Distack
Towards Understanding the Global Behavior of DDoS Attacks
– A Framework for Distributed Attack Detection and Beyond –

Thomas Gamer, Christoph P. Mayer, Martina Zitterbart
29. Aug 2008, EURECOM, France

Institute of Telematics, University of Karlsruhe (TH)
Karlsruhe Institute of Technology (KIT)
DDoS – Huge threat to the Internet

„New Zealand teenager controlled botnet of 1.3 million computers“ (Heise-Online, Nov. 2007)

„DDoS attacks and worms pose biggest threat to the Internet“ (Worldwide Infrastructure Security Report, Arbor Networks, 2007)

How can you detect and block such low traffic early?

→ Cooperation between detection instances seems promising!
On the way to Distributed Attack Detection

Identification of attacks: anomalies → kind of attack

Management and Visualization

PktAnon

distack

distributed attack detection

ReaSE

Local traffic analysis and attack detection

Large-scale simulations

Real traffic/traces

Institute of Telematics
University of Karlsruhe

T. Gamer, C. Mayer, M. Zitterbart
Distack – Framework for Distributed Large-scale Attack Detection

- Distributed Attack Detection
- Evaluation
- Cooperation of distributed detection instances
- Identification of attacks: anomalies
- Self-organization of local detection
- Large-scale simulations
- Real traffic/traces
- Management and Visualization

ReaSE
PktAnon
Distack – Framework for Distributed Large-scale Attack Detection

distack
distributed attack detection
Some exemplary issues

- Little knowledge about global behavior of DDoS
- Attacks highly distributed. Attack detection and countermeasures mostly not!
- Few directly reusable results

Initial challenge:
Complex development and evaluation of mechanisms for local and distributed attack detection and traffic analysis

→ Initial development effort as base for your mechanisms is incredibly high!
What you can do with Distack

- **Attack detection and traffic analysis**
  - Rapidly implement and run your attack detection and traffic analysis schemes
  - Lots of reusable modules (e.g. sampling, plotting)
  - Run on live traffic or captured traces
  - Comfortable communication between remote instances → easier distributed detection

- **Simulations**
  - Run your modules transparently in large-scale simulations
  - Integrates seamlessly with the toolkit OMNeT++/INET/ReaSE

and that`s not even all …
Distack use-cases

Examples
- *Local traffic analysis:* easily analyze online traffic and traffic traces
- *Distributed traffic analysis:* several measurement points in the network, report to a central instance

→ There is more than distributed attack detection!
Distack: Distributed attack detection

Framework for distributed attack detection and traffic analysis

What it gives to *you*

- Fully **concentrate** on your methods for attack detection and traffic analysis
- **Write once** run everywhere: Transparently run your methods, e.g. on a PC or in a simulation environment
- **High reuse** through building blocks
- **Great support** for your attack detection
Rough Architectural Overview

- **Module manager**
  - Mechanisms are implemented in small building blocks → *modules*
  - The environment to implement your modules

- **Network manager**
  - Abstraction from the network
  - Handles the different ways packets come in

- **Local and remote messaging**
  - Communication for the lightweight modules
  - Data-centric communication, local and remote

- **Configuration**
  - Flexible way to configure your modules and Distack
Distack High-level Architecture

ModuleManager
- MessagingSystem
- ChannelManager
- Frame distribution system

Remote Messaging
- Serialization
- Deserialization
- Destinations
- Sources

Utilities
- Level based Logging
- Conversion
- Timer
- Structures
- String Operations
- Routing Table

Communication
- GIST
- Sockets API
- ...
**Lightweight Modules**

- **Modules**: implement well-defined functionality
  - Small building blocks for high reuse
  - Loaded at runtime on demand
  - Easily configurable (next slide)
  - Perform packet inspection ... or other tasks
  - this is where you implement your mechanisms!

- **Channels**: linear linked modules
  - Create more complex functionality

```
Channel A
Sampling → Monitoring → Plotting

Channel B
Protocol filter → Statistics
```
Packet Processing in Distack

ModuleManager

MessagingSystem

Remote Messaging

Utilities

Utilities

Protokoll-Parser

ChannelManager

Frame distribution system

NetworkManager

Protokoll-Parser

ChannelManager

NetworkInterface

OMNeT++

Live

Traffic Traces

GIST

Sockets API

OMNeT++

Live

Traffic Traces

GIST

Sockets API

OMNeT++

Live

Traffic Traces

GIST

Sockets API

Serialization

Deserialization

Destinations

Sources

Conversion

Timer

Structures

String Operations

Routing Table

...
Flexible Configuration

How can I configure my modules?

Module instantiation and configuration

- Can use module libraries multiple times with different configuration!

Channels and actual use of modules

- Flexible grouping of small modules into larger functionality!
Communication

- Modules are lightweight, small, decoupled
  - Enables high reuse, but how can they interact?

- **Data-centric communication** between modules
  - Modules register for message they are interested in
    - Modules send out messages
    - Messages delivered to registered modules
  - Module: `Hmm … interesting information I got here … maybe someone is interested in this` → send

- **Remote communication as easy as local**
  - Send messages locally, remotely, or both
  - Transparent message distribution to remote Distack instances

```c
MessageSynAckBalance msg(291, 33);
sendMessage(msg, REMOTE_DESTINATIONS_NEIGHBOURS);
```
Distrack abstracts from traffic sources
- **Live traffic**: buffers handle bursty traffic
- **Traffic traces**: replayed with original timing
- **Simulated traffic**: packet transformation for OMNeT++

Easy and consistent packet access
- Traffic live, replayed, or simulated … you don‘t care!
- Easy and safe access to protocol parsers

```cpp
TcpPacket* tcp = ippacket->getNextPacket();
if(tcp->isFlagSet(TcpPacket::TCP_FLAG_SYN))
    port = tcp->getDestport();
```

Supported protocols
- Ethernet, ARP, ICMP, IPv4, IPv6, MPLS, TCP, UDP
- More to come. Easy to implement your own!
Integration into simulations

- Few simulations of DDoS attacks and detection

  In our opinion the key to understand the global and distributed behavior of DDoS attacks

- Our simulation toolkit
  - OMNeT++: time discrete simulation environment
  - INET Framework: lots of protocols (TCP, UDP, …)
  - ReaSE: topology, self-similar traffic generation, DDoS zombies

- Distack is integrated into this toolkit
  - Packet formats
    - Transparent transformation into Distacks protocol parsers
  - Time domain
    - The simulation time runs different!
  - Modules source code compatible
    - just need to recompile …
Distack is real!

Everything presented here is *running code*!

- Go and **implement some modules**
  - Try it out! E.g. analyze a trace file
  - Use the communication between remote instances
  - There are already several modules available

- Go and do a **large-scale simulation**
  - Could be DDoS, could be somethings else
  - Find out how easy Distack makes your life!
  - Integrates with ReaSE → coming soon in this talk
What we are doing with Distack

Collaborative Anomaly-based Attack Detection
Thomas Gamer, Michael Scharf, and Marcus Schöller,
Proceedings of 2nd International Workshop on Self-Organizing

A System for in-Network Anomaly Detection
Thomas Gamer, Kommunikation in Verteilten Systemen,
What we are doing with Distack

Management and Visualization
Distributed Attack Detection

Evaluation

Identification of attacks: detection of anomalies

Self-organization of local detection

Large-scale simulations

Real traffic/traces

Management and Visualization

PktAnon

ReaSE – Realistic Simulation Environment

ReaSE

distack

distributed attack detection
We want …
- to understand the global behavior of DDoS attacks
- evaluate our mechanisms implemented in Distack
  → on a large-scale!

Using real systems is extremely costly!
- Where to get e.g. 10,000 machines from?
- Use a real network? We will execute DDoS attacks!

Simulations
- Topologies that match today's Internet infrastructure
- Realistic background traffic, malicious DDoS traffic
Network Topology

- How does today's Internet topology look like?
  - **Power-law distribution in node degree**
    - Lots of nodes with low node degree
    - Few nodes with high node degree
  - **Hierarchical structure**
    - Autonomous Systems (stub/transit) with routers
    - Based on Zhoua et al. ICCCAS06, Li et al. SIGCOMM04

![Diagram showing network topology with stubAS and transitAS]
Network Traffic

- **Legitimate** traffic as well as...
  - Self-similar behavior
    - Heavy-tailed ON/OFF intervals as well as packet sizes
  - Reasonable mix of different kinds of traffic
  - **Traffic profiles** define flow behavior

- ... **malicious** traffic
  - Evaluate the attack detection system
  - Used real-world tools and ported their behavior
    - DDoS attacks: Tribe Flood Network
    - Worm propagations: Code Red v1
ReaSE combines topology and traffic generation for realistic simulation environments.
- Based on up-to-date solutions
- Includes generation of malicious traffic
- Integrates with OMNeT++ and INET Framework

GUI helps to …
- create topologies
- define traffic profiles
PktAnon – Traffic Anonymization

Evaluation

Identification of attacks:
- Anomalies
  - Kind of attack

Self-organization of local detection

Large-scale simulations

Real traffic/traces

Management and Visualization

PktAnon

ReaSE

Local traffic analysis and attack detection

Cooperation of distributed detection instances

distack distributed attack detection

PktAnon – Traffic Anonymization

T. Gamer, C. Mayer, M. Zitterbart
Institute of Telematics
University of Karlsruhe
www.tm.uka.de
How it all began: *We wanted to record network traffic at an ISP gateway to evaluate the attack detection mechanisms* …

- **Anonymization of network traffic**
  - Sharing recorded traffic traces with third parties
  - Legal reasons, protect users, protect your network infrastructure, …

- **Existing tools not flexible enough**
  - Have been built out of a specific need
  - **PktAnon is generic and fully flexible!**
Anonymization profiles

- **Anonymization profiles**
  - Allow complete flexibility in the anonymization
    - every protocol field can be anonymized!

```xml
<TcpPacket>
  <TcpSourceport anon=AnonHashSha1/>
  <TcpSegnum anon=AnonIdentity/>
  ...
</TcpPacket>
```

- **Even more flexibility**
  - Input and Output piping (live anonymization!)
  - Output traces well-formed (checksum, length field)
  - Many anonymization primitives and protocols

- **Current and outlook**
  - FreeBSD package, liveHEX security CD, OpenPacket.org
  - Call to the community for defining standardized anonymization profiles with different security levels
Summary and Conclusion

Road towards distributed attack detection and understanding the global behavior of DDoS is stony!

- We have developed tools to flatten this way
  - Did not build them the way *we* needed them
  - Built them **generic** and **flexible**
  - Now they simplify our daily work
    …and they can simplify *your* work!

distack
Framework

ReaSE
Realistic Simulation Environment

PktAnon
Profile-based Packet Anonymization
Thank you for your Attention!

Distack – A Framework for Large-scale Anomaly-based Attack Detection
Thomas Gamer, Christoph P. Mayer, and Martina Zitterbart
SECURWARE08, Cap Esterel, France, Aug 2008.

ReaSE
Realistic Simulation Environments for IP-based Networks

PktAnon – A Generic Framework for Profile-based Traffic Anonymization
Thomas Gamer, Christoph P. Mayer, and Marcus Schöller
PIK 2-2008 (Journal), Apr 2008.