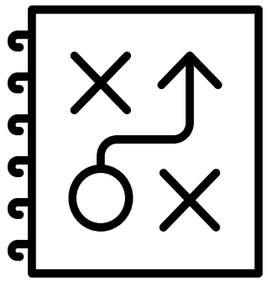
Mobile Malware Mimicking Framework

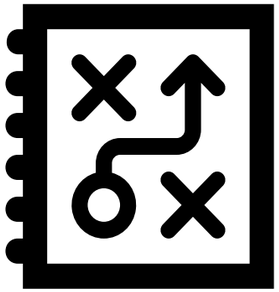
By Max ‘Libra’ Kersten

Table of contents

Motivation

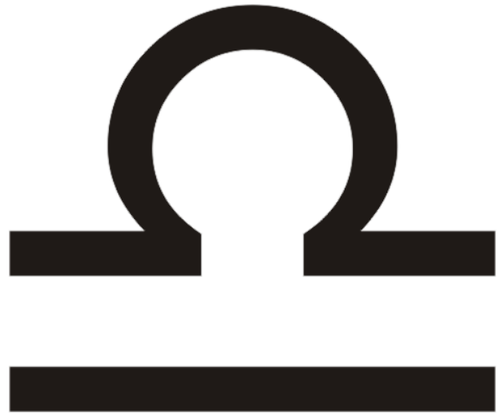
- Started as a private project
  - Should be on a budget
  - Easily extendible
- Unable to find an existing project
  - Android malware emulation is often overlooked
  - Complete virtualisation does not scale





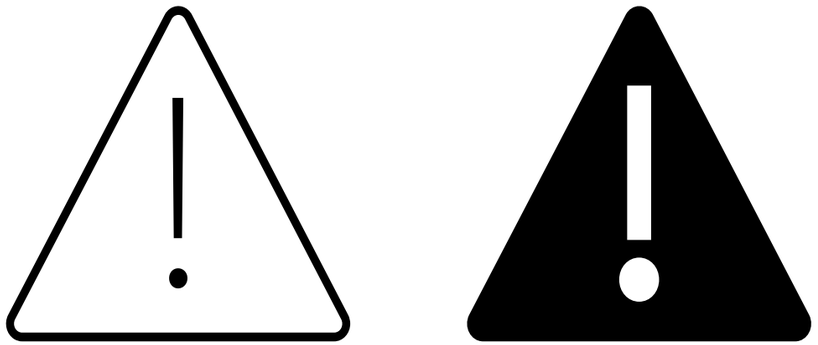
Who am I?

- Max ‘Libra’ Kersten (@Libranalysis)
- Malware analyst and reverse engineer
- Working for Trellix’ Advanced Threat Research team
- I write blogs about reverse engineering
  - Including my own free Binary Analysis Course
- Spoke at several conferences
  - Black Hat Europe, Botconf, atHack, CONFidence, and others
- My tools are open-sourced on Github
  - AndroidProjectCreator is such a project
  - As is the Mobile Malware Mimicking Framework



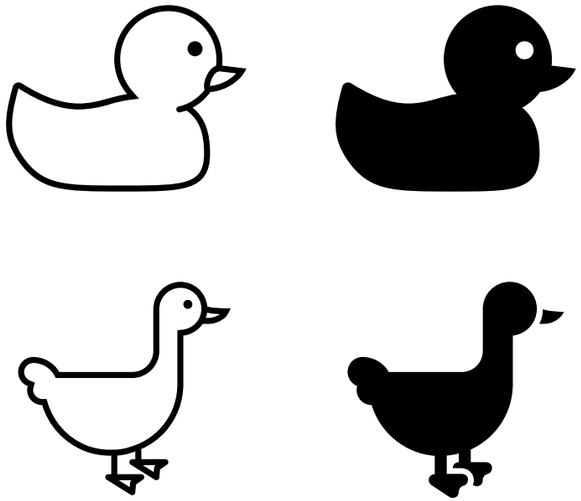
Disclaimer

- Emulated bots connect with actor owned servers
- Only use details that you fully own

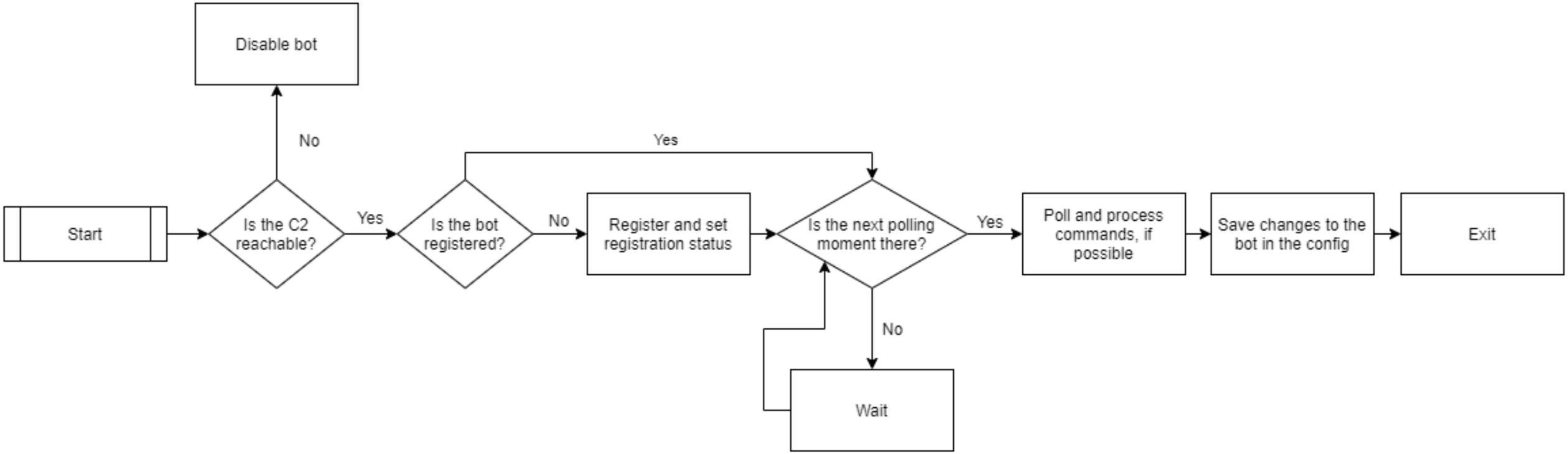


Emulation in short

- Emulating malware
  - Pretending to be a bot
  - Identical behaviour
  - Collect malware updates and actor behaviour
- Common problems
  - Virtualisation does not scale well
  - Incomplete access to the virtualised phone
  - Recreation of the internal bot structure

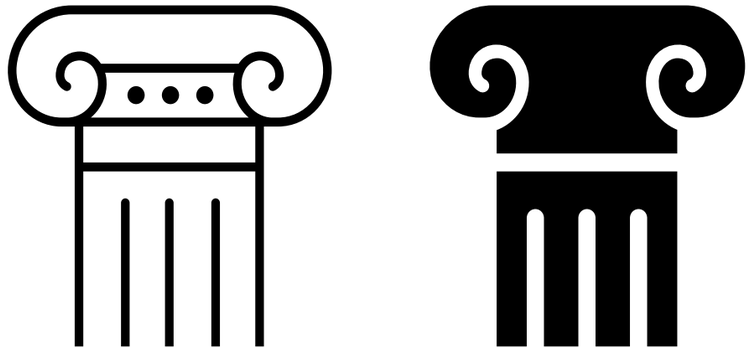


Emulation in short



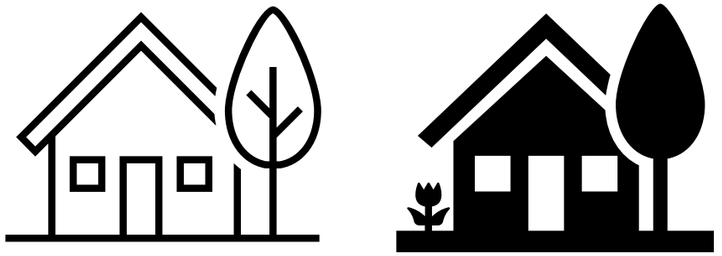
m3's features and usage

- Based on three pillars
  - Simplicity
  - Security
  - Scalability
- Bots can be made via
  - The command-line interface
  - A guided mode



m3's features and usage

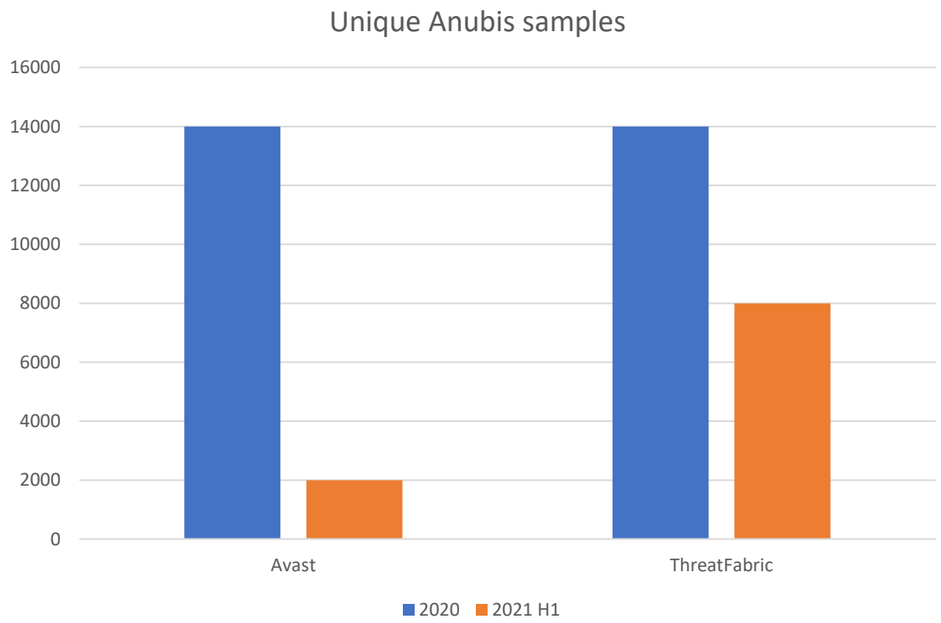
- Adding a family
  - Abstracted layers allow for easy integration
  - This blog provides detailed information
  - All code is documented using JavaDoc
    - Including inline documentation
- Understand the framework's main objects
- Use decompiled code to your advantage

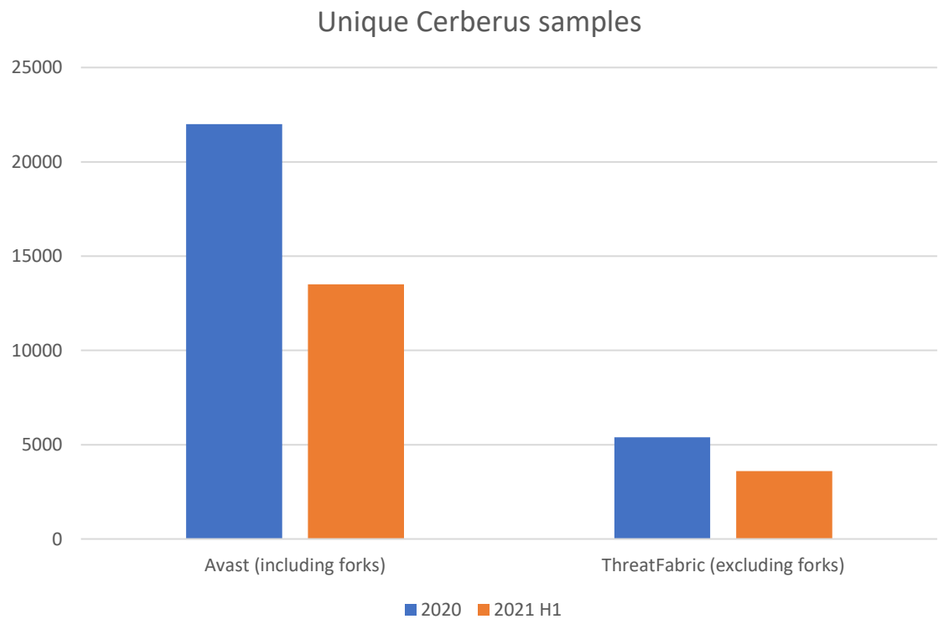


Ready-to-emulate malware families

- m3 contains two renowned families
  - Anubis
  - Cerberus
- Cerberus has two active forks
  - Alien
  - ERMAC

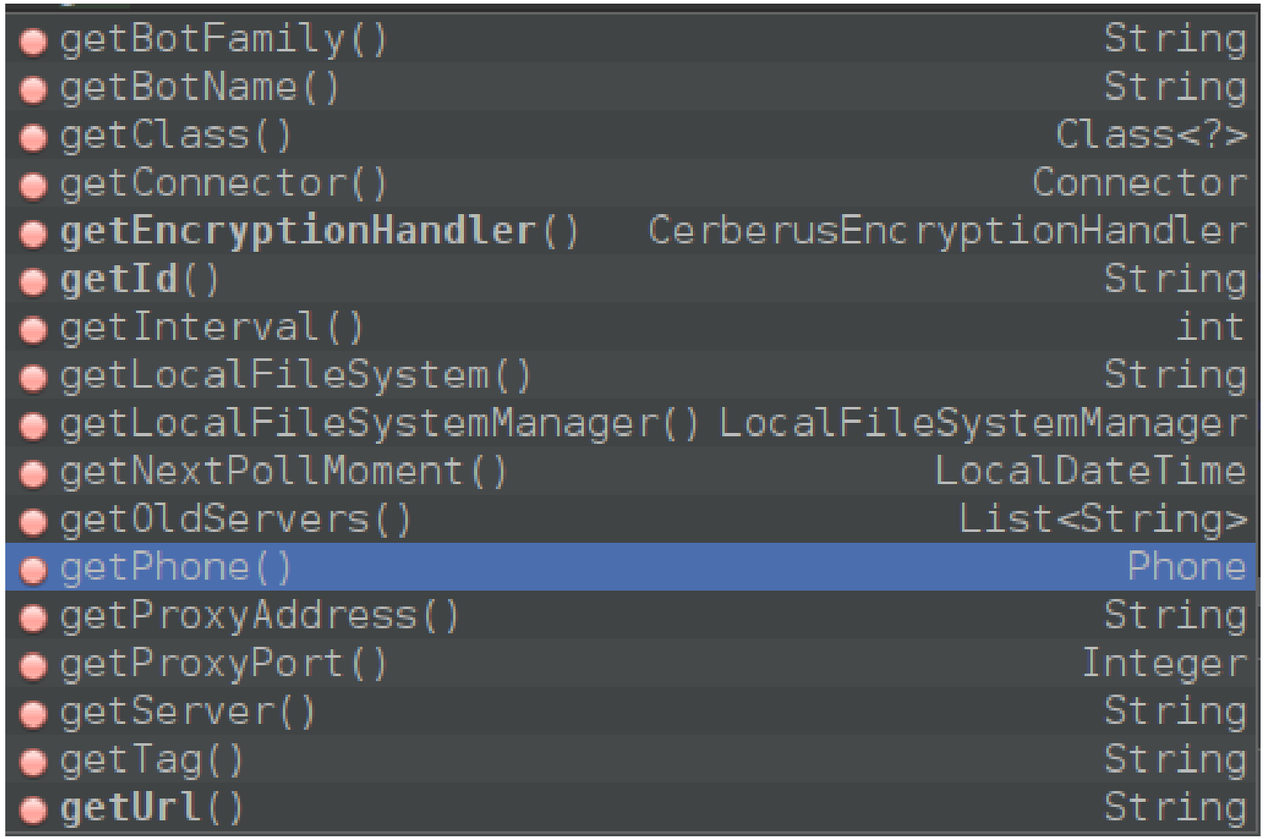
Ready-to-emulate malware families





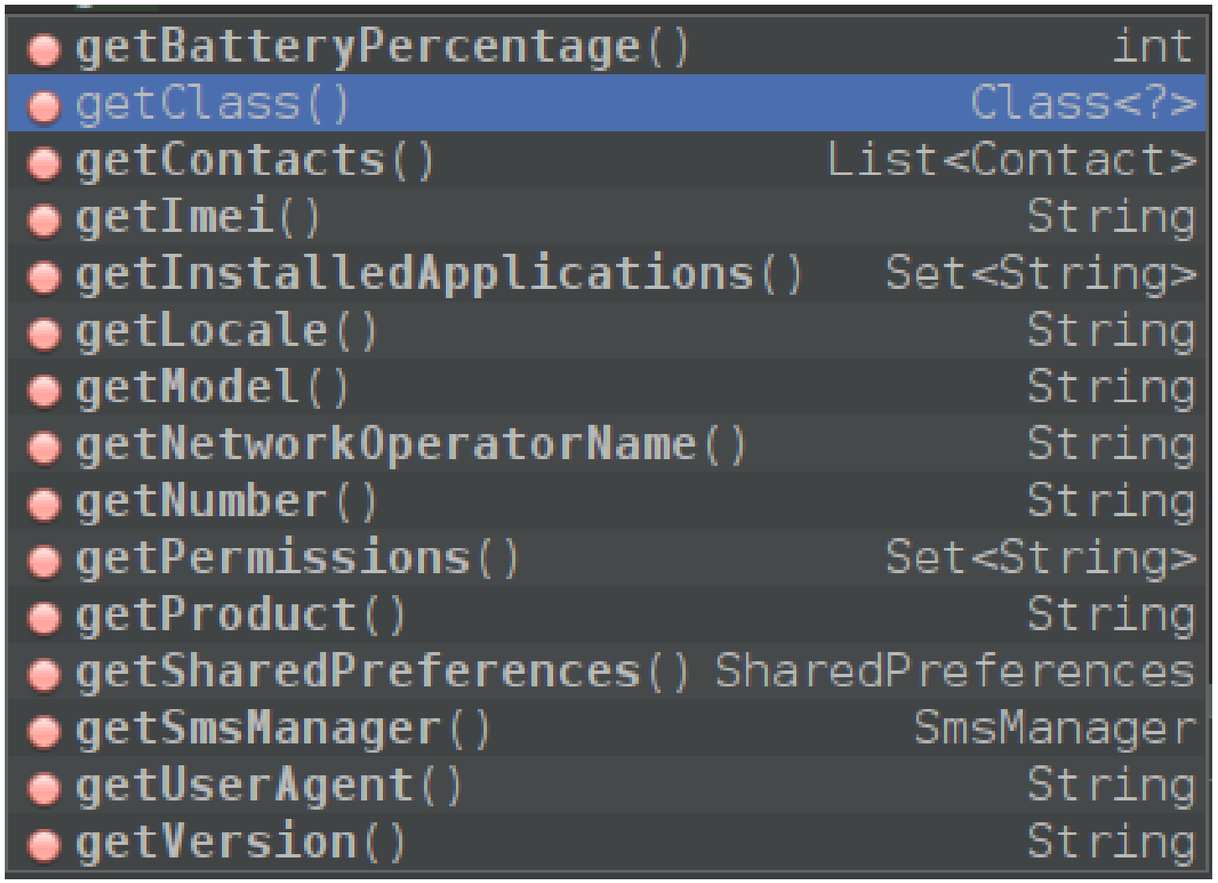
The Bot object

- Contains bot related data and functions
- Based on the `IBot` interface, and the abstract `Bot` class
- Helper classes can be shared between families (i.e. RC4)



The Phone object

- Contains all phone related data and functions
- Embeds other phone related objects



```
public String IDBot(Context context){
    String IDBot = Settings.Secure.getString(context.getContentResolver(), Settings.Secure.ANDROID_ID);
    return IDBot.equals("")?randomString(16):IDBot;
}

public String getScreen(Context context){
    KeyguardManager km = (KeyguardManager) context.getSystemService(context.KEYGUARD_SERVICE);
    boolean locked = km.inKeyguardRestrictedInputMode();
    if (!locked)
        return consts.str_1;
    else
        return consts.str_step;
}
public String getDeviceName() {
    String manufacturer = Build.MANUFACTURER;
    String model = Build.MODEL;
    if (model.toLowerCase().startsWith(manufacturer.toLowerCase())) {
        return capitalize(model);
    } else {
        return capitalize(manufacturer) + " " + model;
    }
}
```

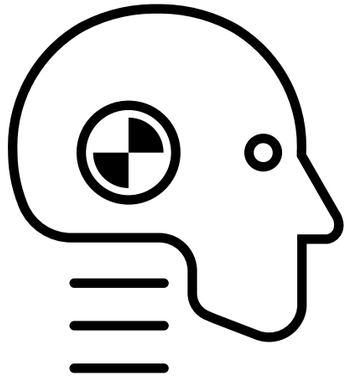
```
public String example(CerberusBot bot) {
    String id = bot.getId();

    boolean isScreenLocked = bot.getPhone().isLocked();

    String model = bot.getPhone().getModel();
    String manufacturer = bot.getPhone().getSharedPreferences().read("manufacturer");
    if (model.toLowerCase().startsWith(manufacturer.toLowerCase())) {
        return model.substring(0, 1).toUpperCase() + model.substring(1);
    } else {
        return manufacturer.substring(0, 1).toUpperCase() + manufacturer.substring(1) + " " + model;
    }
}
```

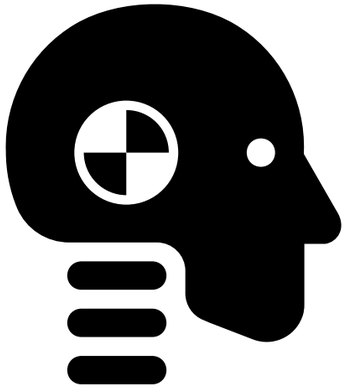
Special thanks

- ThreatFabric
  - Gaetan van Diemen
  - Bozman
  - Cengiz Han Sahin
- Avast
  - Ondřej David
  - Nikolaos Chrysaidos
- Fred HK

Demo

- Guided bot creation
- Command-line interface bot creation
- Bot emulation





Try it yourself!

- The documentation and source code are available
- The slides will be published and shared via @Libranalysis
- Questions can be asked on stream, in-person, or in a DM!