Tracing Attacks on Advanced Persistent Threats in Networked Systems

Hiroshi Koide
Kyushu Institute of Technology, IPA

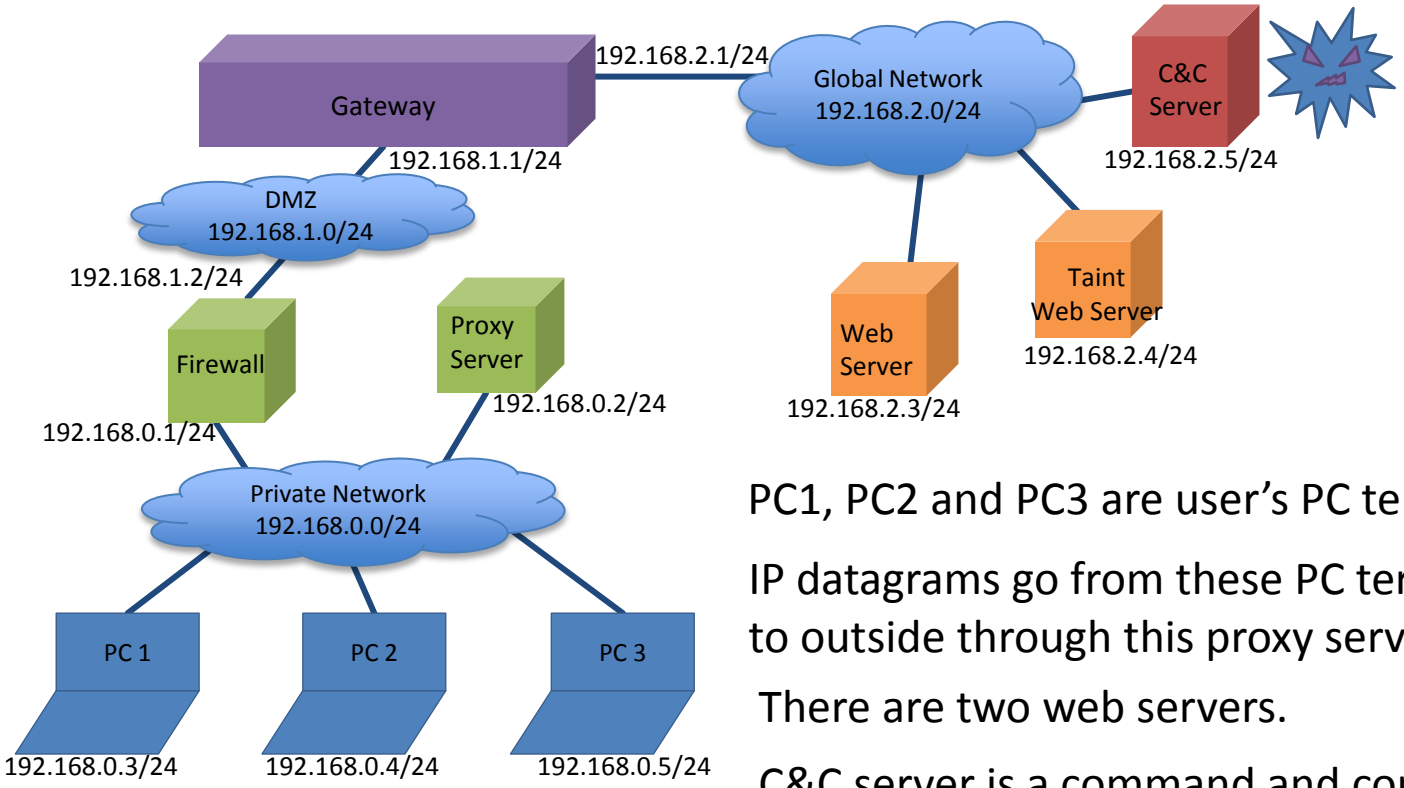
FIRST TC @ KYOTO
Kyoto 2012 FIRST Technical Colloquium
16:00-16:45, 14 November 2012
Kyoto City International Foundation Room 1&2

Demo or die

We introduce the demonstration of the prototyping of simulation engine we are proposing first, because we are thinking demonstration is the most important.

Proof of concept implementation

- We are developing a prototype of the simulator engine to analysis the behavior of malware on networks.
- We prepare sample networks. They are very simple at this time.
- Sample malware emulates behaviors of a part of Gamblar.



PC1, PC2 and PC3 are user's PC terminals.

IP datagrams go from these PC terminals to outside through this proxy server.

There are two web servers.

C&C server is a command and control server.

Background

- Cyber attack methods are changing. Defense is getting more and more difficult.
  - DDoS attacks
  - Web defacement
  - APT (Advanced Persistent Threats)  recent

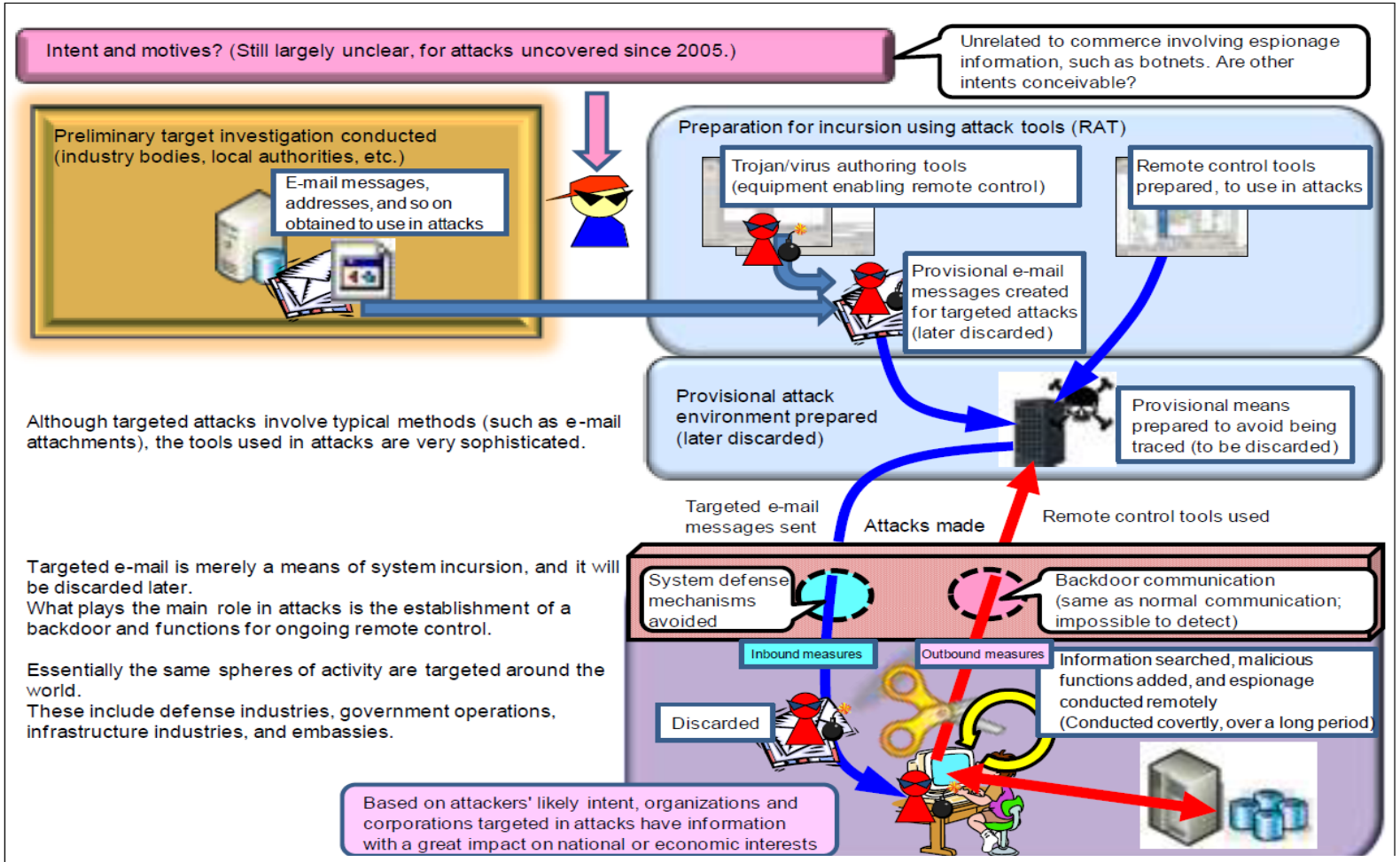
- Information system structures are becoming increasing complex and large in scale.
- It is difficult to get to know whether all systems are operating correctly or not.
  - Load balancer
  - Cloud computing
  - Overlay networks  recent

APT/Background

- APT incidents are increasing. Many organizations are attacked.
  - Operation Aurora (2010)
  - Night Dragon (2011)
  - Operation Shady RAT (2011)

.. are sophisticated cyber attack examples.

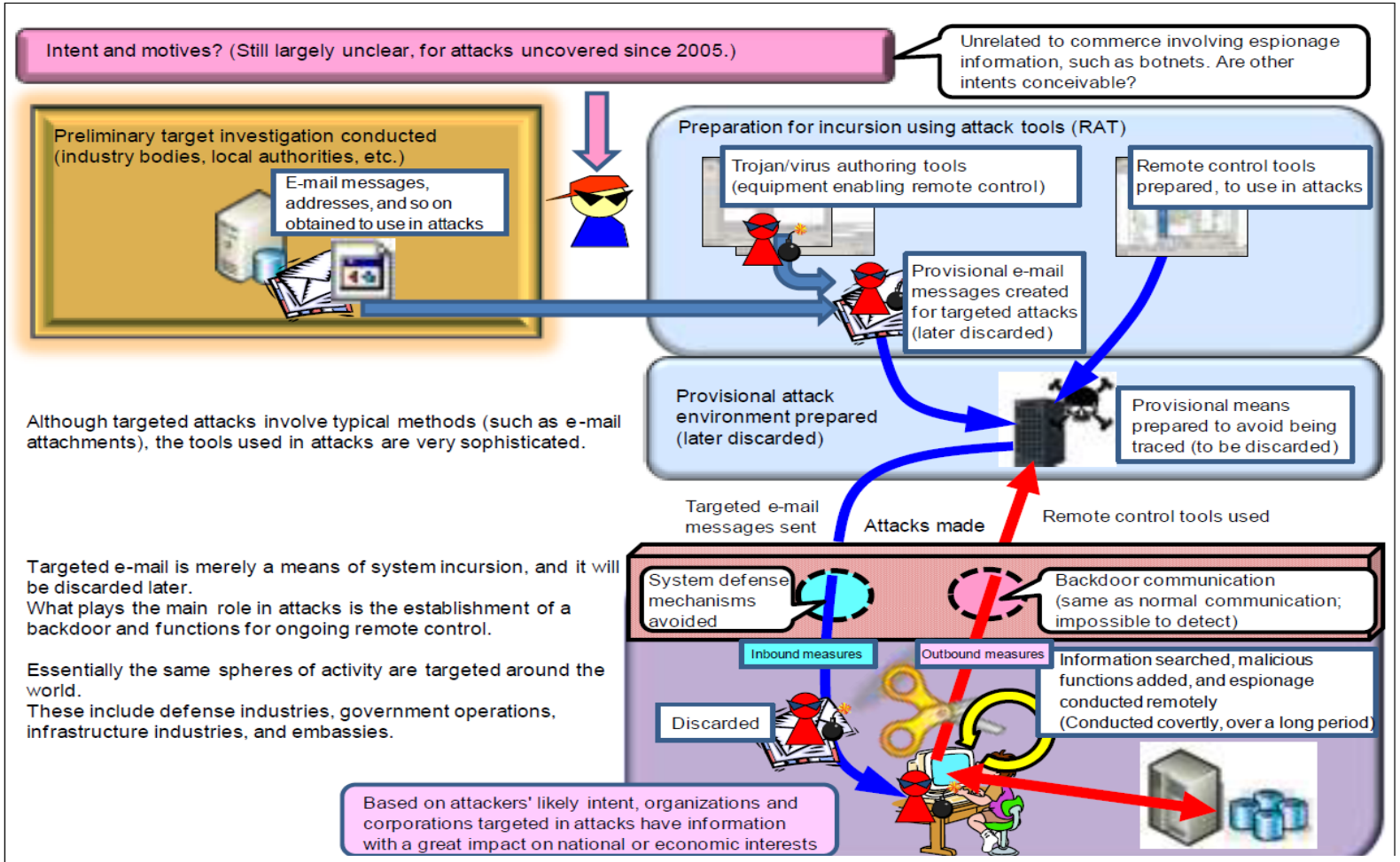
A typical sequence of APT attack



Design and Operational Guide to Protect against “Advanced Persistent Threats” , IPA, 2011

First, an attacker investigates e-mail messages, addresses and other information in preliminary cyber espionage.

A typical sequence of APT attack

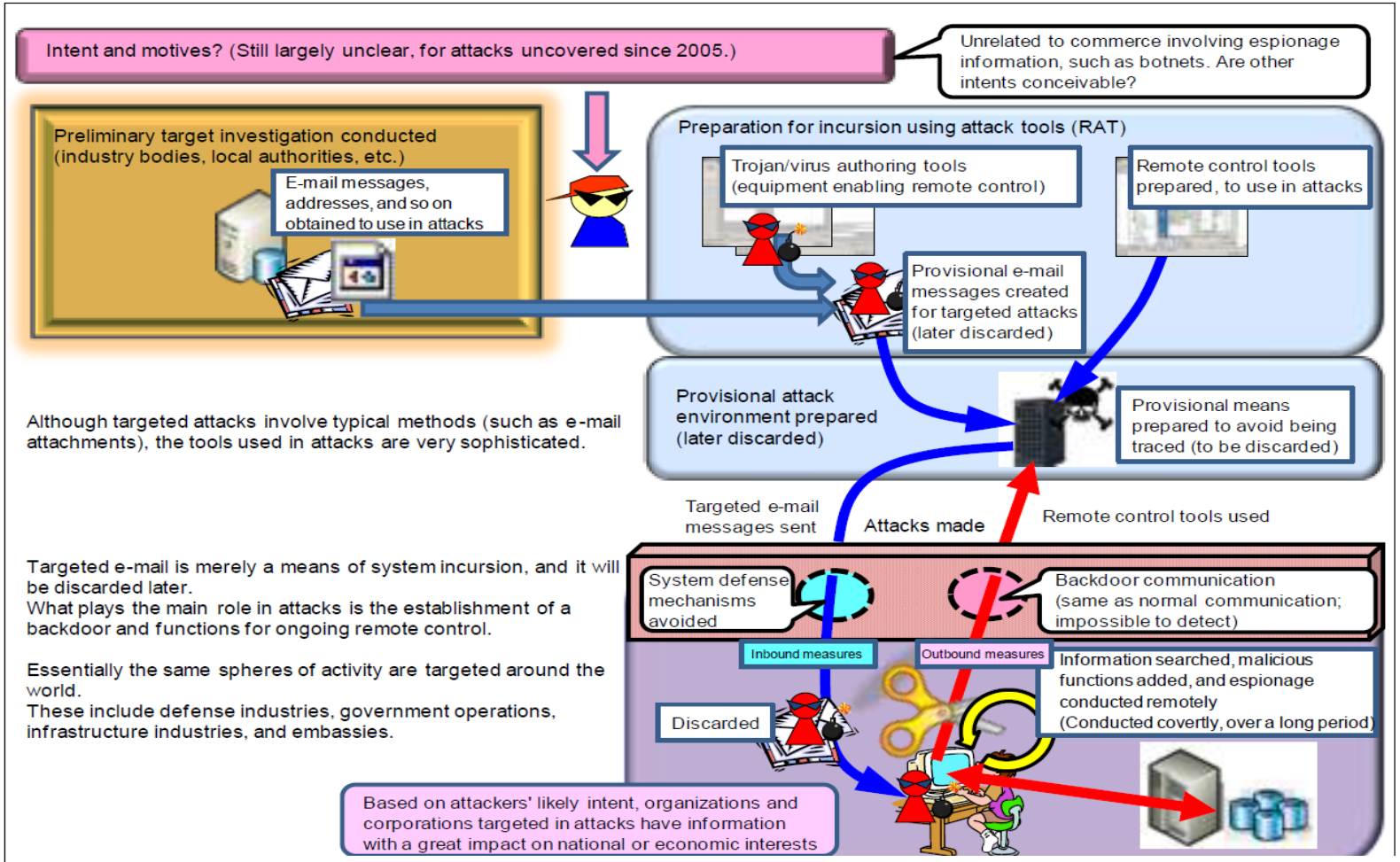


Design and Operational Guide to Protect against “Advanced Persistent Threats” , IPA, 2011

And the attacker tries to invade one of PC terminals in the targeted system by using these information.

At that time the attacker uses targeted e-mail containing remote control tools, RATs.

A typical sequence of APT attack



Design and Operational Guide to Protect against “Advanced Persistent Threats” , IPA, 2011

The attacker can control the PC from outside. And he or she collects confidential information in the storage system of the organization.

Why is it difficult to protect ?

- Malware can pass through initial defense points. They can break into the safety zone through the security boundary.
  - Perimeter defense, FW/IPS, and endpoint protection, Anti-virus, does not work well.
  - Attackers search vulnerable servers and PCs and build an attack infra-structure on it.
  - The attack is delivered not from “outside” but from “inside.”

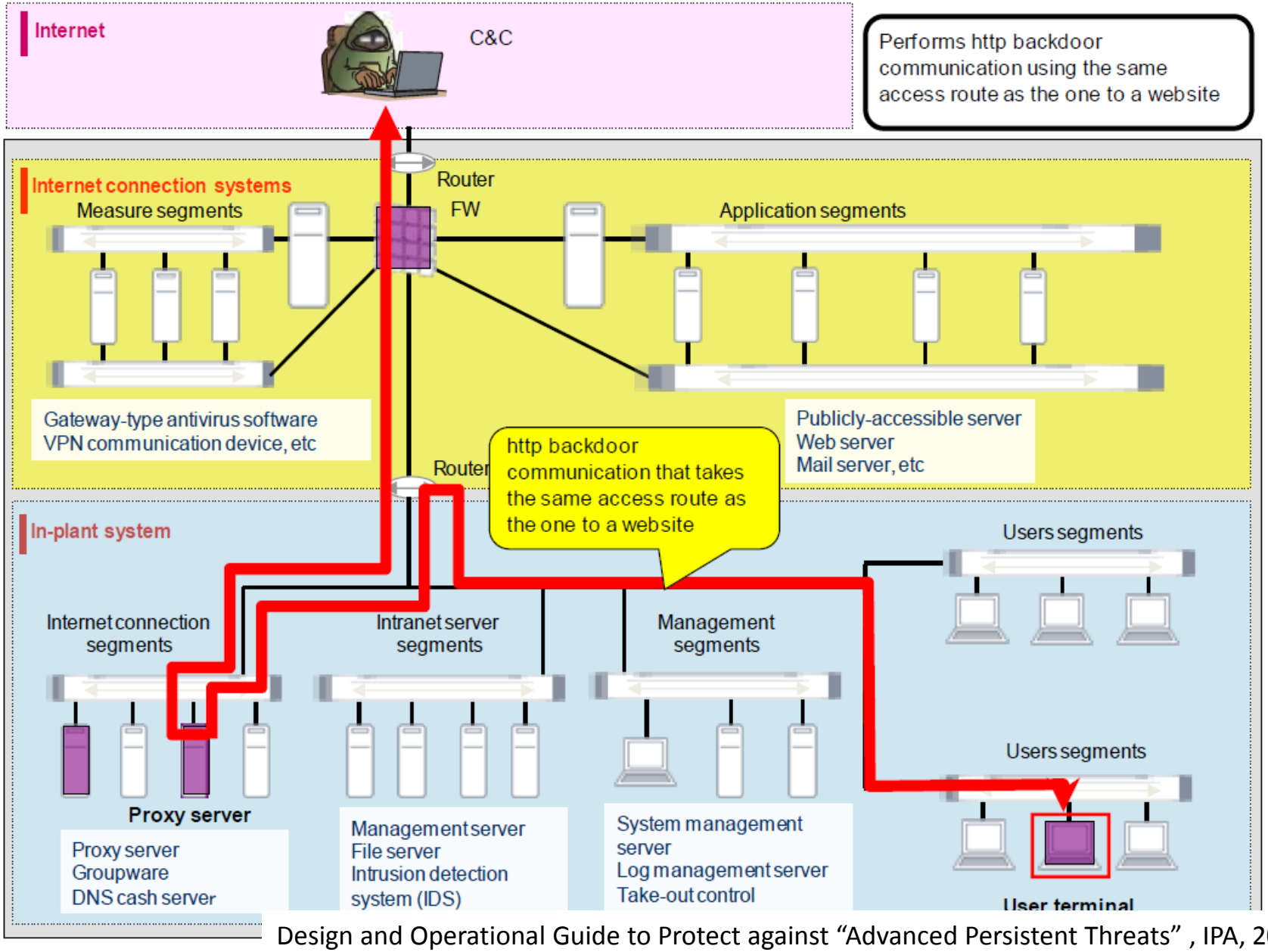
- When we design systems, we have to consider many kinds of APT behaviors.
  - System design is usually manually-produced.
  - Default settings are often used. Designers do not know all functions.
  - APT attack methods consist of many kinds of combinations of several techniques and malwares.
  - To enumerate all combination patterns is very difficult.

System designers believe FW/Anti-virus are enough to defend systems.

There is poor access controls of “attacks from inside.”

Many characteristic features of APT

- Many kinds of attack vectors and sequential infections are used.
  1. An e-mail attached with a virus are received
  2. An user PC is infected
  3. An AD server is infected
  4. The target PC is infected
  5. The confidential information is transmitted
- Targeted e-mail with a zero day attack is often used
- Covert channels for RAT connections are build
- Many kinds of malware are used
- Long term persistence attacks … and so on



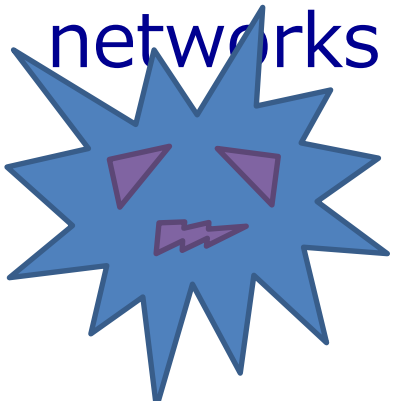
Design and Operational Guide to Protect against "Advanced Persistent Threats" , IPA, 2011

This is an example of a system which is manually produced. Attackers use many methods to communicate between inside and outside.

The manually produced system design often makes easy attacks.

Next measures against APT

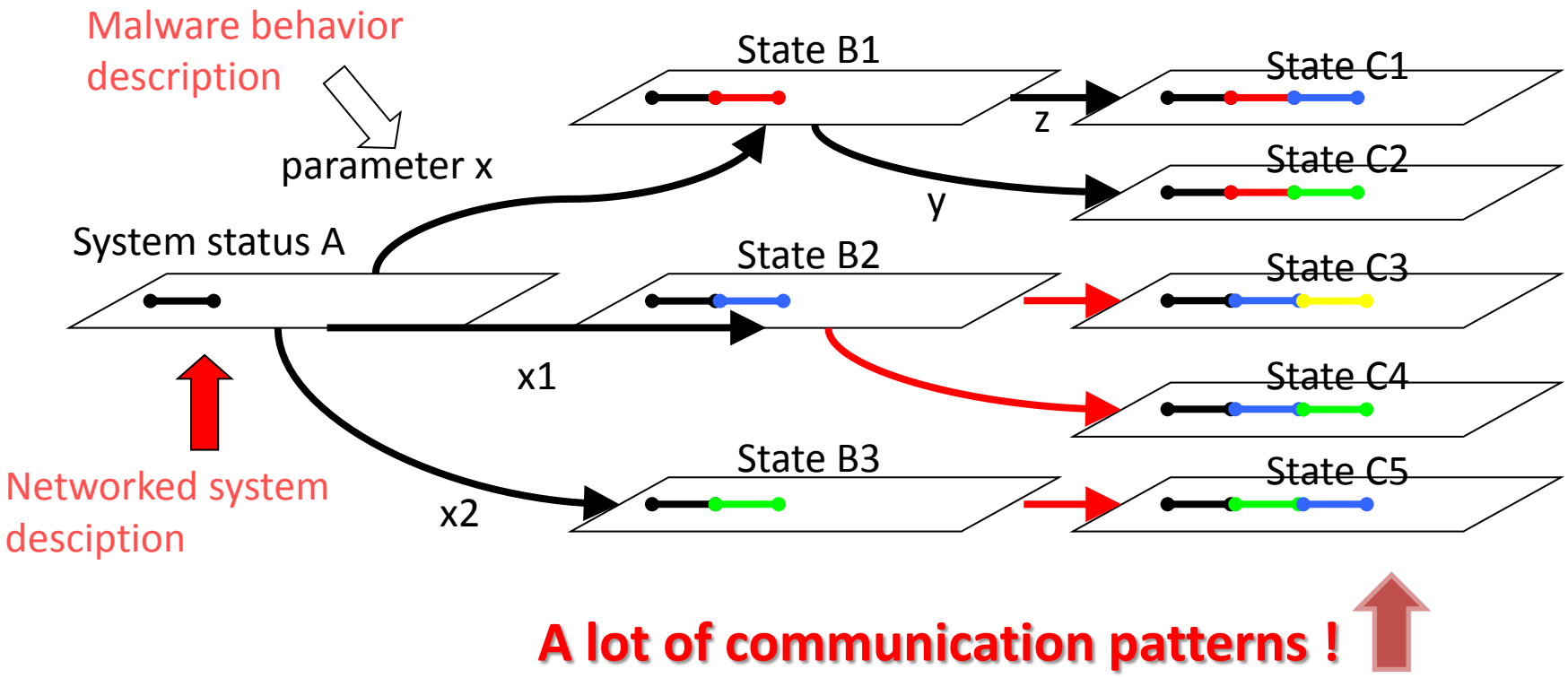
- We have to assume that there are not perfect measures to prevent viruses invade into networked systems.
- We have to prevent working of malware which has invaded.
  - We would like to design information systems in which malware can not bring confidential data to outside.
- We have to know the entire picture in the networks



Outbound defense

Basic Idea of our simulation

- If an action is success and a next action is taken by malware, communication state (protocol, destination node..) is changed.



Basic Idea of our simulation

- If an action is success and a next action is taken by malware, communication state (protocol, destination node..) is changed.
- New communication path which should not be existing essentially is produced by the malware.
- And if this change is repeated, many states of networked systems will be created.
- This simulator searches the condition of creating path from inside to outside as final condition.
- We can prevent design errors if we can use this simulator.

Networked system description

- There are many ways to describe networked systems.
  - NIST Net
  - DummyNet
  - NS2
  - OPNET
  - NetSim

- They are network emulators. The purposes and abstraction levels are different from ours.
- We try to describe access control structures of networked systems.
- We use NSQ model as an access control structure description.

Programmers like easy to do

- Programmers ≒ Hackers
- easy to do ＝ to do efficiently

An example of programming

DMA or punched card

↓

ed　　line editor

↓

vi　　screen editor

↓

emacs　coutomizable and macro functions

↓

multi window system

Integrated programming environments

We have obtained a lot of nice programming environments now we can develop software efficiently.

Easy to do is justice.

Networked Systems

The characters on the networked system (Actors)

Routers, PC terminals, Firewalls, Switches, C&C servers …

- Each hardware connected to the information system.

Malware, Web server, Web browser, Proxy server, …

- Each software is executed on an equipment in the networked system.

- The processing on each character is processing with changing status and passing messages (datagrams) with other characters.
- We would like to apply Actor model to this problem.

Actor model

- It is difficult to implement actor model on ordinary programming languages like C or Java.
  - We have to realize
    - Non-blocking message passing between actors
    - A mechanism of actors

- Erlang and Scala have actor model as one of fundamental functions.
  - We choose Scala.
    - Scala is more popular than Erlang.
    - Erlang is too fundamental.
    - Scala has high affinity with Java.

Programming Language Scala

- One of multi-paradigm programing languages which mixtures object oriented programming languages and functional programming languages.
- Scala has a lot of advanced functions. We can do programming easily.
  - one of Java platform (JVM) languages
  - availability of plentiful Java libraries
  - static typing programming language with advanced type inference
  - pattern matching
  - Mix-in, multiple inheritance
  - XML direct descriptions
  - Actor lightweight process

The Base class of character in our simulator

```
import scala.actors.Actor
import scala.collection.mutable._

abstract class MessageActor extends Actor {

  var responses: ListBuffer[(Datagram)=>Unit] = ListBuffer.empty

  OperationTool.register(this)

  override def act(): Unit = {
    loop {
      react {
        case "exit" => { Debug.println(this+": exit")
                         exit }
        case d: Datagram => {
            responses.foreach(f=>f(d))
        }
        case _ => { Debug.println(this+": unkonwn message.")
                    exit }
      }
    }
  }

  ...
  def addResponse(task: (Datagram)=>Unit): Unit = {
    responses += task
  }
}
```

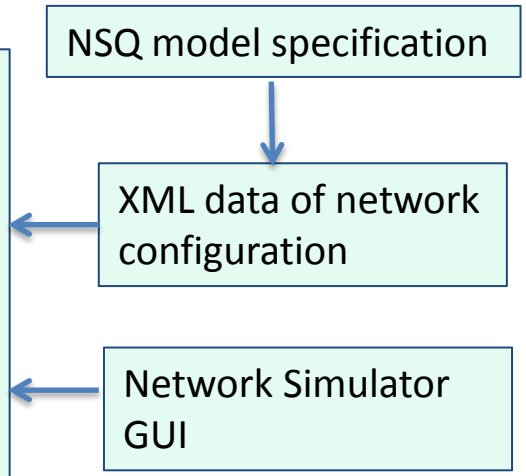
Very simple. We can extend this base class to each character.
We can append some processes which respond to datagram by using “AddResponse.”

DSL for describing networked systems

- We have to simulate the behavior of malware in practical complicated networks.
- We need to describe networks with appropriate level of abstraction, not too fine and not coarse.
- NSQ model specification can be easily converted to XML network format.
- XML network format can be easily converted to DSL description which can be executed directly as a Scala code.
- This network description also includes the information of all L5 applications (web browser, server software), L3 routing and IP address.

```
object SampleNetwork extends MalwareSimulation {
  val net1 = Network “net1” 7.7.7.0/24
  val net2 = Nonetwork “net2” 6.6.6.0/24
  val node1 = Node “My PC” 7.7.7.1/24
    net1 addNode node1
    node1 addRoute default 7.7.7.2/24 net1
  val browser = WebBrowserApp
    browser setProxy 7.7.7.3/24
    node1 setApplication browser
  …
}
```

Network description（NSQ XML -> DSL)

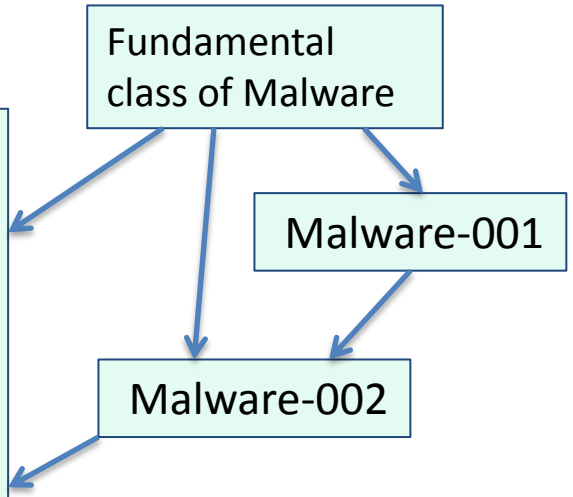


DSL for describing APT malware

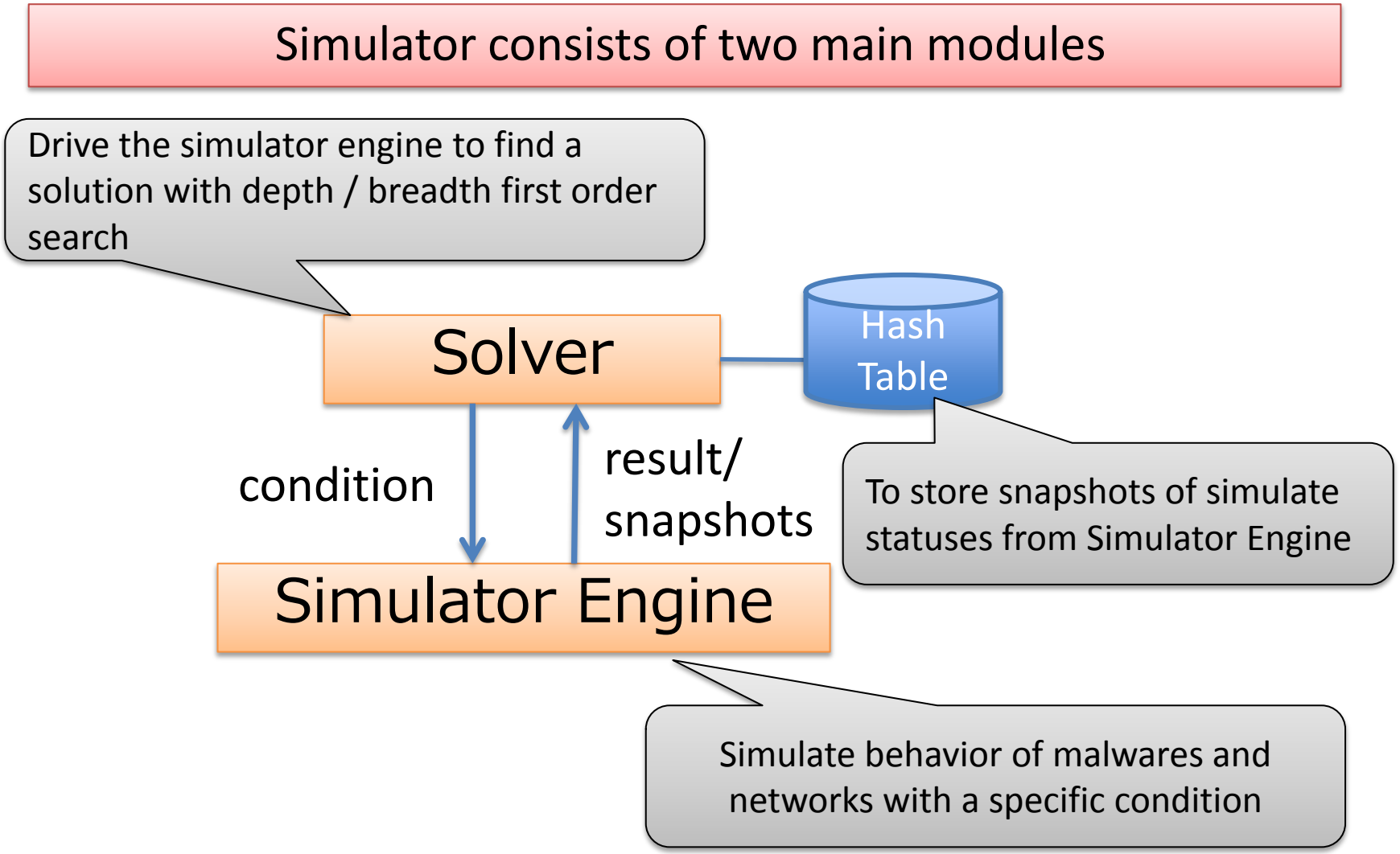
- Almost all malwares seemed to be coded by general purpose programming languages.
- Behavior of malwares must be described easily, because they are very complex and large quantities.
- The code which represents the behavior of malwares has to be executed efficiently.
- We design DSL (Domain Specific Language)
  - We can use the multi-inheritance function to describe new malware.
  - Scala supports all modern functions of programming languages.
  - The base class supplies the fundamental functions including discovery targets, infection, update itself and so on to describe new malwares.
  - We can accumulate the definitions of malware behavior as inheritance relationships.

```
object NewMalware extends Malware {
  def init {/* Malware initialize routine */}
  def code {
    /* Malware main routine */
    targetNodes foreach { node => infect(node) }
   …
  }
 …
}
```

Malware description （Malware-003)



Implementation of simulator prototype



Implementation of simulator engine

Simulator Engine is implemented with actor model

Node	Router, Switching hub, PC terminal, Server
Application	Firewall, various malwares, application software, server software
Datagram	IP Datagram emulated message (This message is exchanging between actors.)
Utility	Routing table, Internet Address and so on.

Actors exchange datagrams according to defined routing tables and network topology.

A simulation result

malware(Malware, node=node(PC1, address=[192.168.0.3/24])): Malware takes action.

Malware[info/action/exit]:node(PC1, address=[192.168.0.3/24]): NetworkNode has to deliver a datagram Datagram([192.168.0.3/24],10002,[192.168.0.2/24],8080,$$ Inportand Info. $$,[192.168.0.3/24],[192.168.2.5/24])

malware(Malware, node=node(PC1, address=[192.168.0.3/24])): unknown command

network(private): Network received a data, Datagram([192.168.0.3/24],10002,[192.168.0.2/24],8080,$$ Inportand Info. $$,[192.168.0.1/24],[192.168.2.5/24])

node(Firewall, address=[192.168.0.1/24]): NetworkNode has to deliver a datagram Datagram([192.168.0.3/24],10002,[192.168.0.2/24],8080,$$ Inportand Info. $$,[192.168.0.1/24],[192.168.2.5/24])

<<snip>>

network(global): Network received a data, Datagram([192.168.0.2/24],10002,[192.168.2.5/24],80,$$ Inportand Info. $$,[192.168.2.5/24],null)

node(Command and Control Server, address=[192.168.2.5/24]): NetworkNode has received a datagram Datagram([192.168.0.2/24],10002,[192.168.2.5/24],80,**$$ Inportand Info. $$**,[192.168.2.5/24],null)

application(CCServer, node=node(Command and Control Server, address=[192.168.2.5/24])): Received a http request.

node(Command and Control Server, address=[192.168.2.5/24]): NetworkNode has to deliver a datagram Datagram([192.168.2.5/24],80,[192.168.0.2/24],10002,http response,null,<function1>)

network(global): Network received a data, Datagram([192.168.2.5/24],80,[192.168.0.2/24],10002,http response,[192.168.2.1/24],<function1>)

node(Router, address=[192.168.2.1/24]): NetworkNode has to deliver a datagram Datagram([192.168.2.5/24],80,[192.168.0.2/24],10002,http response,[192.168.2.1/24],<function1>)

<<snip>>

network(private): Network received a data, Datagram([192.168.2.5/24],80,[192.168.0.3/24],10002,http response,[192.168.0.3/24],<function1>)

node(PC1, address=[192.168.0.3/24]): NetworkNode has received a datagram Datagram([192.168.2.5/24],80,[192.168.0.3/24],10002,http response,[192.168.0.3/24],<function1>)

malware(Malware, node=node(PC1, address=[192.168.0.3/24])): success

We have demonstrated the simulator engine at the beginning of this presentation.

Work in progress and future work

Work in progress

- Simulation on large scale networks
- More complicated attack patterns description
- Implementation of a prototype of simulation engine

Future work

- Implementation of the solver (dynamic simulation)
- Making data from real systems or existing design
- Considering how to use Mitre MAEC (Malware Attribute Enumeration and Characterization)
- Considering how to use another network model
- Visualization for system designer (GUI)
- Parallel and distributed processing of the simulation

Concluding remarks

- Network system design is important to measure APT.
- Nowadays, networked systems and malware behaviors are much more complex.
- Computer aided system design is needed and effective to understand behavior of malwares using by APT.
- We developed two types of data model; networked system and attack behavior.
- And we designed and implemented a prototype of simulation engine.
- The proposed method is effective to design networked systems.
- When we design networked systems correctly, the defense against APT will be much easier.

Presentation

Kato, M., Matsunami, T., Kanaoka, A., Koide, H. and Okamoto, E. : Tracing Attacks on Advanced Persistent Thread in Networked Systems, The 21st USENIX Security Symposium Poster Session, August 2012.

Kato, M., Matsunami, T., Kanaoka, A., Koide, H. and Okamoto, E.: Tracing Advanced Threats in Networked Systems, SafeConfig, October 2012.