Agent-based Forensic Investigations with an Integrated Framework

Buchanan WJ, Graves J, Saliou L, Al Sebea H, and Migas N
Disributed Systems and Mobile Agents Group,
School of Computing, Napier University, Edinburgh
e: w.buchanan@napier.ac.uk

KEYWORDS: AGENTS, MOBILE AGENTS, INTRUSION DETECTION, SNMP, INTEGRATED FRAMEWORK

I. ABSTRACT

Forensics investigations can be flawed for many reasons, such as that they can lack any real evidence of an incident. Also, it can be the case that the legal rights of an individual has been breached, or that the steps taken in the investigation cannot be verified. This paper outlines an integrated framework for both data gathering, using mobile and static agents, and also in the creation of a data gathering system which logs data in a verifiable and open way. Forensic information which is gathered over a network is often more verifiable over host-based data gathering. The framework for logging data for future investigations uses a formal approach where a forensics policy is defined, which is then compiled into an implementation which can run on agent systems, such as with SNMP agents, and IDS (Intrusion Detection System) agents. The paper also proposes a system which uses mobile and static agents to formalize the investigation process. This should produce investigations which can be verified, and which are programmed the expertise of an investigator, and also contain legal and moral programming to constrain the limits of a forensic investigation.

II. INTRODUCTION

The concept of a software agent which could investigate criminal activities is one which has intrigued modern society. It is unlikely that software will ever replace human abilities for discovering new patterns of activity, but it is possible for them to act as software tools which will intelligently filter information, and make reasoned judgements. They do, though, have many issues which would have to be overcome to make them possible. One of the main ones is to allow them to travel as programs across networks, and then intelligently gather information, and move away from their target and back to their source. This research proposes a network of intelligent crime agents which included to main types of mobile agent: a forensic investigation agent and a covert agent. These agents could migrate, investigate criminal data, and meet in a safe meeting place and exchange information, which would be passed onto the security services. The forensic investigation agent is programmed with the expertise of a forensic investigator, and securely gathers information on hosts around a network, and returns this to the forensic investigator in a form which allows a human to make judgements. In a similar way, the covert mobile agent is programmed with the expertise of surveillance activities, and its main objective is to gather information that could be used in a criminal activity, especially for large-scale criminal activities. There are, of course, many issues related to privacy, but these would be programmed into the agent in order that it did not breach the laws of the land.

At present, most programs require some sort of program running on the machine which provides a hook for a remote computer to make a connection to it. The problem for criminal investigators is thus to get a program to run on the criminal's computer, and then for it not to be detected, if it was investigating a possible crime, or for it to robustly deliver the information over a network, when involved in a forensic investigation. A new concept has arisen, called the mobile agent, which allows this to happen, where an autonomous program has the ability to initiate its migration across a network, carrying with it its program code, its current state of execution, and also its data. These agents can freeze themselves and migrate to another host on the network. They are, in effect, the equivalent of security agents who go to an investigation and then gather the information based on the relevant data, and leave. The intelligence of the investigation is thus built into the agent and not within any programs which could run on the investigated host. It is thus simple to recall agents from hosts, as required, and to reissue them with new objectives.

There is thus the need for mobile agents who can migrate themselves onto remote hosts and gather information, and to intelligently filter it and return with useful conclusions. If the law allowed, the mobile agent could be programmed with its objectives, and migrate itself onto a remote host, in order to determine information on criminal activities. On the other hand, a forensic agent, raises fewer issues and could also be used to migrated itself onto a remote host, and sift through the information in a methodical way, and gather key information, without either sending information over a network. The agent can then return to its originator with the key elements of the investigation. After which it can return with new objectives. In order for these agents to be secure against users stealing their information, or, in the case of covert activities, detecting their presence, they must be hardened against any form of detection, thus their code must be encrypted, and they must have sensors which detect the presence of a monitor. In the case of a mobile agent, it is possible for the agent to actually kill its own presence, as required.

In order to optimize the proposed system, the data gathering must fit within an overall framework for data gathering, thus this research proposes an integrated framework for the gathering of data in an organised way.

III. Objectives of the framework

This research focuses on the implementation of mobile agents which contain the expertise of forensic and covert investigators. The objectives are to:

- Provide a framework for the integrated collection of data which can be used for forensics investigations.
- Support the early detection, and possible resolution, of criminal activities.
- Provide a legal framework which controls the moral operation of the agents.
- Create models of a forensic agent and a covert investigation agent, with embedded investigation intelligence and filtering abilities.
- Implement mobile agents which contain their own execution environment, and would thus allow agents to migrate and execute on a remote computer without the need of a system hook.
- Integrate a secure communication facility to allow these mobile agents to intercommunicate in a secure way.
- Provide a robust mechanism to secure any code and data which the agents may carry.

IV. Mobile Agents

The mobile agent paradigm is a relatively new technology that has its origins in intelligent agents, and is proposed as an alternative approach to client-server communications model. A mobile agent is a software entity that inherits some of the features of an intelligent agent and requires an agent environment to execute. A mobile agent can suspend its execution on a host computer, and then transfer its code, data state, and possibly its execution state (strong migration) to another host on the network that must provide an agent environment, and resume execution on the new host. The aim of an agent environment [9] is to provide the appropriate functionality to mobile agents to execute, communicate, migrate, and use system resources in a secure way. In general, a mobile agent comprises of an agent model, a life-cycle model, a computation model, a security model, a communication model, and finally a navigation model [1]. Mobile agent applications include information retrieval [2], e-commerce [3], network management [4], intrusion detection [5], and collaborative applications [6], and wireless computing. Although each application can be run with the existing technologies [10], the use of mobile agents can contribute to build these distributed applications in a simpler and more effective [7]. Mobile, or wireless, computing is the most frequently proposed application area of mobile agent technology [8]. This is because mobile agents have two important features that make them particularly useful in such environments: task continuation and minimal connection.

V. Mobile Agent Development

Mobile agents run on specialized platforms such as Grasshopper. These allow agents to be invoked and allow for a preservation of state and context, including the migration of a mobile agent from one

machine to another. Grasshopper takes care of inter-host communication that involves agent migration from between hosts.

In a typical situation, the execution of the Grasshopper platform would result in the declaration of an *agency* and a GUI would be displayed allowing the administrator to access the agency's specific tasks. Agencies must be named to differentiate between them. Agents can move around hosts by knowing the names of the different available agencies.

To allow agent identification, agents must be given distinct names, preferably human names. In addition, a unique agent identifier is automatically generated by the running agency during the creation of an agent. From a general perspective, as previously stated, a Grasshopper agent is implemented by means of a Java class or a set of classes. Each agent has one agent class which characterizes the agent and which must be derived from one of the predefined *super-classes*. Several super-classes exist, including mobile-agent and stationary-agent. Due to the nature of our project, with the involvement of mobile agents, the mobile-agent class shall be adopted. Through its superclass, each Grasshopper agent has access to the following methods [11]:

- **action**(): This method is automatically invoked by the agency if a user performs a double-click on the corresponding agent entry in the agency GUI.
- **getInfo**(): This method returns a set of information that is associated with the agent. Among others, this set of information comprises the agent's identifier, type, and name.
- **getName**(): This method returns the name of the agent. In contrast to the unique agent identifier which is automatically generated by an agency during the creation of an agent, the name can be specified by the agent programmer during the implementation phase or by the user when creating the agent, provided that this is supported by the agent implementation
- **init**(): This method is automatically called by the hosting agency when an agent is created. It offers the possibility to provide creation arguments to the agent.
- **live**(): This is the core method of each Grasshopper agent, since its implementation realises the agent's active, autonomous behaviour.
- **log**(): Allows an agent to print textual messages onto the text console of the local agency.
- **move**(): With this method, an agent is able to migrate to another agency

The basic framework of the mobile agent code which migrates to a single host (in this case to 10.0.0.1), is:

```
package examples.simple;

import de.ikv.grasshopper.agency.*;
import de.ikv.grasshopper.type.*;
import de.ikv.grasshopper.communication.*;
import java.util.*;
import java.io.*;

public class ForensicAgent extends de.ikv.grasshopper.agent.MobileAgent {

   String[] files;
   int numfiles;
   GrasshopperAddress HomePlatform, suspectAddress;
   String state;

   public void init(Object[] creationArgs) {
      // Initialize data state
      files[] = new String[1000];
      numfiles = 0;
      HomePlatform = getAgentSystem().getInfo().getLocation();
      suspectAddress = new GrasshopperAddress("10.0.0.1");
      // Pass arguments to the Mobile Agent
      if (creationArgs.length < 2) {
         System.out.println("Creation arguments needed: <String> <String>");
         System.out.println("Exiting.");
```

```
            throw new RuntimeException();
        }
        else {
            setProperty(creationArgs[0].toString(),
                    creationArgs[1].toString());
        }
    }

    public String getName() {
        return "ForensicAgent";
    }
    public void live() {
        if (getProperty("state").equals("INVESTIGATING")) {
            setProperty("state", "RETURN");
            move(suspectAddress);
        }
        else if (getProperty("state").equals("RETURN")) {
            try {
                Runtime rt = Runtime.getRuntime();
                Process proc = rt.exec("dir *.jpeg /s");
                InputStream stderr = proc.getErrorStream();
                InputStreamReader isr = new InputStreamReader(stderr);
                BufferedReader br = new BufferedReader(isr);
                String line = null;
                System.out.println("<ERROR>");
                while ( (line = br.readLine()) != null) {
                    files [i] = line;
                    numfiles++;
                    if (numfiles > 998) {
                        break;
                    }
                }
                int exitVal = proc.waitFor();
            } catch (Throwable t) { t.printStackTrace(); }
            move(HomePlatform);
        }
        else
            remove();
    }
}
```

The migration path of the mobile agent is initially set within the init() method, and the main code to execute the forensic investigation is contained within the live() method. In the above example the mobile agent has retrieved a list of JPEG files, but in most cases the mobile agent would only communicate with a local static agent. This is achieved by opening a local TCP port on 127.0.0.1, such as to communication using port 2005:

```
tcpSocket = new Socket("127.0.0.1", 2005);
os = new DataOutputStream(tcpSocket.getOutputStream());
is = new DataInputStream(tcpSocket.getInputStream());
```

Thus, the local static agent simply creates a server socket, and the mobile agent connects to it using a local connection. The communicate is then achieved locally, and not over the network, as with a standard client-server program.

VI. MOBILE AGENT FRAMEWORK

Figure 1 shows an outline of the usage of the mobile agent in a forensics system. Within this, the investigator has no direct interface to the host-under-investigation (HUI). This thus keeps the integrity of the system, where a static agent is immediately installed on the HUI, and guards against any changes to the information stored on the HUI. The investigator then gives the mobile agent require-

ment, such as the names of the JPEG images on the HUI, and the mobile agent migrates itself to the HUI, and authenticates itself to the static agent, and vice-versa. The mobile and static agents can then intercommunicate with each other and pass information about the required information. Both the mobile and the static agent are programmed with a framework which supports a legal and moral framework, thus any part of the investigation which breach the limits of the current investigation will be stopped.

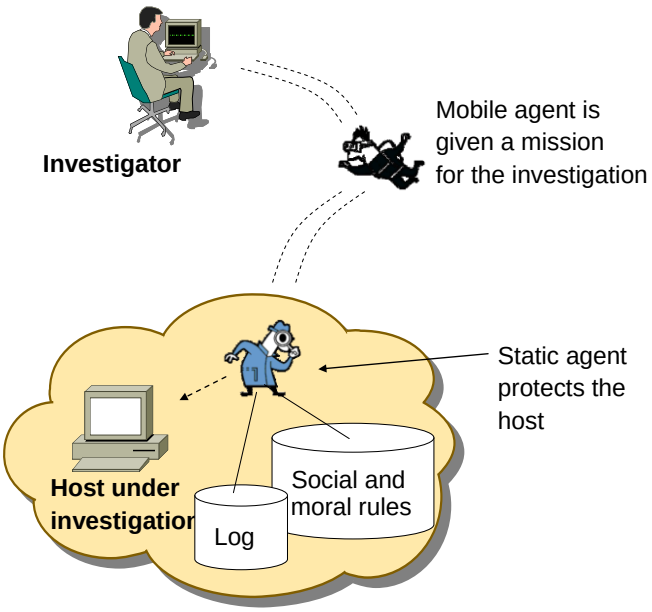

Figure 1: Forensics system using mobile and static agents

VII. DATA GATHERING FRAMEWORK DESIGN

Some forensic investigations can be flawed when there is not enough care taken in the logging information that could be used in the investigation. Most involve data generation after an event has occurred, thus in modern systems the forensic logging should be carefully considered as part of the overall security policy and implementation of an organisation. Figure 2 show a standard model for forensic computing investigations, which care is taken before the incident to prepare the policies and procedures for the detection of an event. In modern day systems, organisations must be open about their forensic policies, and the data that will be logged, as a failure to do this may result in the organisation being held liable for not detecting events, or in not informing their staff on the data that is logged.

Typically forensic investigations and system security are seen as different entities, where they can be integrated into the same policy, where the security policy defines the basic services that can be operated on the system, and how threats can be dealt with. These threats are typically detected using alerts and alarms, which are based on network monitoring, and in the investigation of log files. This is where network security joins with forensic computing, as the monitoring, logs, alerts, and alarms are basic building blocks for any future forensics investigation. Security thus focuses on the active protection of systems, users and data, while forensic computing focuses on the collection, preservation, analysis and reporting (CPAR) of events. The two come together in requiring reliable logging of data, as illustrated in Figure 3. It thus makes sense of integrate both the definition and implementation of the security policy with the definition and implementation of the forensic computing policy. Saliou [12] has developed a security framework which has a formal security policy as its input and this is used to create an implementation which maps onto the aims and objectives of the organisation, but also fits in with the legal and moral responsibilities of the organisation. This type of approach should also be applied to the generation of data for future forensic investigations, but creating a forensics policy, which integrates with the security policy, and is then used to generate the implementation of logging rules, which create formal data logs. Figure 4 shows an example of the steps taken from generating the forensic policy into implementation, and then using data agents to gather the required information into logs. The main elements:

- **Forensics Compiler**. This coverts the forensic policy, which defines the rules that define the logging of data on a network and on hosts, and coverts into a formal forensic language, which can be modelled to determine if there are any conflicts with the implementation.
- **Log implementer**. This converts the output from the forensics compiler into an implementation, such as for rules for IDSs, or to enable router logs.
- **Forensic verification traffic**. This is the network traffic which is used to verify that the system is working correctly, and includes a capture of normal traffic along with the required additional traffic which verifies that the system is working as required.
- **Live deployment**. This involves downloading the rules from the Log implementer to the devices. This includes agents such IDS agents, SNMP agents, and router/switch log agents. The agents will be responsible for filtering network traffic and logging it the required log file.
- **Data gathering**. This includes gathering the data which are stored by the agents, into a useable format.

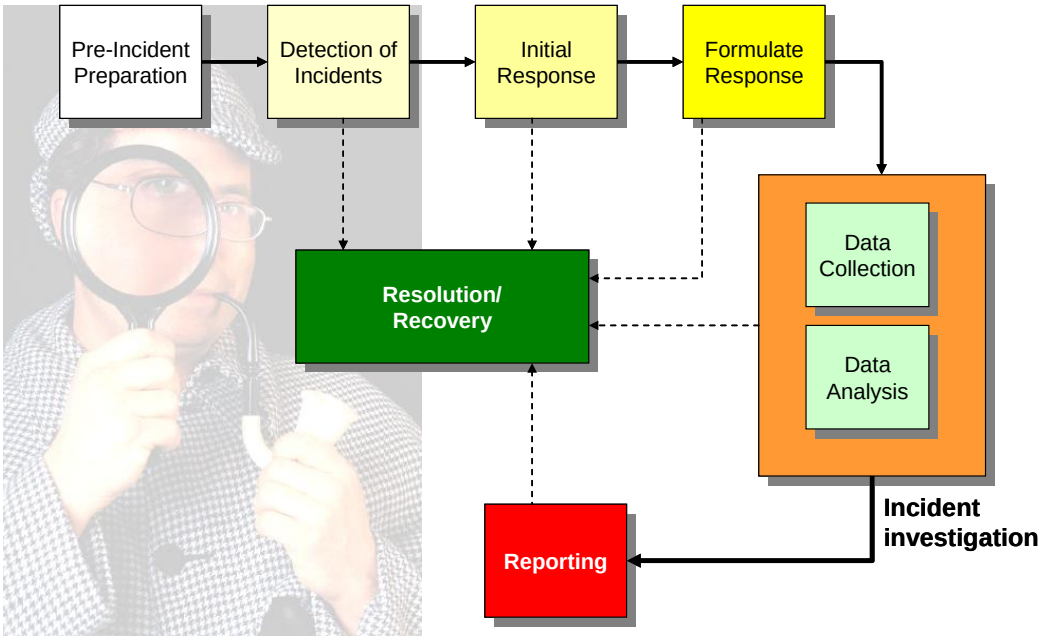

Figure 2: Forensic investigation stages

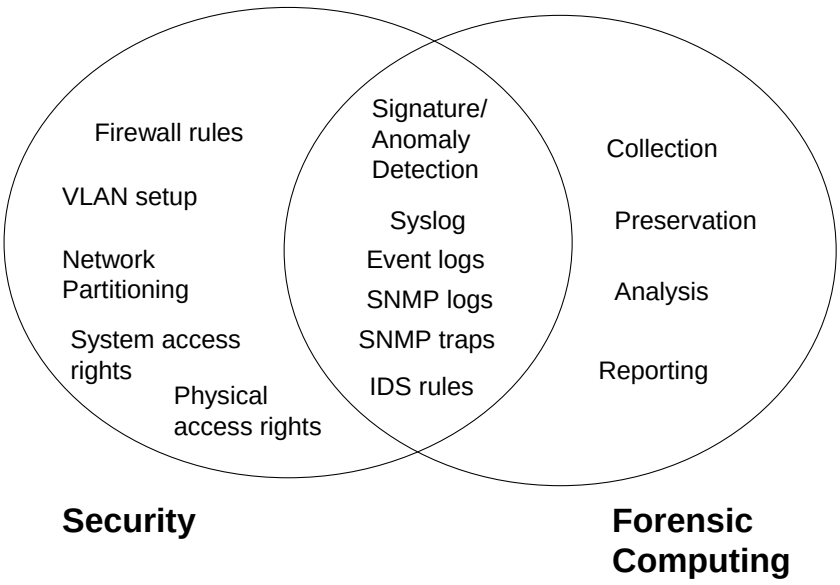

Figure 3: Intersection of forensic computing and security

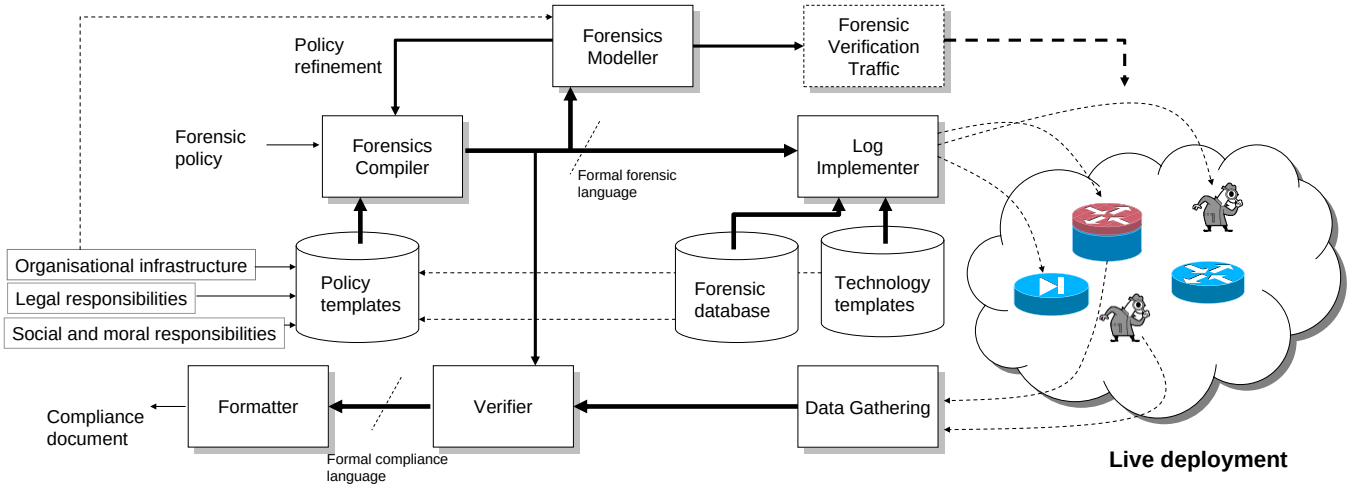

Figure 4: Formal definition and implementation of the forensic computing policy

I. CONCLUSIONS

Some forensic computing investigations can be flawed in that they can lack the required information. The framework proposed in this paper aids the collection of data for forensics investigation. This framework integrates with the general security definition for an organisation, as security policies also require data gathering. The cross-over between the security definition and the data logging enables the data gathering policy to be defined, so that everyone in an organisation understands the range and type of data that is logged. Along with this, alerts can be setup to provide an early detection of criminal activities before they can develop into more serious activities. These network-based logging of data for investigations are typically more reliable in its source as host-based, which can often be tampered with, or which may not leave a trace.

For host-based investigations, the combination of mobile and static agents approach proposed in the paper has advantages that it allows the verifiable logging of the investigations, and also provides the investigations to be constrained within the limits defined by moral and legal restrictions. Along with this, the mobile agent can be programmed with the intelligence of a criminal investigator, where it can be given high-level investigation command, which it then communicates with a static agent on the host under investigation. The mobile and static agents cooperate to make sure that the investigation is conducted within a legal framework.

II. REFERENCES

[1] Harrison, C. G. et al, 1995. Mobile Agents: Are they a good idea. *Technical Report*. IBM T.J. Watson Research Centre. New York, USA.

[2] Cardi, G. et. al., 2000. Agents for information retrieval: Issues of mobility and coordination. *Journal of mobility and coordination*. Vol. 46, No. 15. pp. 1419-1433.

[3] Lee, T. O. et. al., 2001. An agent-based micropayment system for E-commerce. *E-commerce agents, Marketplace solutions, security issues, and supply and demand*. Berlin, Germany. pp.247-63.

[4] Marques, P. et. al., 2001. Providing applications with mobile agent technology. *IEEE Open Architectures and Network Programming Proceedings*. Piscataway, USA. pp.129-36.

[5] Spafford, E. H. and Zamboni, D., 2000. Intrusion detection using autonomous agents. Computer Networks. Vol. 34, No. 4. pp.547-70.

[6] Wong, D. et. al., 1997. Concordia: An Infrastructure for Collaboration Mobile Agents. *In Proceedings of the first International Workshop on Mobile Agents (MA97)*. Berlin, Germany. pp. 86-97.

[7] Puliafito, A. et. al., 2000. MAP: Design and implementation of a mobile agents' platform. *Journal of Systems Architecture*. Vol. 46, No. 2. pp.145-62.

[8] Kotz, D. et. al., 1997. Agent TCL: Targeting the needs of Mobile Computers. *IEEE Internet Computing*. Vol. 1, No. 4. pp. 58-67.

[9] Silva, A. R. et. al., 2001. Towards a reference model for surveying mobile agent systems. Autonomous Agents and Multi Agent Systems. Vol. 4, No. 3. pp.187-231.
[10] Harrison, C. G. et al, 1995. Mobile Agents: Are they a good idea. *Technical Report*. IBM T.J. Watson Research Centre. New York, USA.
[11] IKV++ GmbH (1999). Grasshopper Programmer's Guide (Release 2.2). Germany.
[12] Saliou L, Buchanan WJ, Graves J, and Munoz J, Novel Framework for Automated Security Abstraction, Modeling, Implementation, and Verification, EICW 2005.