Distributed Platform for IM and Social Network Monitoring

Marcos Orallo Rodríguez
Juan Carlos Montes Senra

INTECO-CERT

May, 2009

27th TF-CSIRT Meeting, León

Agenda

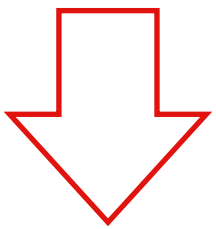
1. INTECO-CERT
2. Motivation & objectives
3. Features
    1. Monitoring
    2. Processing
    3. User Services
4. Technology
5. Architecture

1. INTECO-CERT

- **Information Services**:
  - ✓ Subscription to security reports, alerts
  - ✓ News, events
  - ✓ Online virus warnings, software vulnerabilities, spam.
- Training Services: Tutorials, manuals, online courses.
- Prevention Services: free tools, software updates.
- **Response and Support Services**:
  - ✓ Security Incidents management.
  - ✓ Malware infections.
  - ✓ Phishing attacks.
  - ✓ Legal support.
  - ✓ Security forums.

2. Motivation & objectives

- New vector for malware and spam propagation
- High popularity
- No publicly available tools



- Honeypot/Spimtrap for IM and social networks
- Analyze existing threats
- Early detection of new ones
- Flexible tool, adaptable to new sources

3. Features: Monitoring

- IM Networks
  - ✓ MSN
  - ✓ Yahoo
  - ✓ Google Talk
  - ✓ AIM
  - ✓ Skype
- Social networks
  - ✓ Twitter
  - ✓ Facebook
  - ✓ MySpace
  - ✓ Tuenti

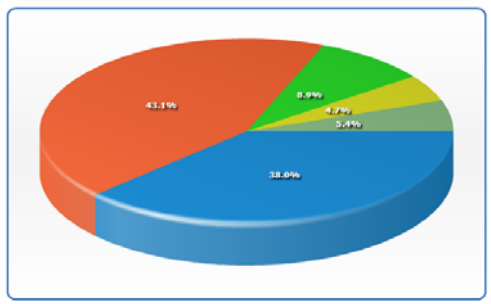
… mainly for **malware**, but also **spam**

3. Features: Processing

Parsing

- URLs
- File transfers
- Spimmer accounts
- Infected users

Aggregation



Query

- Data visualization
- User services

3. Features: Services for users

- Bot functionalities
  - ✓ Passive
    - Answer queries
    - Provide access to CERT services
  - ✓ Active
    - Security Alerts
    - Notification of infection
- Users contribute as sensors without drawbacks

4. Technology (I)

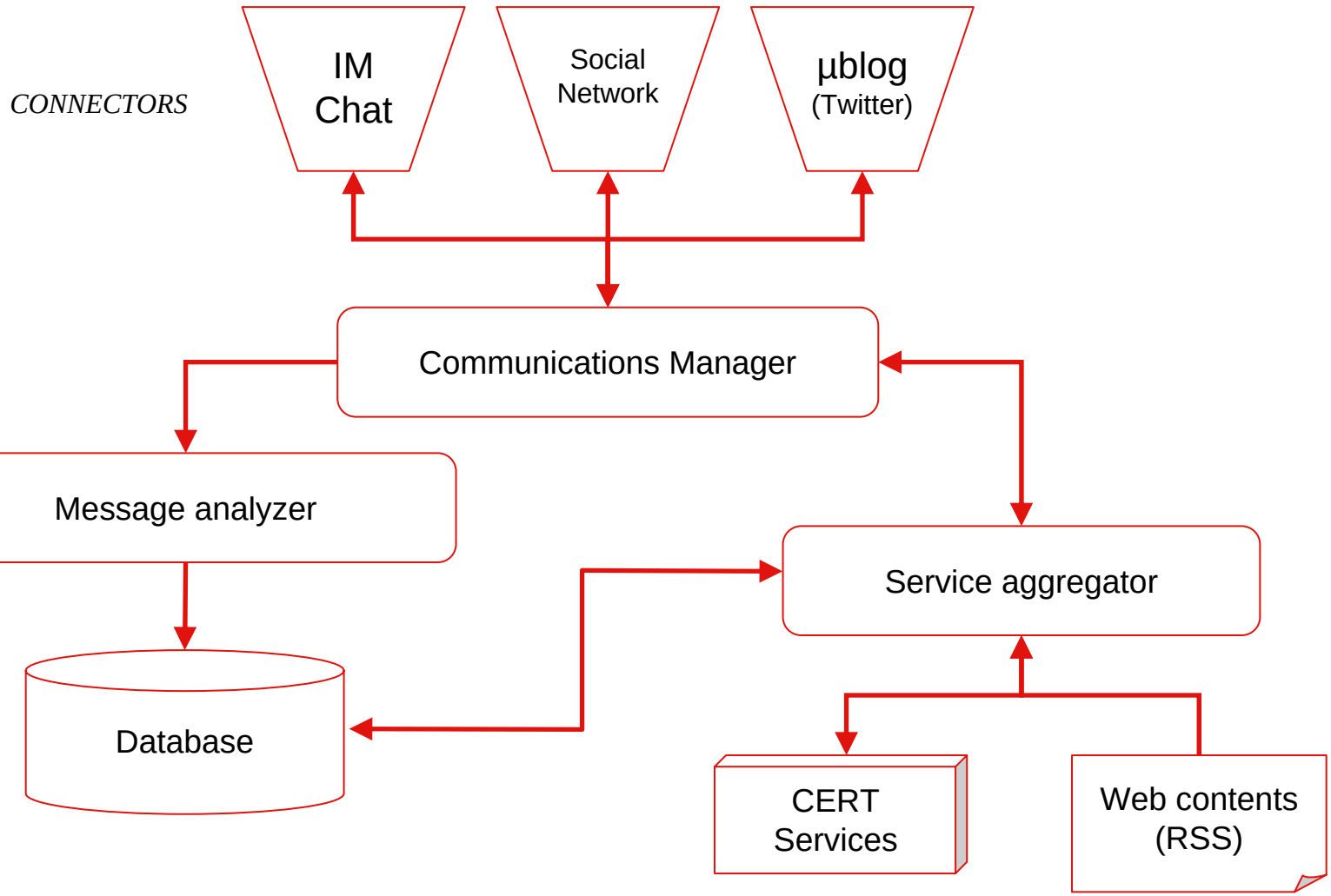
- Network Client
  - ✓ Python
  - ✓ Libpurple (*Finch*)
  - ✓ D-BUS
- Data management and visualization
  - ✓ MySQL
  - ✓ CakePHP

4. Technology (II)

poc.py

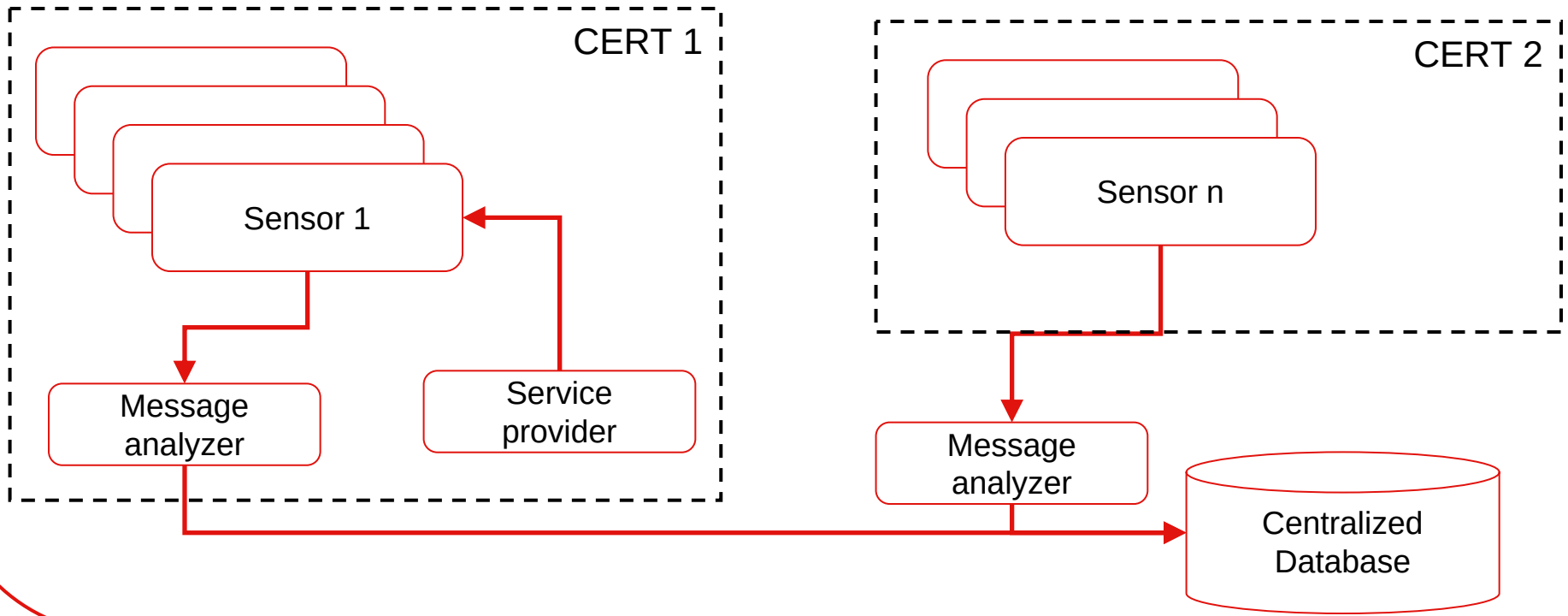
```
#!/usr/bin/env python
import os, subprocess
import dbus,gobject
from dbus.mainloop.glib import DBusGMainLoop

def print_message(account, sender, message, conversation, flags):
    print sender, "said:", message
#Connect to D-BUS
DBusGMainLoop(set_as_default=True)
bus = dbus.SessionBus()
bus.add_signal_receiver(
            print_message,
            dbus_interface="im.pidgin.purple.PurpleInterface",
            signal_name="ReceivedImMsg")
#Create the Finch process
finch_cmd = ['finch',
             '--config='+os.path.join(os.pardir,'.purple_config')]
out_file = file('output.log', "w")
finch_process = subprocess.Popen(finch_cmd, stdout = out_file)
#Start listening
loop = gobject.MainLoop()
loop.run()
```

5. Architecture (I)



5. Architecture (II)

- Distributed
  - ✓ Many IM accounts or *sensors*
  - ✓ Different services from different CERTs
  - ✓ Centralized database for data aggregation



Your turn!

- Questions?
- Suggestions?
- Criticisms or pitfalls?
- **Does anyone want to join?**

www.inteco.es