------
-- Chargement et vérif de l'UI
---------------------------------
local function loadUI()
  if package and package.loaded then package.loaded["ui"] = nil end
  local ok, mod = pcall(dofile, "ui.lua")
  if not ok then showErrorAndExit(501, "ui.lua: "..tostring(mod)) end
  if type(mod) ~= "table" or not mod.drawUI or not mod.showMessages then
    showErrorAndExit(502, "ui.lua invalide (fonctions manquantes)")
  end
  return mod
end
local ui = loadUI()

-----------------------------
-- App / runtime
-----------------------------
local w, h     = term.getSize()
local uiHeight = h - 6
```

```
local usersDB  = "users.db"

-- Nettoyage d?artefacts connus
for _, f in ipairs({"HACKER.db"}) do if fs.exists(f) then pcall(fs.delete, f) en
d end

local messages, users, spamTracker, blacklist, banDuration = {}, {}, {}, {}, {}
local username, isAdmin, lockdown = "?", false, false

local function xorCrypt(msg, keyStr)
  local out = {}
  for i = 1, #msg do
    local m = msg:byte(i)
    local k = keyStr:byte((i - 1) % #keyStr + 1)
    out[i] = string.char(BXOR(m, k))
  end
  return table.concat(out)
end

local function addMessage(from, text, adminFlag)
  table.insert(messages, { id = #messages + 1, from = from, text = text, admin =
 adminFlag or false })
  ui.drawUI(username, isAdmin, w, h, CODE_VER)
  ui.showMessages(messages, uiHeight, blacklist, cfg.adminUser)
end

-- Users (hash + migration)
local function loadUsers()
  if not fs.exists(usersDB) then return {} end
  local f = fs.open(usersDB, "r"); local d = textutils.unserialize(f.readAll());
 f.close()
  return d or {}
end
local function saveUsers(u)
  local f = fs.open(usersDB, "w"); f.write(textutils.serialize(u)); f.close()
end
local function randHex(n) local s={} for i=1,n do s[i]=string.format("%x",math.r
andom(0,15)) end return table.concat(s) end
local function fnvRounds(s) local r=tonumber(cfg.pwdHashRounds or 512) or 512; l
ocal h=s; for _=1,r do h=fnv1a(h) end; return h end
local function hashPassword(pwd, salt) return fnvRounds(tostring(salt or "") ..
tostring(pwd or "") .. tostring(cfg.pepper or "")) end
local function verifyPassword(stored, input)
  if type(stored)=="string" then return stored==input, "legacy"
  elseif type(stored)=="table" and stored.salt and stored.hash then return store
d.hash==hashPassword(input,stored.salt),"hashed" end
  return false,"unknown"
end

-- Modem
for _, side in ipairs({"left","right","top","bottom","front","back"}) do
  if peripheral.getType(side) == "modem" then rednet.open(side); break end
end

-- Login
users = loadUsers()
math.randomseed(os.epoch and os.epoch("utc") or os.time() or os.clock())
```

```
term.setTextColor(colors.yellow) write("Pseudo > "); term.setTextColor(colors.wh
ite)
username = read()

if users[username] then
  term.setTextColor(colors.yellow) write("Mot de passe > "); term.setTextColor(c
olors.white)
  local pwd = read("*")
  local ok, mode = verifyPassword(users[username], pwd)
  if not ok then print("Mot de passe incorrect.") return end
  if mode == "legacy" then
    local salt = randHex(16)
    users[username] = { salt = salt, hash = hashPassword(pwd, salt) }
    saveUsers(users)
    addMessage("SYSTEM", "Compte migre vers hachage.", false)
  end
else
  term.setTextColor(colors.yellow) write("Creer un mot de passe > "); term.setTe
xtColor(colors.white)
  local pwd = read("*"); local salt = randHex(16)
  users[username] = { salt = salt, hash = hashPassword(pwd, salt) }
  saveUsers(users)
end

if username == cfg.adminUser then
  write("Code Ragnar > ")
  if read() == cfg.adminCode then isAdmin = true else print("Code incorrect.") r
eturn end
end

-- UI initiale
ui.drawUI(username, isAdmin, w, h, CODE_VER)
ui.showMessages(messages, uiHeight, blacklist, cfg.adminUser)

-- Watchdog runtime : re-vérifie l?essentiel régulièrement
local function watchdog()
  while true do
    -- re-check fichiers essentiels
    local ok, err = pcall(function()
      local marks = {
        { "ui.lua",     "return", "drawUI" },
        { "config.lua", "return", "{"      },
        { "users.db",   nil,      nil      },
      }
      for _, spec in ipairs(marks) do
        local path, m1, m2 = spec[1], spec[2], spec[3]
        if not fs.exists(path) then error("Essentiel supprimé: "..path) end
        local data = readAll(path)
        if #data == 0 and path ~= "users.db" then error("Essentiel vide: "..path
) end
        if m1 and not data:find(m1, 1, true) then error("Structure invalide (man
que "..m1..") dans "..path) end
        if m2 and not data:find(m2, 1, true) then error("Structure invalide (man
que "..m2..") dans "..path) end
      end
    end)
    if not ok then
      handleBreach("tamper", "Watchdog: "..tostring(err), "error")
```

```
    end
    sleep(math.max(2, tonumber(cfg.watchdogDelay or 5)))
  end
end

-- Threads
local function spamReset() while true do sleep(cfg.spamResetTime) spamTracker =
{} end end
local function handleClick()
  while true do
    local _, _, x, y = os.pullEvent("mouse_click")
    if y == 1 and x >= w - 12 then
      term.setTextColor(colors.red) print("\nConfirmer l'arret ? (o/n)")
      term.setTextColor(colors.white)
      if (read() or ""):lower() == "o" then error("Arret utilisateur", 0) end
    end
  end
end
local function receiver()
  while true do
    local _, encrypted = rednet.receive(cfg.protocol)
    local raw = (encrypted and cfg.key) and (function(m,k)
      local out, b = {}, nil
      for i = 1, #m do
        local mm = m:byte(i)
        local kk = k:byte((i - 1) % #k + 1)
        out[i] = string.char(BXOR(mm, kk))
      end
      return table.concat(out)
    end)(encrypted, cfg.key) or ""
    local from, text = raw:match("(.+):(.+)")
    if not from or not text then
      -- DoS fix : on ignore le paquet mal formé
      sleep(0)
    else
      if #text > cfg.maxMessageLength then
        addMessage("SYSTEM","Message trop long de "..from)
      elseif blacklist[from] then
        -- ignore
      elseif from ~= username then
        spamTracker[from] = (spamTracker[from] or 0) + 1
        if spamTracker[from] > cfg.spamLimit and from ~= cfg.adminUser then
          banDuration[from] = (banDuration[from] or 7200) * 5
          blacklist[from] = true
          addMessage("SYSTEM", from.." banni pour spam")
        else
          addMessage(from, text, (from == cfg.adminUser))
        end
      end
    end
  end
end
local function sender()
  while true do
    term.setCursorPos(2, h); term.setTextColor(colors.white); write("Vous > ")
    local input = read()
    if input ~= "" and not lockdown then
      if #input > cfg.maxMessageLength then
```

```
        addMessage("SYSTEM","Message trop long (max "..cfg.maxMessageLength..")"
)
      else
        local plain = username..":"..input
        rednet.broadcast(xorCrypt(plain, cfg.key), cfg.protocol)
        addMessage(username, input, isAdmin)
      end
    end
  end
end

parallel.waitForAny(receiver, sender, handleClick, spamReset, watchdog)
```

FICHIER : config.lua

1. Objectif du fichier :
Stocke la configuration globale de RagnarNet : nom d'utilisateur, paramètres de l'OS, sécurité, etc.

2. Fonctionnement général :
Le fichier est analysé pour détecter ses fonctions principales, boucles, et appels externes.

3. Code source :

```
return {
  updateURL_update = "gNHAVd7D",
  spamLimit = 5,
  errorCodeOutdated = 279,
  maxMessageLength = 200,
  adminUser = "ragnar",
  updateURL_ui = "DWHJU4bC",
  manifest = {
    [ "update.lua" ] = "723186416",
    [ "startup.lua" ] = "2620602876",
    [ "ui.lua" ] = "322429472",
  },
  pepper = "RAG-PEPPER-2025",
  key = "RAGNAR123456789KEYULTRA2025",
  expectedStartupVersion = "7.1.0",
  tamperAction = "error",
  pwdHashRounds = 512,
  autoSeal = true,
  askUpdateAtBoot = true,
  errorCodeTamper = 163,
  updateURL_config = "jK7srvyY",
  updateURL_startup = "m7wpD8wF",
  protocol = "ragnarnet",
  adminCode = "2013.2013",
  outdatedAction = "error",
  spamResetTime = 300,
}
```