The Carbonara Project

The front window of malware research

Main features

What is Carbonara about?

Binary analysis

Perform advanced static analysis directly from your browser.

Procedure matching

Find common patterns across binaries and speedup your work

Machine Learning

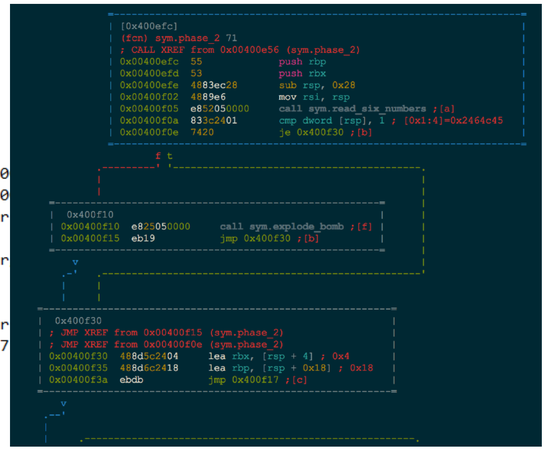
Run your binaries against our machine learning engine to discover if it’s malicious.

Binary analysis

- Upload a raw binary and get useful information about it
- Retrieve all the procedures that compose the uploaded program

```
q00003,     AND fnc026   ;Address of q0
fnc026,     CLE          ;Function q000
            BSA pop      ;Getting retur
            STA temp1
            BSA pop      ;Processing ar
            STA q00028
            LDA temp1
            BSA push     ;Putting retur
            CLE          ;{WORD} q00027
            LDA q00032
            BSA push
            BSA pop
            STA q00027

            CLE          ;{WORD} q00035 = q00030
            LDA q00030
            BSA push
            BSA pop
            STA q00035
```



Machine Learning

Is it malicious?

Procedure matching

- Read the disassembled code of each one of the procedures and see other info about it uploaded by other users.
- Given a procedure, get a list of similar procedures.

The Infrastructure

Machine Learning engine

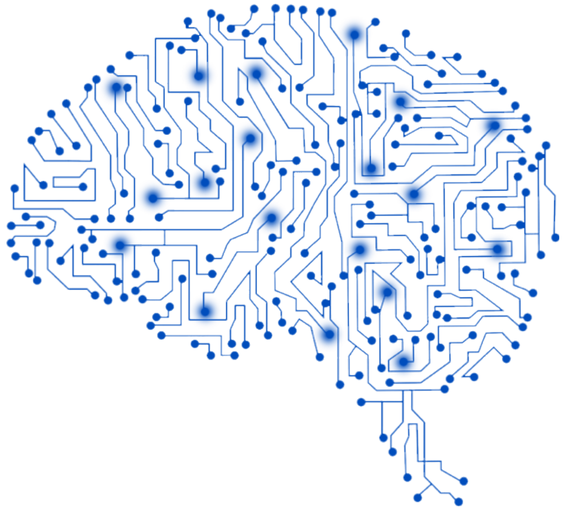
Our progress

✓ We built the server infrastructure

✓ We’ve prototyped the model

What are the issues?

- Finding good data for training is hard
- Training a good model requires a lot of computing power



Guanciale (Analysis Engine)

Our progress

✓ Retrieval data from various tools and disassemblers (radare2, IDA Pro)

✓Procedure abstraction and hashing

What are the issues?

- Extract info from IDA Pro database
- Testing the algorithm takes a lot of time

Front-end

Our progress

- ✓ Created Login/Register pages
- ✓ Created Dashboard page for updates and searching
- ✓ Created Profile page showing user activities
- ✓ Created Upload, Binary and Procedure pages

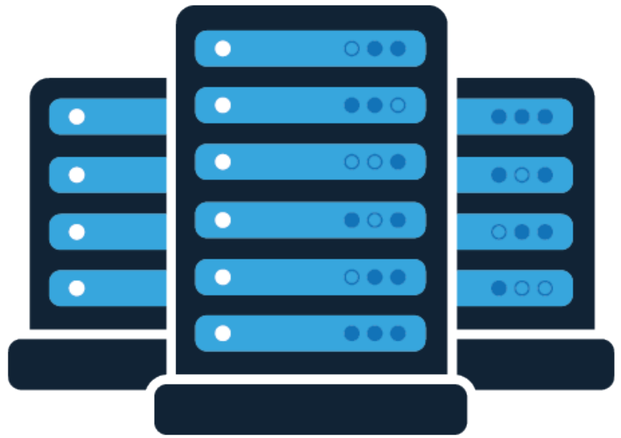
Back-end

Our progress

- ✓ User registration
- ✓ Managing DB for procedures descriptions
- ✓ Save report from analysis server

What are the issues?

- Choose the right way to represents procedures and link them to programs.
- Optimise singular procedures to work in disk.

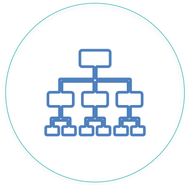


Roadmap

What's ahead?

Make it social

Comments, discussions, ratings, reputation.



Add data and interaction between components

Add more meaningful information on binaries, link data together to create an ecosystem.

Design

Graphic enhancements through better design and meaningful animations.

```
if ( gamemode == commercial)
        if ( gamemode == pack_tnt )
        if ( gamemode == pack_plut ) )
{
        skytexture = R_TextureNumForName ("SKY3");
        if (gamemap < 12)
          skytexture = R_TextureNumForName ("SKY1");
        else
          if (gamemap < 21)
                skytexture = R_TextureNumForName ("SKY2");
}

levelstarttic = gametic;        // for time calculation

if (wipegamestate == GS_LEVEL)
        wipegamestate = -1;             // force a wipe

gamestate = GS_LEVEL;

for (i=0 ; i<MAXPLAYERS ; i++)
{
        if (playeringame[i] && players[i].playerstate == PST_DEAD)
          players[i].playerstate = PST_REBORN;
        memset (players[i].frags,0,sizeof(players[i].frags));
}

P_SetupLevel (gameepisode, gamemap, 0, gameskill);
displayplayer = consoleplayer;          // view the guy you are playing
starttime = I_GetTime ();
gameaction = ga_nothing;
Z_CheckHeap ();

// clear cmd building stuff
memset (gamekeydown, 0, sizeof(gamekeydown));
joyxmove = joyymove = 0;
mousex = mousey = 0;
sendpause = sendsave = paused = false;
memset (mousebuttons, 0, sizeof(mousebuttons));
memset (joybuttons, 0, sizeof(joybuttons));
}

void G_PlayerReborn (int player)
{
  player_t*     p;
  int           i;
  int           frags[MAXPLAYERS];
  int           killcount;
  int           itemcount;
  int           secretcount;

  memcpy (frags,players[player].frags,sizeof(frags));
  killcount = players[player].killcount;
  itemcount = players[player].itemcount;
  secretcount = players[player].secretcount;

  p = &players[player];
  memset (p, 0, sizeof(*p));

  memcpy (players[player].frags, frags, sizeof(players[player].frags));
  players[player].killcount = killcount;
  players[player].itemcount = itemcount;
  players[player].secretcount = secretcount;

  p->usedown = p->attackdown = true;    // don't do anything immediately
  p->playerstate = PST_LIVE;
  p->health = MAXHEALTH;
  p->readyweapon = p->pendingweapon = wp_pistol;
  p->weaponowned[wp_fist] = true;
  p->weaponowned[wp_pistol] = true;
  p->ammo[am_clip] = 50;

  for (i=0 ; i<NUMAMMO ; i++)
        p->maxammo[i] = maxammo[i];
}
```

```
  secretexit = false;
}
else
  secretexit = true;
gameaction = ga_completed;
}

void G_DoCompleted (void)
{
int             i;

gameaction = ga_nothing;

for (i=0 ; i<MAXPLAYERS ; i++)
  if (playeringame[i])
    G_PlayerFinishLevel (i);        // take away cards and stuff

if (automapactive)
  AM_Stop ();

if ( gamemode != commercial)
  switch(gamemap)
  {
    case 8:
      gameaction = ga_victory;
      return;
    case 9:
      for (i=0 ; i<MAXPLAYERS ; i++)
        players[i].didsecret = true;
      break;
  }

//jff 3/28/98 Hommy - why?
if ( (gamemap == 8)
  && (gamemode != commercial) )
{
  // victory
  gameaction = ga_victory;
  return;
}

if ( (gamemap == 9)
  && (gamemode != commercial) )
{
  // exit secret level
  for (i=0 ; i<MAXPLAYERS ; i++)
    players[i].didsecret = true;
}
//#endif


wminfo.didsecret = players[consoleplayer].didsecret;
wminfo.epsd = gameepisode -1;
wminfo.last = gamemap -1;

// wminfo.next is 0 biased, unlike gamemap
if ( gamemode == commercial)
{
  if (secretexit)
    switch(gamemap)
    {
      case 15: wminfo.next = 30; break;
      case 31: wminfo.next = 31; break;
    }
  else
    switch(gamemap)
    {
      case 31:
      case 32: wminfo.next = 15; break;
      default: wminfo.next = gamemap;
    }
}
else
{
  if (secretexit)
    wminfo.next = 8;        // go to secret level
  else if (gamemap == 9)
  {
    // returning from secret level
    switch (gameepisode)
    {
      case 1:
        wminfo.next = 3;
        break;
      case 2:
        wminfo.next = 5;
```

```
if (paused)
{
        paused = false;
        S_ResumeSound ();
}


if (skill > sk_nightmare)
        skill = sk_nightmare;


// This was quite messy with SPECIAL and commented parts.
// Supposedly hacks to make the latest edition work.
// It might not work properly.
if (episode < 1)
  episode = 1;

if ( gamemode == retail )
{
  if (episode > 4)
        episode = 4;
}
else if ( gamemode == shareware )
{
  if (episode > 1)
        episode = 1;    // only start episode 1 on shareware
}
else
{
  if (episode > 3)
        episode = 3;
}



if (map < 1)
        map = 1;

if ( (map > 9)
        && ( gamemode != commercial) )
  map = 9;

M_ClearRandom ();

if (skill == sk_nightmare || respawnparm )
        respawnmonsters = true;
else
        respawnmonsters = false;

if (fastparm || (skill == sk_nightmare && gameskill != sk_nightmare) )
{
        for (i=S_SARG_RUN1 ; i<=S_SARG_PAIN2 ; i++)
          states[i].tics >>= 1;
        mobjinfo[MT_BRUISERSHOT].speed = 20*FRACUNIT;
        mobjinfo[MT_HEADSHOT].speed = 20*FRACUNIT;
        mobjinfo[MT_TROOPSHOT].speed = 20*FRACUNIT;
}
else if (skill != sk_nightmare && gameskill == sk_nightmare)
{
        for (i=S_SARG_RUN1 ; i<=S_SARG_PAIN2 ; i++)
          states[i].tics <<= 1;
        mobjinfo[MT_BRUISERSHOT].speed = 15*FRACUNIT;
        mobjinfo[MT_HEADSHOT].speed = 10*FRACUNIT;
        mobjinfo[MT_TROOPSHOT].speed = 10*FRACUNIT;
}


// force players to be initialized upon first level load
for (i=0 ; i<MAXPLAYERS ; i++)
        players[i].playerstate = PST_REBORN;

usergame = true;                // will be set false if a demo
paused = false;
demoplayback = false;
automapactive = false;
viewactive = true;
gameepisode = episode;
gamemap = map;
gameskill = skill;

viewactive = true;

// set the sky map for the episode
if ( gamemode == commercial)
```

```
switch (gameaction)
{
  case ga_loadlevel:
    G_DoLoadLevel ();
    break;
  case ga_newgame:
    G_DoNewGame ();
    break;
  case ga_loadgame:
    G_DoLoadGame ();
    break;
  case ga_savegame:
    G_DoSaveGame ();
    break;
  case ga_playdemo:
    G_DoPlayDemo ();
    break;
  case ga_completed:
    G_DoCompleted ();
    break;
  case ga_victory:
    F_StartFinale ();
    break;
  case ga_worlddone:
    G_DoWorldDone ();
    break;
  case ga_screenshot:
    M_ScreenShot ();
    gameaction = ga_nothing;
    break;
  case ga_nothing:
    break;
}
}

// get commands, check consistancy,
// and build new consistancy check
buf = (gametic/ticdup)%BACKUPTICS;

for (i=0 ; i<MAXPLAYERS ; i++)
{
  if (playeringame[i])
  {
    cmd = &players[i].cmd;

    memcpy (cmd, &netcmds[i][buf], sizeof(ticcmd_t));

    if (demoplayback)
      G_ReadDemoTiccmd (cmd);
    if (demorecording)
      G_WriteDemoTiccmd (cmd);

// check for turbo cheats
    if (cmd->forwardmove > TURBOTHRESHOLD
      && !(gametic&31) && ((gametic>>5)&3) == i )
    {
      static char turbomessage[80];
      extern char *player_names[4];
      sprintf (turbomessage, "%s is turbo!",player_names[i]);
      players[consoleplayer].message = turbomessage;
    }

    if (netgame && !netdemo && !(gametic%ticdup) )
    {
      if (gametic > BACKUPTICS
        && consistancy[i][buf] != cmd->consistancy)
      {
        I_Error ("consistancy failure (%i should be %i)",
                 cmd->consistancy, consistancy[i][buf]);
      }
      if (players[i].mo)
        consistancy[i][buf] = players[i].mo->x;
      else
        consistancy[i][buf] = rndindex;
    }
  }

// check for special buttons
  for (i=0 ; i<MAXPLAYERS ; i++)
  {
    if (playeringame[i])
    {
      if (players[i].cmd.buttons & BT_SPECIAL)
      {
```