

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
  - XMPP as communication layer
  - Distributed Components
  - Connection Management
  - Privacy and Traceability
- Demo
- Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

# Towards Offensive Security Tools Integration

Fyodor Y.

Guard-Info

July 14, 2008

## Outline

Introduction  
Experience with Opensource Security Projects  
Offensive Security Tools Integration  
XMPP as communication layer  
Distributed Components  
Connection Management  
Privacy and Traceability  
Demo  
Downloads  
RFID tool release and small demo  
Acknowledgements  
questions and answers

Introduction

Experience with Opensource Security Projects

Offensive Security Tools Integration

XMPP as communication layer

Distributed Components

Connection Management

Privacy and Traceability

Demo

Downloads

RFID tool release and small demo

Acknowledgements

questions and answers

## Introducing Presenter

My name is Fyodor Yarochkin (often simply Fyodor Y.). I am not the nmap guy (snort faq Q1.2). I did some stuff for snort, xprobe and some other obscure public projects. I like to code and experiment with fun stuff.

# Content

- ▶ My past projects: snort, xprobe - what is the experience!
- ▶ Present: Offensive security tools: we'll discuss problems common to a typical user of offensive security tools. I will propose solution that could allow to integrate these tools into a distributed system.

- Outline
- Introduction
- Experience with Opensource Security Projects**
- Offensive Security Tools Integration
  - XMPP as communication layer
  - Distributed Components
  - Connection Management
  - Privacy and Traceability
- Demo
- Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

# Snort

- ▶ Been doing snort for about 2-3 years
- ▶ Managed snort mailing list at one time
- ▶ Worked on distributed snort concepts in 1999
- ▶ SnortNet - distributed IDS prototype in 2000

- Outline
- Introduction
- Experience with Opensource Security Projects**
- Offensive Security Tools Integration
- XMPP as communication layer
- Distributed Components
- Connection Management
- Privacy and Traceability
- Demo
- Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

## Snort: Lessons

- ▶ Gained community support due to ease of use and installation
- ▶ Pretty straight-forward syntax for snort rule creation
- ▶ Strong support via mailing list
- ▶ Remained opensource after Snort founder started a commercial company
- ▶ Still enjoys large popularity
- ▶ Does one thing but does it well.

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
  - XMPP as communication layer
  - Distributed Components
  - Connection Management
  - Privacy and Traceability
- Demo
- Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

## Snort: Issues

- ▶ Unmodular code. Hard to integrate in 3rd party projects (good or bad?)
- ▶ Hard to extend code with dynamical modules due to wide use of global variables, non-reentrant functions etc.
- ▶ The first multi-threaded snort implementation wasn't successeful (pcap didn't work with threads, sequential packet parsing logic)
- ▶ Multiple interface support depended on kernel support

- Outline
- Introduction
- Experience with Opensource Security Projects**
- Offensive Security Tools Integration
  - XMPP as communication layer
  - Distributed Components
  - Connection Management
  - Privacy and Traceability
  - Demo
- Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

## Xprobe: History

- ▶ Based on research by Ofir Arkin (called "X").
- ▶ First version of xprobe implemented probes in hardcoded form.
- ▶ Later versions of xprobe implemented dynamic modules
- ▶ Hardcoded logic was replaced with signatures.
- ▶ First to implement fuzzy-logic based fingerprinting methods.



- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
- XMPP as communication layer
- Distributed Components
- Connection Management
- Privacy and Traceability
- Demo
- Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

## Xprobe: Lessons learnt

- ▶ Project needs user community feedback/contributions to live.
- ▶ Interface for code/data contributions should be simple/straight-forward.
- ▶ Complex APIs/data structures should be replaced with simpler-to-use wrappers, data generators and so on.
- ▶ expect from your users to contribute as much as you need, but not more.

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration**
- XMPP as communication layer
- Distributed Components
- Connection Management
- Privacy and Traceability
- Demo
- Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

## Terminology

the Offensive Security Tools - computational tools designed for network attacking purposes.

- ▶ "Offensive" - 1. a. making attack b. relating to or designed for attack. (Merriam Webster dictionary).

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration**
- XMPP as communication layer
- Distributed Components
- Connection Management
- Privacy and Traceability
- Demo
- Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

## Purpose

What we are attempting to integrate opensource security tools into a system or framework, parts of which can be executed on different systems and the system functionality can be orchestrated from single spot, or a number of spots.

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration**
- XMPP as communication layer
- Distributed Components
- Connection Management
- Privacy and Traceability
- Demo
- Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

## Problems

- ▶ Reliable communication is needed
- ▶ Components are standalone programs that need to communicate
- ▶ Components and data are heterogeneous
- ▶ Coordination need
- ▶ Privacy and traceability concerns

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
- XMPP as communication layer**
- Distributed Components
- Connection Management
- Privacy and Traceability
- Demo
- Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

## Communication

We need a reliable communication protocol that could be proxied, routed, decentralized, has authentication mechanisms, transport layer security, in-bound file transfer ..

## XMPP protocol

so we switched to xmpp protocol

- ▶ robust messaging
- ▶ connections can go through http-proxy, socks5
- ▶ in-bound file transfer
- ▶ TLS support
- ▶ implementations exists for C, ruby, perl, python, lisp, java ..
- ▶ new tools adoption is extremely easy

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
- XMPP as communication layer**
- Distributed Components
- Connection Management
- Privacy and Traceability
- Demo
- Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

# XMPP

We'll need at least one Jabber server to connect to. We can (ab)use one of the public servers, or have ours. We can even route xmpp messages through other servers.

You can talk to your systems through GTALK.

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
  - XMPP as communication layer
  - Distributed Components**
  - Connection Management
  - Privacy and Traceability
  - Demo
  - Downloads
  - RFID tool release and small demo
  - Acknowledgements
  - questions and answers

## Distributed Components

what to integrate?

- ▶ something to do scans?
- ▶ something to build connections on demand to compromised hosts?
- ▶ something to manage knowledge



## Knowledge Machine

Knowledge Machine was picked up for knowledge storage/management function.

- ▶ The original Knowledge machine was written in lisp by Utehas.edu: <http://www.cs.utexas.edu/~mfkb/km.html>
- ▶ cl-xmpp was used to integrate the system
- ▶ the original KM was missing some components of knowledge objects versioning, merge/split operations and so on Is being modified as part of research process

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
- XMPP as communication layer
- Distributed Components**
- Connection Management
- Privacy and Traceability
- Demo
- Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

## KM usage

We use Knowledge Machine (KM) to store all collected data in form of First-Order Predicates We define Ontology of classes that describe basic objects that we work with: hosts, ports, networks, applications, users, urls, telephone numbers, relationships between hosts, networks, users and so on.

The ontology can be extended **real-time** as new types of data appears.

The purpose of building such KB is to be able to represent **ALL** knowledge regarding target organization in easily queryable, searchable manner.

## KM inference algorithm

Uses "rewrite" rules to decompose expressions into "basic" logic expressions

KM: (the ip-address of (the peers of HostA))

$\Rightarrow$  (the peers of HostA)  $\Rightarrow$  find (the ip-address of x)

LOGIC:  $y \text{ — peers(HostA,x), ip-address(x,y)}$

$\Rightarrow$  find x in peers(HostA,x)  $\Rightarrow$  find y in ip-address(x,y)

If instance changes, classification recheck (recalculation).

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
  - XMPP as communication layer
- Distributed Components**
- Connection Management
- Privacy and Traceability
- Demo
- Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

## Other useful KM features

- ▶ Automatic classification
- ▶ Data unification
- ▶ Sophisticated data mining
- ▶ Ability to save snapshots and restore from certain snapshot points.

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
  - XMPP as communication layer
- Distributed Components**
- Connection Management
- Privacy and Traceability
- Demo
- Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

## Impacts of using KM

- ▶ All data has to be presented in lisp-like format
- ▶ Code = data (very useful feature, but possible security impacts)

So we are porting other tools to present their data in lisp form (examples below).

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
- XMPP as communication layer
- Distributed Components**
- Connection Management
- Privacy and Traceability
- Demo
- Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

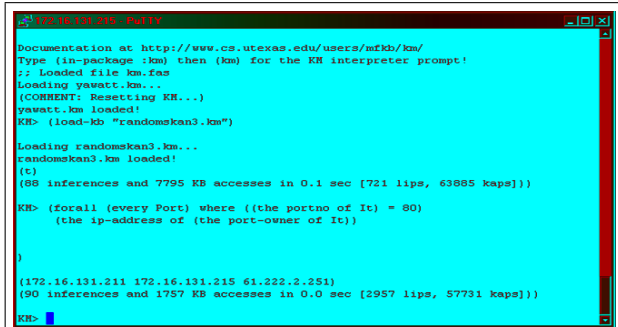
## Examples with Network scanning data

We patched nmap and xprobe to produce data in lisp-like format.  
So here what you'd have:

```
i:: Nmap 4.2250C2 scan initiated Tue Jul 31 16:17:52 2007
oK randomness3.km
(a Host with
  (ip-address (172.16.131.1))
  (hostname ())
  (ports (
    (a Port with
      (portno (21))
      (status (open))
      (protocol (tcp))
      (ip ((the ip-address of Self)))
      (owner ())
      (service ("ftp"))
      (rpcinfo ())
      (software (""))
    )
  )
  (a Port with
```

## Queries

After the data is in KM you can run your queries on it



```
172.16.131.215 - PuTTY
Documentation at http://www.cs.utexas.edu/users/mfkb/km/
Type (in-package :km) then (km) for the KM interpreter prompt!
:: Loaded file km.fas
Loading yawatt.km...
(COMMENT: Resetting KM...)
yawatt.km loaded!
KM> (load-kb "randomskan3.km")

Loading randomskan3.km...
randomskan3.km loaded!
(t)
(88 inferences and 7795 KB accesses in 0.1 sec [721 lips, 63885 kaps])

KM> (forall (every Port) where ((the portno of It) = 80)
      (the ip-address of (the port-owner of It))
)

(172.16.131.211 172.16.131.215 61.222.2.251)
(90 inferences and 1757 KB accesses in 0.0 sec [2957 lips, 57731 kaps])
KM>
```

## Sample KM queries

```
(forall (forall (every Port) where ((the portno of It) = 80) (the  
port-owner of It)) (showme It))  
(forall (every Port) where ((the portno of It) = 80) (showme It))  
(forall (every Port) where ((the portno of It) = 80) (the ip-address  
of (the port-owner of It)) (make-sentence (the full-address of (allof  
(every Port) where ((the portno of It) = 80)))) )  
(forall (the all-instances of Host) (if (the operating-system of It)  
then (the ip-address of It)) )
```



## Connection forwarding

- ▶ we want to talk to hosts behind firewall.
- ▶ once we have our node running in target network, we want to be able to access other systems too.
- ▶ NATs, default-deny firewall rules make it harder

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
- XMPP as communication layer
- Distributed Components
- Connection Management**
- Privacy and Traceability
- Demo
- Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

## Pbounce over XMPP

pbounce is the tool that supports connection "pivoting" using remote drones that connect to controlling node.

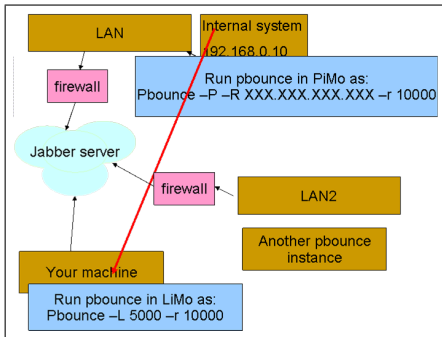
New pbounce version supports connection tunneling over XMPP

New pbounce works in "many" to "many" form. That means you can have multiple controlling nodes that can forward connection from a number of remote drones (pivot drones). Like-wise pivot drones can forward connections to a number of control nodes simultaneously.

KM can be used to resolve "right drone to talk" finding issues

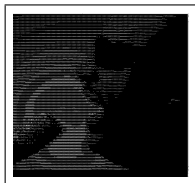
- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
- XMPP as communication layer
- Distributed Components
- Connection Management**
- Privacy and Traceability
- Demo
- Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

# pbounce architecture



- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
  - XMPP as communication layer
  - Distributed Components
  - Connection Management**
  - Privacy and Traceability
  - Demo
  - Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

## pbounce architecture



our local pr0n-manager Grugq had another fantastic idea - SSH connection forwarding using the similar model.

A drone collects credentials of compromised ssh boxes and can provide connections on demand.

Saves your fingertips. :)

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
  - XMPP as communication layer
  - Distributed Components
  - Connection Management**
  - Privacy and Traceability
  - Demo
  - Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

## Coordination with meta agents

In addition to KM you may have Meta-agents - agents that capable of resolving complex queries into simpler ones. (so you can tell it "hax0r network X" and it knows what and where to run, what data to send to whom and so on.

This is still research project :)

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
- XMPP as communication layer
- Distributed Components
- Connection Management
- Privacy and Traceability**
- Demo
- Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

## Anonymisation of Jabber

I've been experimenting with running Jabber servers as hidden TOR services.

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
  - XMPP as communication layer
  - Distributed Components
  - Connection Management
  - Privacy and Traceability
- Demo**
- Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers

# Demo

Now comes the demo

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
  - XMPP as communication layer
  - Distributed Components
  - Connection Management
  - Privacy and Traceability
  - Demo
- Downloads**
  - RFID tool release and small demo
  - Acknowledgements
  - questions and answers

## Code availability

<http://o0o.nu>



## RFID

After seeing Adam Laurie's talk I got inspired enough to play with



some RFID stuff. I got this toy from GIGA-TMS.

- ▶ used protocol GnetPlus - variation of modbus protocol
- ▶ interface type - usb (usb2serial) or direct rs232

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
  - XMPP as communication layer
  - Distributed Components
  - Connection Management
  - Privacy and Traceability
  - Demo
  - Downloads
- RFID tool release and small demo**
- Acknowledgements
- questions and answers

## RFID small demo

I've wrote small set of tools <http://o0o.nu/rfid> to support some basic operations such as key cracking and block read/writing on this device.

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
  - XMPP as communication layer
  - Distributed Components
  - Connection Management
  - Privacy and Traceability
- Demo
- Downloads
- RFID tool release and small demo
- Acknowledgements**
- questions and answers

## Acknowledgements

- ▶ Grugq - xmpp was his wonderful discovery. :)
- ▶ Halvar - lots of talks and ideas :)
- ▶ Mederych - alot of code is by his sweat ;)

- Outline
- Introduction
- Experience with Opensource Security Projects
- Offensive Security Tools Integration
  - XMPP as communication layer
  - Distributed Components
  - Connection Management
  - Privacy and Traceability
  - Demo
  - Downloads
- RFID tool release and small demo
- Acknowledgements
- questions and answers**

## Questions and answers

Shot yours ;-)