Applied Soft Computing 12 (2012) 1607–1619

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

iDetect: Content Based Monitoring of Complex Networks using Mobile Agents

Umar Manzoor*, Samia Nefti

Department of Computer Science, School of Computing, Science and Engineering, The University of Salford, Salford, Greater Manchester, United Kingdom

ARTICLE INFO

Article history:
Received 21 September 2010
Received in revised form
10 September 2011
Accepted 23 October 2011
Available online 9 November 2011

Keywords:
Complex network monitoring
Content based monitoring
Mobile agent
Distributed proxy server
Collaborative Multi-Agent System

ABSTRACT

With the evolution in computer networks over the last decade, researchers are trying to come up with efficient approaches which can help network administrator in implementing the acceptable use policy for large complex networks. In this paper we have modified An Agent Based System for Activity Monitoring on Network – ABSAMN architecture and proposed iDetect: Content Based Monitoring of Complex Network using Mobile Agents which uses the content (i.e. text, image and video) of the application for categorization purpose. iDetect is implemented in Java using Java Agent Development (JADE) framework and supports platform independence; however, the framework has been tested only on Microsoft Windows (any version) environment. We have evaluated iDetect and ABSAMN on same configuration concurrently at the university campus having four labs equipped with 60–120 number of PCs in various labs; experimental results shows that iDetect efficiently detects known/unknown illegal activities (applications/websites) running on the network as compared to ABSAMN.

1. Introduction

Computer Network contains two or more computers connected together for information sharing and communication. In the last few decades, researchers have proposed many efficient methods for building large networks and efficient protocols for communication or information sharing [3]. Computer networks are deployed in order to connect computers with each other for resource (Printer, Internet, etc.) and data sharing. Network administrator usually applies restriction on the network commonly known as Acceptable Use Policy or Fair Use Policy to (1) secure the network (i.e. by blocking illegal application) and (2) restrict the usage of network nodes (Internet usage, node configuration, etc.) [2]. Acceptable use policy contains a set of rules managed by the owner/manager of the network to restrict the ways in which the network system may be used.

With the evolution in computer networks over the last decade, researchers are trying to come up with efficient approaches which can help network administrator in implementing the acceptable use policy for complex networks. Proxy Server is usually deployed on the network to implement the acceptable use policy (i.e. monitor and restrict Internet usage over the network). Proxy Server acts as a mediator between the local network and the Internet (i.e. all the Internet traffic goes to/from the proxy server). It evaluates all the Internet requests from the network users according to its filtering criteria (i.e. URLs, keywords) which are defined/updated by the network administrator manually. If some network node sends an Internet request, the same is forward to the proxy server for validation, if the request does not passes the filtering criteria (i.e. violates the criteria set by network administrator) the request is discarded and an error message is returned to the user as shown in Fig. 1(a). If the request is validated, the same is forward to the external server through Internet and the response is forwarded to the network node as shown in Fig. 1(b).

Proxy Server currently available in the market has the following problems

- It becomes bottleneck as all the Internet traffic is diverted to/from the proxy server and the overall speed of the response tends to decrease as requests have to be processed for validation before passing them to the appropriate servers especially when large numbers of requests are forwarded to the proxy.
- It validates the request based on URL/keyword(s), therefore all websites which contains those keyword(s) will be blocked regardless it is legal or illegal site (e.g. if video keyword is defined in the filtering criteria than all website URLs which contains video as part of link or video as keyword will be blocked). Moreover, network administrator has to define the criteria and update it manually. Few proxy servers available on the Internet provide dynamic update of rules (URLs) by downloading the list from the server. However, all proxy servers available are unable to detect similar/correlated websites (i.e. If YouTube is banned in the network, proxy server will never be unable to block all video streaming websites except YouTube).

* Corresponding author.
E-mail addresses: umarmanzoor@gmail.com (U. Manzoor), s.nefti-meziani@salford.ac.uk (S. Nefti).

1568-4946/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2011.10.025

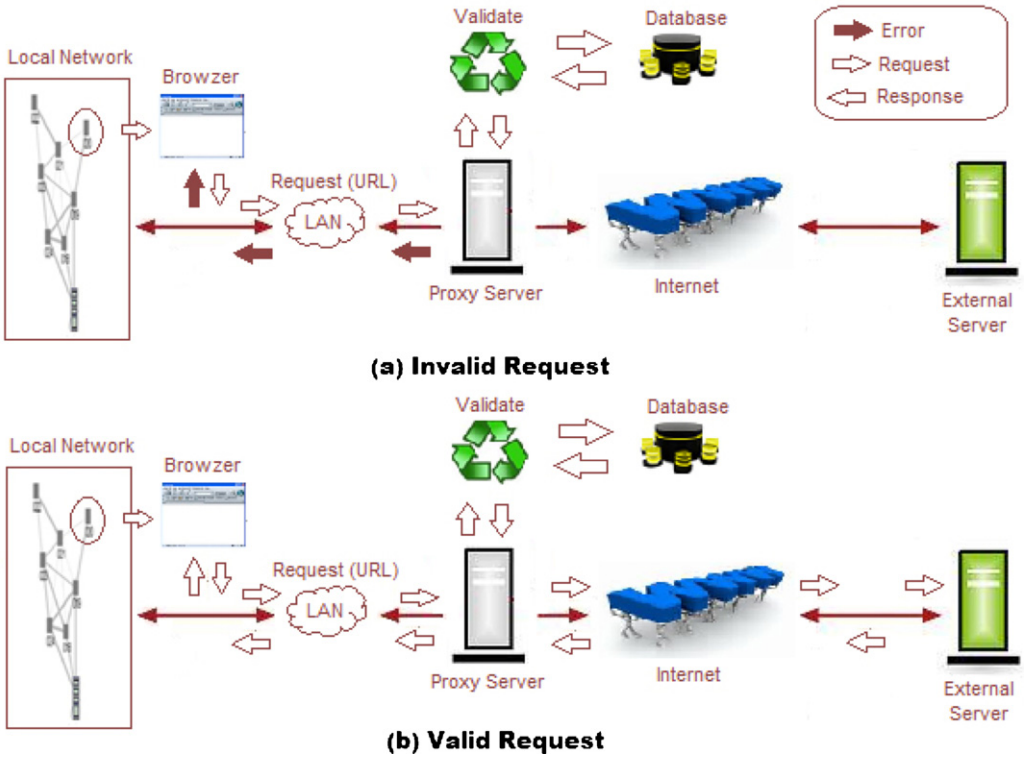

Fig. 1. (a) Invalid Internet request. (b) Valid Internet request.

- It can easily be by-passed using anonymous proxy websites [5,6] available on the Internet and with the help of these websites user can access banned/blocked websites over the Internet. Fig. 2(a) shows a normal request of YouTube website and Fig. 2(b) shows a request of YouTube using an anonymous proxy website Zend2 [5]. Anonymous proxy website encrypts the request (URL) before sending it to the proxy server, making it impossible for proxy server to detect the actual request as shown in Fig. 2(b).

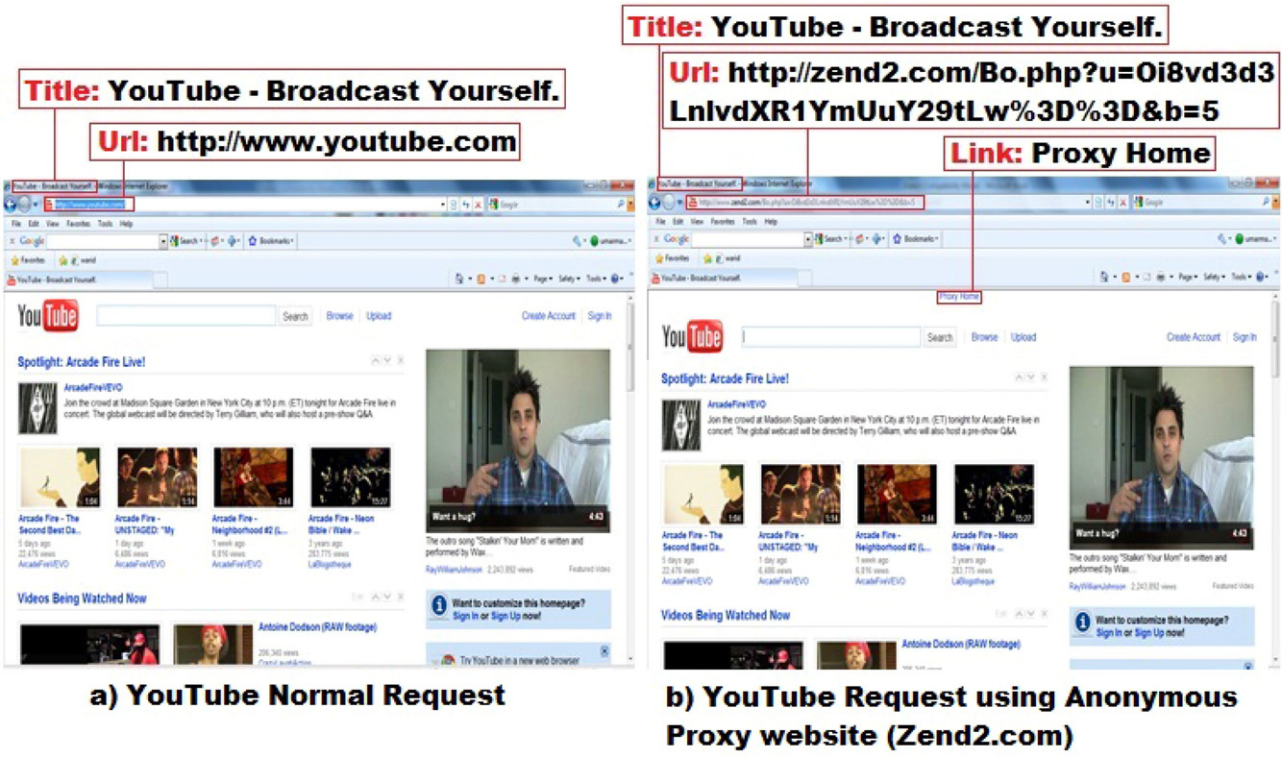

Fig. 2. (a) YouTube normal request. (b) YouTube request using anonymous proxy website (Zend2.com).

Internet explorer 6 or later version provides content adviser which can be used to control the Web sites the user can view, the administrator of the system has to provide list of websites which they like to be seen/viewed by the user. Content advisor is password protected, however the user can easily disable it by using the tips available on the Internet [12]. Content Advisor is available only in Internet Explorer; if the user installs some other browser (i.e. Firefox, Opera, etc.) filtering will not be possible. Many softwares/tools are freely available on the Internet for Internet control and monitoring such as K9 Web Protection [15], Safe Eyes [16], Cyber Patrol [17], etc., however, all of these requires client side installation which requires manual maintenance.

Monitoring large complex networks includes Internet monitoring, node configuration, illegal application monitoring (i.e. application which are not allowed or banned on the network), and monitoring network user activities. Network administrator along with his team is responsible to monitor the network and usually they use different softwares to accomplish the above mentioned tasks [18]. With the evolution in computer networks, many companies are trying to come up with automated monitoring tool(s) to make life easier for network administrator. However, currently available monitoring tools have the following problems (1) service or tools should be installed on the network node for monitoring which requires manual maintenance, (2) requires human interaction to perform the task (i.e. rules needs to be defined and updated manually), (3) unable to detect correlated softwares (i.e. if torrent software is banned, it can only detect those torrent softwares which are in its database) and (4) can easily be detected/stopped using available tools [21,22].

The need of the time is to develop an autonomous framework which is (1) autonomous (does not require human interaction in monitoring), (2) intelligent (learns and updates its knowledge base automatically), (3) light (uses less resource such as bandwidth), (4) fault tolerant (based on decentralized design) and (5) able to perform native operations such as killing application, extracting text from application/softwares and extracting frames from video Umar et al. in [4] proposed a multi-agent based framework "An Agent Based System for Activity Monitoring on Network – ABSAMN" for monitoring illegal activities over the network, suitable for complex networks. ABSAMN uses Mobile Agents to monitor the illegal activities such as video streaming, games and torrent softwares on the network nodes and has effective mechanism for blocking these activities to avoid the exploitation of network.

Multi-Agent System [27,29] is composed of several agents which collaborate and communicate their actions with each other in order to achieve common goals that are difficult to achieve by an individual agent [1,7,26]. Agent technology offers the concept of mobility (i.e. An Agent which has the capability of moving from one node to another node of the network autonomously commonly is known as Mobile Agent) [10,19,31]. Agent paradigm has been used in diverse areas such as semantic web services, Bit-Torrent protocol, spam filtering, health monitoring, manufacturing, supply chain, economics/finance, and inventory control [24,8,25,9,28,23,30,20] because of its unique characteristics.

ABSAMN manages the resources autonomously (i.e. no user interaction required) using Mobile Agents and does not require any software installation on the network node. However, it has the following problems (1) Rules are defined/updated manually and these are defined in the form of process name to action pair. (2) It uses process name defined in the Rule set XML to classify the application as malicious or non-malicious (3) It is unable to detect correlated malicious applications/websites.

```
<Rule>
    <PROCESS_NAME>GoogleTalk</PROCESS_NAME>
    <ACTION>KILLPROC</ACTION>
</Rule>
```

Sample Rule

In this paper we have modified An Agent Based System for Activity Monitoring on Network – ABSAMN architecture and proposed iDetect: Content Based Monitoring of Complex Networks using Mobile Agents which uses the content (i.e. text, image and video) of the application to predict it as malicious or non-malicious. iDetect is fully autonomous and uses WordNet [33] for text analysis and hybrid (skin and shape based) technique for malicious image detection. iDetect dynamically update its Knowledge Base (KB) by detecting new malicious unknown illegal application/website.

The remainder of this paper is organized as follows. In Section 2, data extraction from process (i.e. application/website) running in Microsoft Windows Operating System is discussed. This section is followed by the discussion of the iDetect architecture including the text/image analysis and classification method. In

Fig. 3. Internet explorer windows.

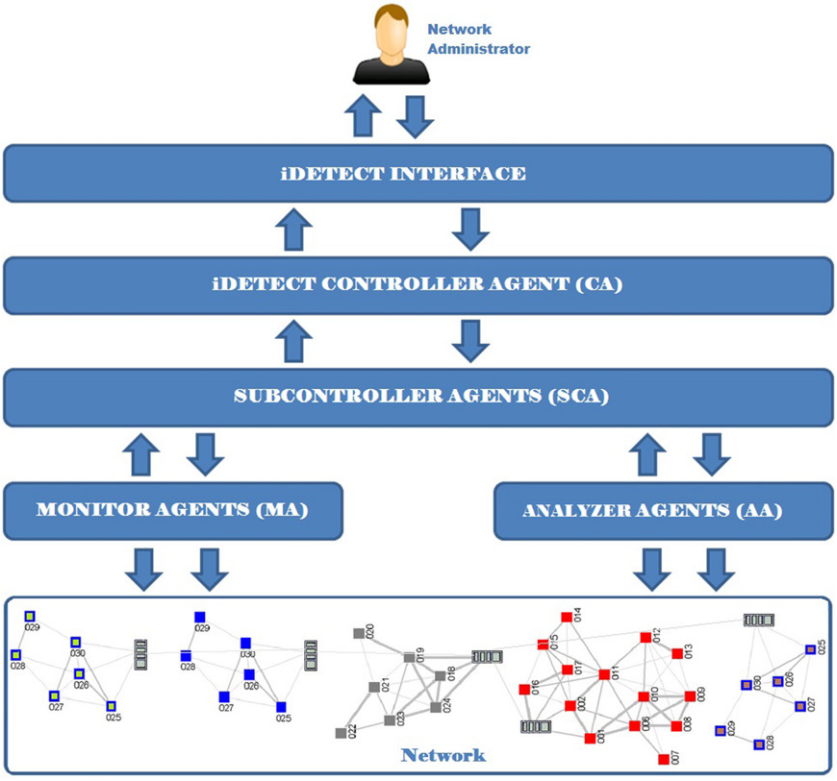

Fig. 4. Layered Architecture of iDetect: Content Based Monitoring of Complex Networks using Mobile Agents.

Section 4, we critically evaluate and compare the performance of iDetect with ABSAMN. Finally, the conclusion is drawn in Section 5.

2. Data extraction

Windows applications consist of a main window frame and may contain a number of child windows as shown in Fig. 3. Each window corresponds to a separate class; each class has associated methods to get information about that particular window. After obtaining a handle to the main window all child windows handles can be enumerated and for each child window (that is also a parent) all of its child windows can also be enumerated.

Internet explorer is shown in Fig. 3 as an example, after extracting the handle of the main window, child windows can be extracted (i.e. child window 1, child window 2) and data can be extracted from the handles during the traversal. Since child window 1 is also a parent its children can also be enumerated (e.g. child window 2). Similarly any windows application can be enumerated and information such as opened file name and title can be obtained from the windows handle.

In the example above the window title can be extracted after obtaining a handle to the 'Main Window', this gives useful information about the opened URL. Furthermore, upon enumeration of child windows a handle to 'child window 1' is obtained and from its enumeration the 'child window 2' handle is obtained which provides information about the opened URL. Similarly further data can be extracted from other handles.

3. System architecture

iDetect: Content Based Monitoring of Complex Networks using Mobile Agents is an automated system which detects malicious activities over network and uses ontology knowledge base proposed by Manzoor et al. in [11] for classification. iDetect is implemented in Java using Java Agent Development (JADE) framework and supports platform independence; however, one component (SysOpr) of the framework is platform dependent. SysOpr is responsible to perform native operations (such as extracting application data and killing applications) and each operating system has its own mechanism to perform these tasks, therefore, we developed SysOpr for Microsoft platform and tested this framework only on Microsoft Windows (any version) environment. In order to run this framework on heterogeneous environment, SysOpr component needs to be developed for different platforms present in the environment.

The proposed framework extracts the content of the application/website and uses it for classification. It may be possible that application/website is classified wrongly using the classification method, therefore, in the proposed framework we have incorporated High Level Guidance from Administrator (HLGA) mechanism whose purpose is to reduce the number of false positives. Using this mechanism the network user can lunch a complaint (i.e. valid website/application banned) which is forwarded to network administrator for analysis purpose. After manual analysis, network administrator assigns category to the website/application (i.e. legal or illegal) and the same is added to the framework banned or allowed applications list. iDetect supports multi-profile monitoring option (i.e. different rules for different parts of the network). The framework is implemented in Java using Java Agent Development framework (JADE) [35] which is based on FIPA standards and does not require any software installation on network node, the only requirement is Java Virtual Machine (JVM) which is usually installed on every machine. The architecture of the framework is based on layered approach as shown in Fig. 4 and consists of the following agents

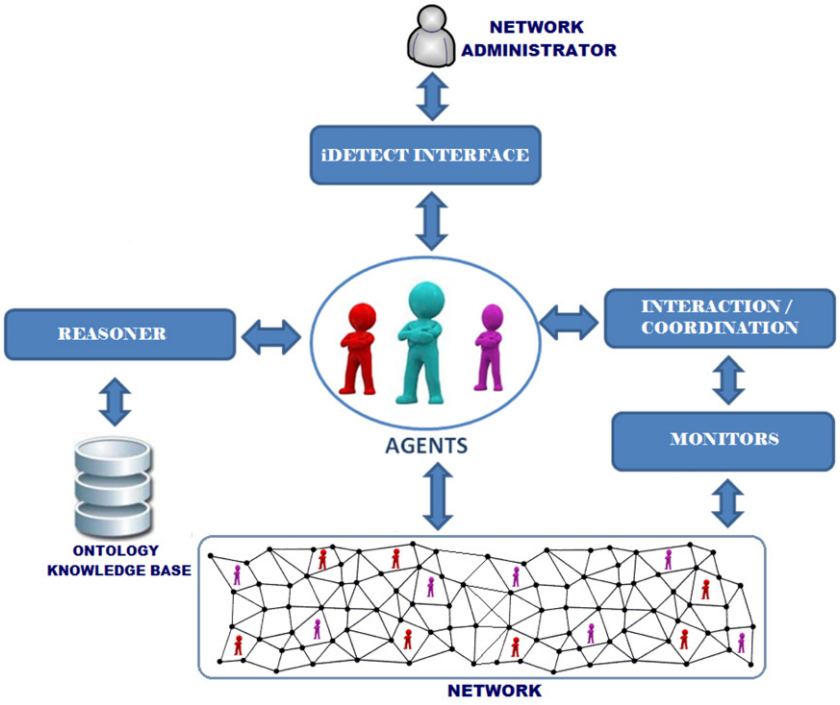

Fig. 5. Zero level diagram of iDetect: Content Based Monitoring of Large Complex Networks using Mobile Agents.

- iDetect Controller Agent (CA)
- iDetect SubController Agent (SCA)
- Monitor Agent (MA)
- Analyzer Agent (AA)

To load the framework for the very first time over the network, a batch file is provided to the network administrator; all he has to do is run the batch file. The batch file contains all the necessary commands to create the JADE containers on the network nodes and connect them to the JADE server. However, before running the batch file, network administrator has to perform the following two steps (Fig. 5).

(1) Create a shared folder (which is accessible by every node of the network) and copy the JADE libs in this folder.
(2) Add the shared folder path in the batch file.

Once the batch file is run, JADE containers are created on the network nodes and each one is connected to the JADE server. After this step, JADE loads the default configuration of the system and creates iDetect Controller Agent who is responsible for the management of the whole system.

3.1. *iDetect Controller Agent (CA)*

iDetect Controller Agent (CA) is the brain of the system as it manages and initializes the system autonomously. In initialization CA performs the following tasks

- It loads the network configuration XML which contains the information about the sub-networks of the network (i.e. Node Ids, Node IP addresses, JADE container names), activity XML which contains activities that are not allowed (illegal) on the network or sub-network(s), banned applications list and allowed applications list. Sample Activity XML file is shown as follows.

```
Sample Activity XML File
<Networks>
  <Network1>
      <Nodes_ID>
         <From>PC_33</From>
         <To>PC_128</To>
       </Nodes_ID>
       <IP>
         <From>172.16.11.33</From>
         <To>172.16.11.128</To>
       </IP>
       <JADE_Container>
         <From>iDetect_172.16.11.33</From>
         <To>iDetect_172.16.11.128</To>
       </JADE_Container>
      <ACTIVITIES>
       <Systemconfiguration>
          <Actions>
            <Action>Rollback</Action>
            <Action>Warning</Action>
          </Actions>
       </Systemconfiguration>
       <Chat>
          <Actions>
            <Action>Kill</Action>
            <Action>Warning</Action>
          </Actions>
       </Chat>
       <Internet>
          <Adult_Web>
            <Actions>
              <Action>Blocked</Action>
              <Action>LockAccount</Action>
            </Actions>
          </Adult_Web>
          <Video>
            <Actions>
              <Action>Error</Action>
              <Action>LockAccount</Action>
            </Actions>
          </Video>
       </Internet>
       ...
       ...
      </ACTIVITIES>
   </Network1>
   <Network2>
     ...
     ...
   </Network2>
   ...
   ...
</Networks>
```

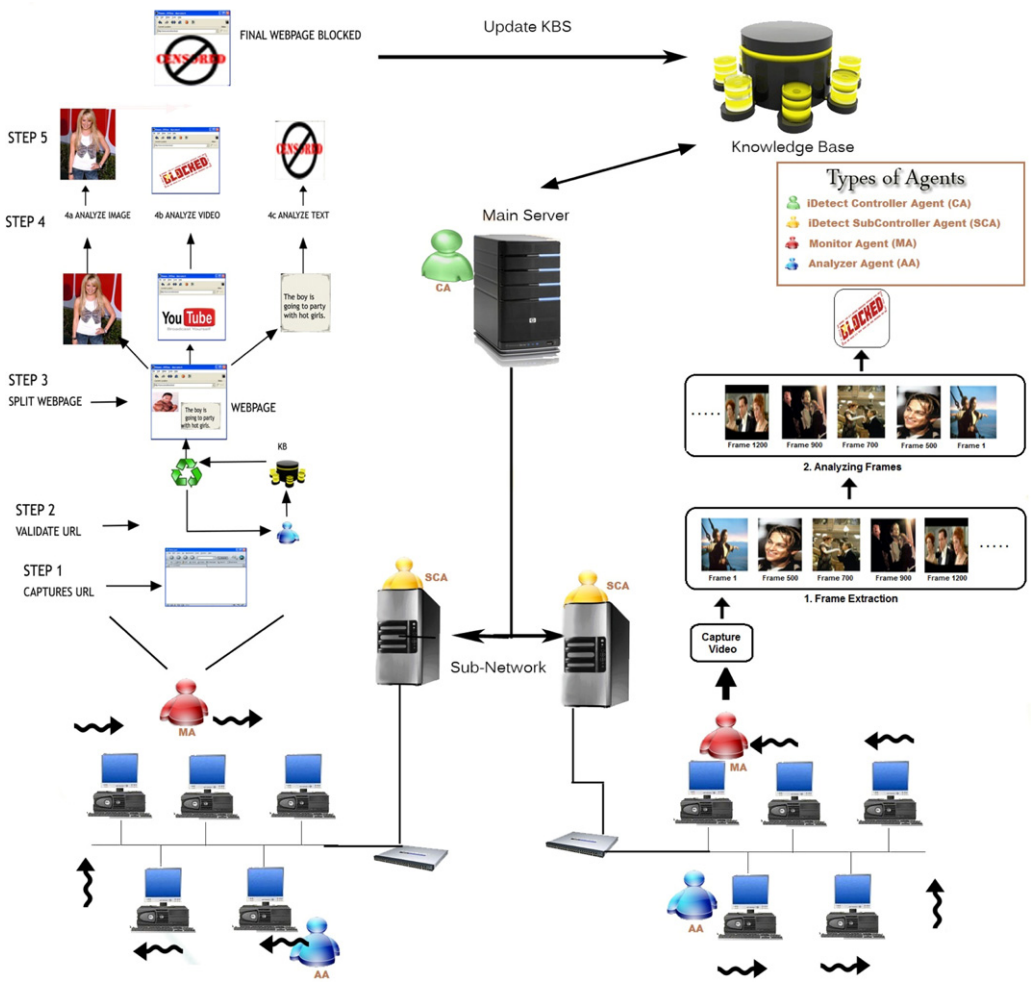

Fig. 6. System Architecture of iDetect: Content Based Monitoring of Complex Networks using Mobile Agents.

- CA makes different Rule Profiles for the sub-networks based on the activity XML file.
- CA stores the information of banned and allowed application in the hash table (BA-List).
- CA loads ontology from Ontology based Knowledge Base.
- CA creates *N* number of iDetect Sub-Controller Agents (SCA) where *N* depends on number of sub-networks (i.e. 1 to 1 ratio) and passes it destination server, rule profile, ontology, banned application list, allowed application list and configuration of the sub-network as arguments. After initialization these Agents move to the assigned destination to monitor the activities of that network.
- Once SCA reaches its destination, CA transfers the compiled code of Monitor Agent and SysOpr (executable file written in C++ to perform native operation) to CA (Fig. 6).

After initialization of the system, CA waits for the updates from SCA and in case of any violation or newly found allowed application reported, it updates banned or allowed list accordingly and send this update to other SCAs. It also keeps track of the SCAs by constantly monitoring the activities of these agents.

3.2. Sub-Controller Agent (SCA)

After reaching the destination server, Sub-Controller Agents (SCA) performs set of tasks

(1) SCA loads BA-List (banned and allowed application hash table), ontology, network and rule profile configuration passed by the CA.
(2) SCA creates N number of Monitor Agents (MA) where N depends on Agent to Network nodes (currently 1–4) ratio which is configurable. Agent to network nodes ratio denote the number of nodes to be monitored by a single agent. SCA initializes the MAs with ontology, rule profile, banned application list, allowed application list, and list of network nodes to monitor. After initialization these MAs moves to the destination nodes one by one, monitor the running applications and report any malicious activity to SCA. On getting the update from MA, SCA updates its knowledge base with the violation details and also send these details to CA.
(3) SCA creates N number of Analyzer Agents (AAs) where N depends on Agent to Network nodes ratio (currently 1 to 8) and initializes these with history (i.e. IP address details, node ID, previously installed softwares, etc.) and list of network nodes to monitor. After initialization these AAs moves to the destination nodes one by one, verify system configuration and report any change(s) to SCA. On getting the update from AA, SCA updates its knowledge base and also forward these details to CA.

3.3. Monitor Agent (MA)

Monitor Agent (MA) can be thought of brain of the system as it classify applications running on the network node as legal or illegal.

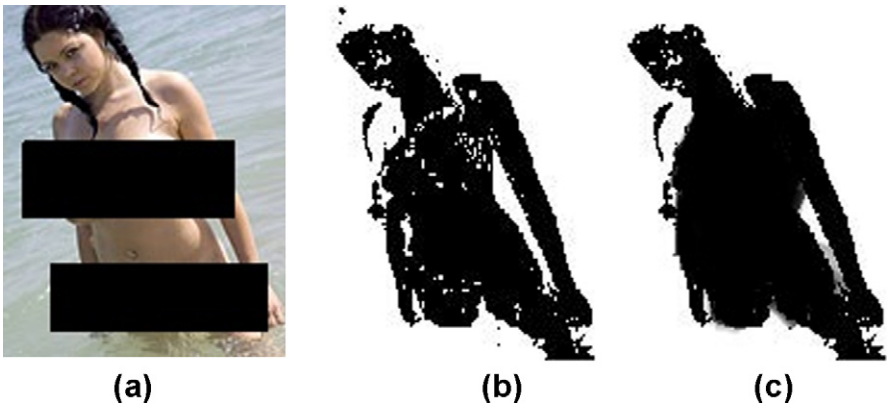

Fig. 7. (a) Original image but adult parts not shown. (b) Image after skin detection. (c) Refined image after applying Canny Algorithm.

After dispatching from Sub-Controller server, it moves to the nodes in the itinerary one by one, extracts all user processes running on that node using SysOpr and stores it in Process vector set model P.

$$P = \{P_1, P_2, P_3, P_4, \ldots, P_n\} \quad (1)$$

For each process in P, it extracts the information which includes name, url, size, path, company and stores the information with the process in the vector model P. After extracting the information, pre-processing is performed on P. In pre-processing, MA matches the information of each process in P with the BA-List hash table, if any illegal process is found, it performs the action defined against this activity, updates its knowledge base and also reports the violation to SCA with node number, process id and action taken. After completing the scan, it excludes all the processes from P which are allowed or which have been identified and reported to SCA.

$$P_f = \{P_1, P_2, P_3, P_4, \ldots, P_m | m \ll n\} \quad (2)$$

After performing pre-processing, for each remaining process in P_f, MA extract the core information which includes content (text, image, video) or built-in strings (content in case of website and built-in strings in case of software), process title, ports (TCP, UDP, Remote-Address), keystrokes frequency, mouse clicks frequency, DLLs, process screen resolution, network connection, packet rate, files opened, file extensions etc. and stores the information with the process in filtered vector model P_f. However, each process might not have all the attributes mentioned above and may contain useless information which needs to be removed. MA classification algorithm consists of three steps: (I) Text analysis, (II) Image/Video analysis and (III) classification.

3.3.1. Text analysis

iDetect text analysis algorithm uses WordNet [33] as knowledge base for word sense disambiguation and is similar to the approach proposed by Manzoor et al. in [11]. In step 1, all stoplist/stemmer/non-alphabetical words are eliminated, homogenous words are substituted by a single word and multiple entries for each word are eliminated from the text or built-in attribute of the process P_i. In step 2, each keyword is assigned the most related sense by comparing all possible senses by taking two keywords at a time and concept set is generated for theses related senses which contain either the Lowest Super Ordinate or the individual keywords concept as proposed in [32]. Malicious keywords list, its synonyms/hypernym synsets (extracted from WordNet) are compared with the concept set and matches are stored in a Resultant set (R) with uniform score (i.e. 1).

$$T_A = 2 \times \text{Score(Title)} + \sum_i \text{Score}(R_i) \times \text{Frequence}(TRi) \quad (3)$$

It compares the process title attribute with the original keyword list, synonyms/hpernyms sets and assign it score based on the comparison (i.e. in case of match one will be assigned else zero). The total score of text analysis is calculated by multiplying each individual score of resultant set with the frequency of occurrence in the text then summing up all of these scores plus adding twice the score of title attribute. If the total score of text analysis is more then threshold value ∂ malicious tag is assigned to the text. ∂ value is initially set by calculating the text score of malicious known websites and updated when new application text is assigned as malicious.

3.3.2. Image/Video analysis

iDetect Image/Video analysis algorithm uses skin and shaped based technique to detect whether the image or video frame contains the malicious (i.e. adult) content or not and the approach is similar to the approach proposed by Zheng et al. in [34]. In step 1, skin pixels present in an image (if any) are detected by analyzing the image pixel by pixel. The greater the number of skin pixels present in an image, the greater is the probability that the image is malicious (Fig. 7).

We have used skin detection algorithm proposed by Zarit et al. in [13] and created skin color model based on hue, saturation and intensity using six different skin samples for identifying skin pixels in the image. It may be possible that non-skin pixels are categorized as skin pixel in step 1, therefore we apply Canny Edge detection algorithm [14] in step 2 to filter out the useless information. In step 3, after refining the skin pixels, shape features of the skin region is extracted to classify the image as malicious or non-malicious. Three different shape descriptors are used for classification of the image as proposed in [14] and AdaBoost is used as classifier. In case of video, three random frames per second are extracted and analyzed using the same technique, if any of the frames contains malicious content malicious tag is assigned.

3.3.3. Classification

After performing text and video/image analyzes, Monitor Agent (MA) assigns other attributes of process vector set model as malicious or non-malicious based on the technique proposed by Manzoor et al. in [11]. After assigning tags to the attributes, MA substitutes all attributes into ontology concepts and assigns each concept a score. Text and video/image concept are assigned two where as rest of the attributes are assigned one score.

$$S_{Oi} = \sum_i Score(O_i) \quad (4)$$

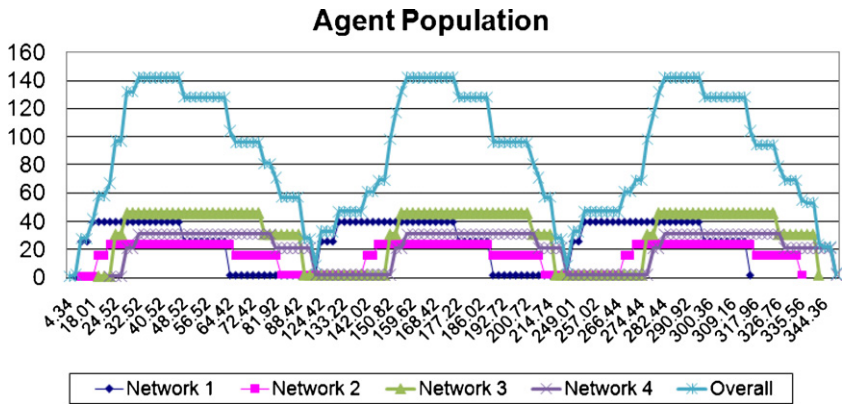

Fig. 8. Agent population over network.

MA calculates the score of each ontology class O_i by summing up all the individual scores of concepts in O_i and the ontology class which has the maximum weight is assigned to the process. After successfully assigning the ontology class, it loads the action to be performed against this class. MA uses the SysOpr executable to perform the action (i.e. kill, lockuser, etc.) and reports the violation along with user id, process, ip address, node id, etc. to SCA which updates its knowledge base and also forwards the report to CA.

3.4. Analyzer Agent (AA)

Dispatched from SCA, Analyzer Agent (AA) follows a list of nodes for verification. In initialization, AA takes two arguments (1) list of destinations to verify and (2) is the history of the nodes to be verified (i.e. ip address details, node id, softwares already installed, etc.) as hash table. AA uses the SysOpr executable to extract all the softwares installed on the network node and compare them with

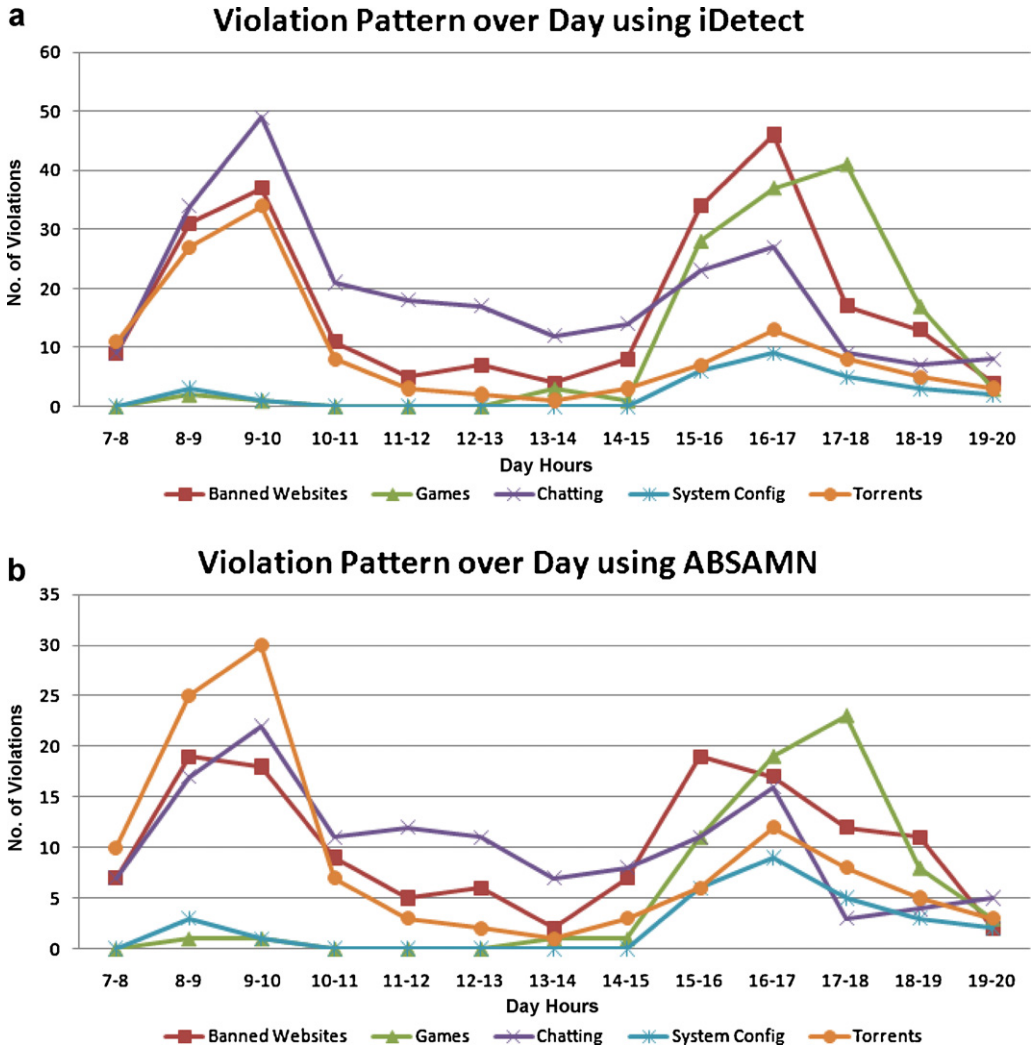

Fig. 9. (a) Violation Pattern over Day using iDetect. (b) Violation Pattern over Day using ABSAMN.

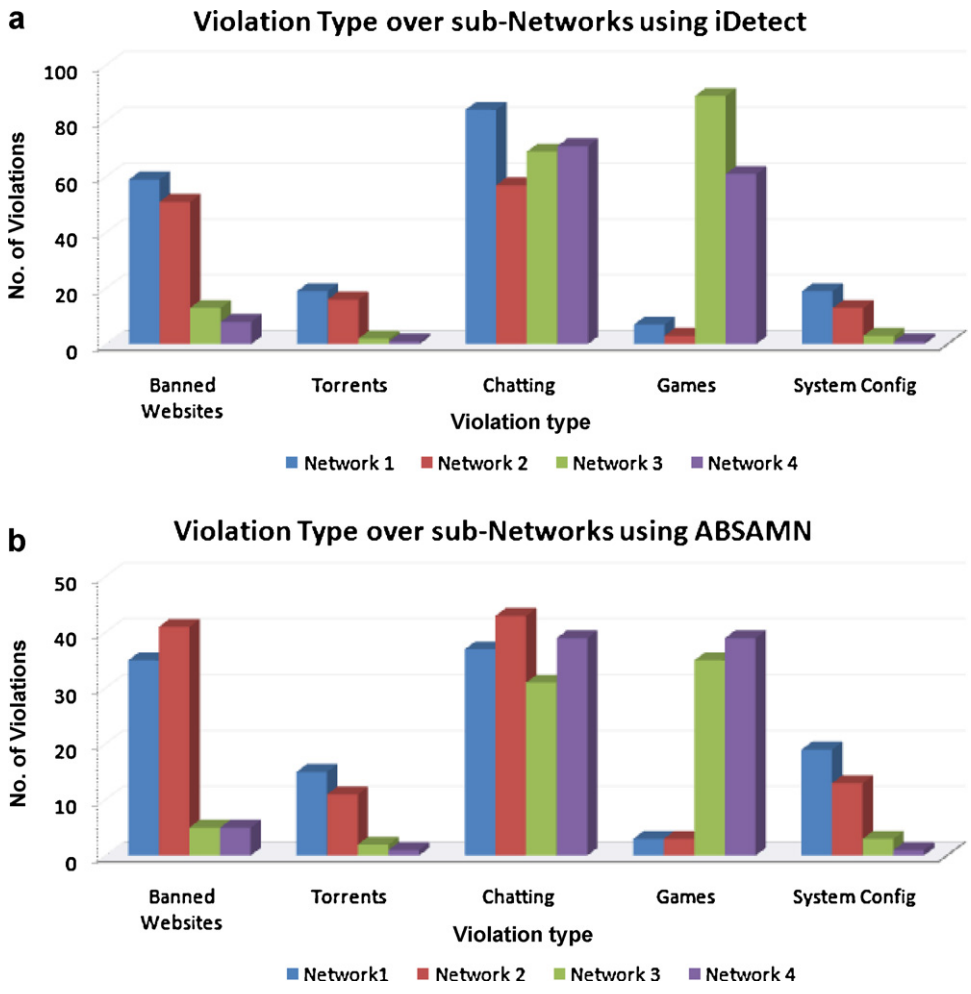

Fig. 10. (a) Violation type over sub-network using iDetect. (b) Violation type over sub-network using ABSAMN.

the history hash table. If AA captures any change, it reports SCA about the change with the node ID which forwards the report to CA for network administrator information. After verifying one node, Analyzer Agent moves to the next node in the list and repeats the same steps. After completing one round of verification Analyzer Agent sleeps for 60 s (configurable) and repeats the same steps afterwards.

4. Performance analysis

We have evaluated iDetect: Content Based Monitoring of Complex Networks using Mobile Agents and An Agent Based System for Activity Monitoring on Network (ABSAMN) on same configuration concurrently at the university campus having four labs equipped with 60–120 number of PCs running Microsoft Windows

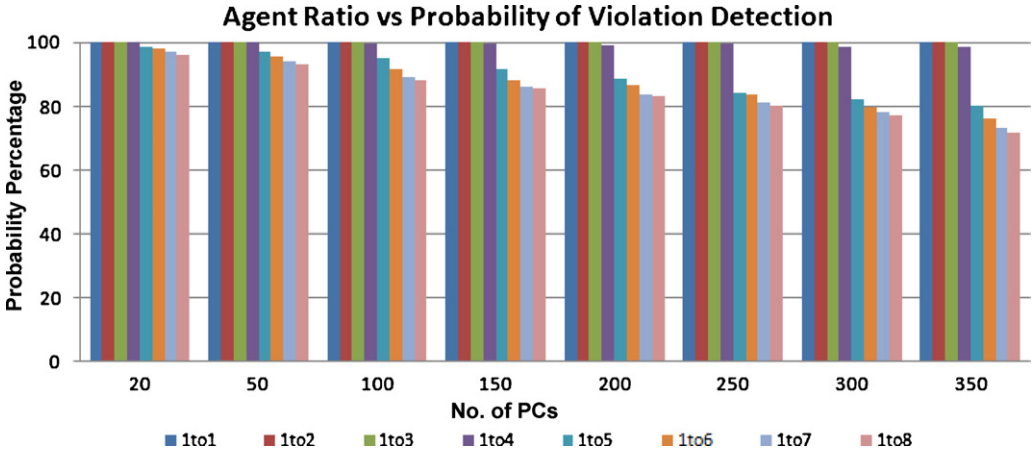

Fig. 11. Agent ratio vs. probability of violation detection.

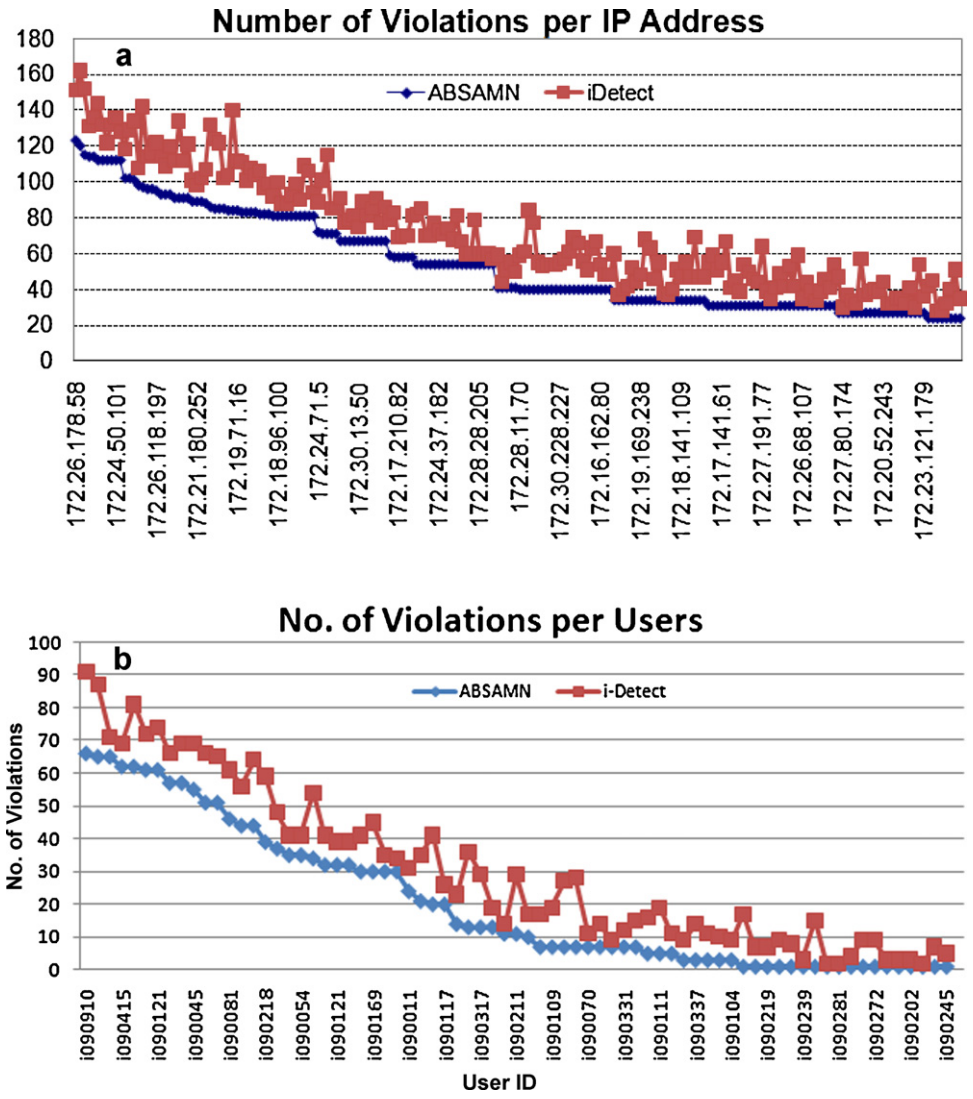

Fig. 12. (a) No. of violations per IP address. (b) No. of violations per users.

(any version) in various labs. Initially, one hundred volunteers were selected for experimentation in these labs and all of them were told about the system. After initial experimentation, we announced about the system on the university blackboard, walls/doors of the labs, etc. and open the labs for all users. The announcement clearly states that "Your activities will be monitored and the collected data will be used only for educational/research purpose". The activities were monitored for one month and the experimental results shows that iDetect efficiently detects known/unknown illegal activities (applications/websites) running on the network as compared to ABSAMN which only detects known illegal activities. Fig. 8 shows the agent population on different sub-networks and overall agent population over network required for monitoring of four labs having 60–120 PCs using i-Detect.

When the system starts up CA is created and it initializes the system autonomously, this activity takes 4.34 s as shown in Table 1. After initialization CA creates/initializes four SCAs sequentially and creating/initialing one SCA takes about 5.04 s. After a delay of 9.38 s from the start, the overall agent count is increased to 2 as one SCA is created and moved to network 1 which contains 100 network nodes (PCs). SCA (Network 1) creates/initializes 25 Monitor Agents (MAs) in parallel (i.e. one agent responsible to monitor 4 network nodes), this activity takes 3.89 s and network 1 agent count increases from 1 to 26 and these agents remains in the system for 64 s approximately. At the same time CA creates/initializes second SCA for network 2, after a total delay of 18.52 from start overall agent count increases to 41. Similarly SCA (network 2) creates/initializes MAs/AAs and agent count in network 2 increases and so on.

After initializing MAs, SCA (Network 1) creates/initializes 8 Analyzer Agents (AAs) in parallel (i.e. one agent responsible to verify system configuration of 8 network nodes), this activity takes 2.79 s and network 1 agent count increases from 26 to 40 and these agents remains in the system for 45 s approximately. As the analyzer agent complete one round of verification, the agent count reduces to 26 since all analyzer agents are deliberately moved from the network to the sub server.

Table 1
Average time taken by different operations.

Operation description	Average time (s)
Controller Agent Initialization	4.34
Sub-Controller Agent creation & initialization	5.04
Monitor Agent Creation & Initialization	3.89
Monitor Agent Monitoring Time	15.98
Analyzer Agent Creation & Initialization	2.79
Analyzer Agent Verifying Configuration Time	5.62
Agents Movement on the Network	1.02

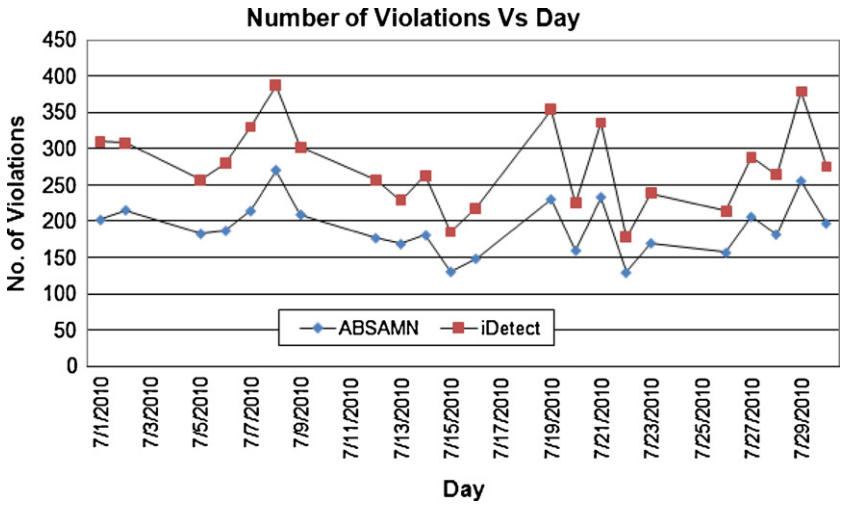

Fig. 13. Number of violations per day.

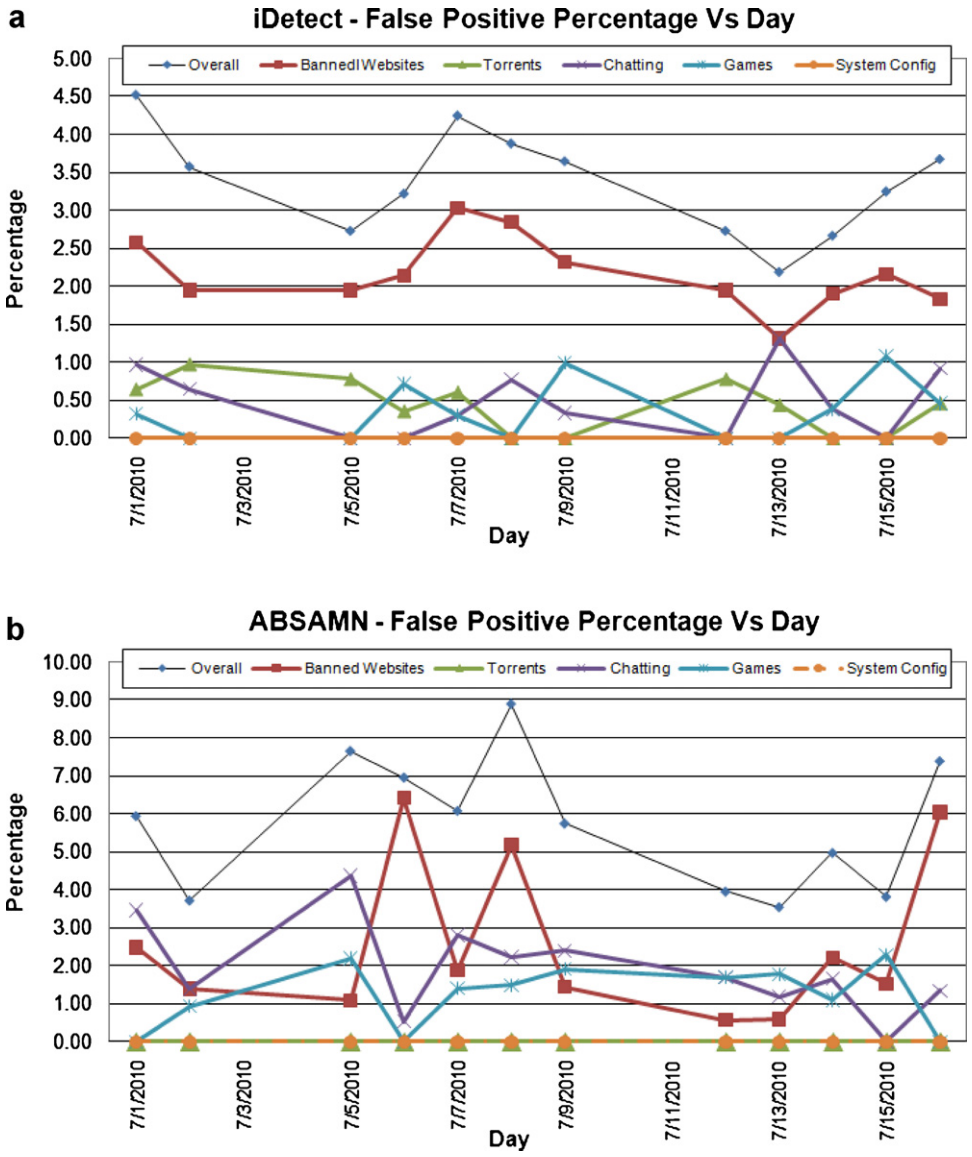

Fig. 14. (a) iDetect false positive percentage vs. day. (b) ABSAMN false positive percentage vs. day.

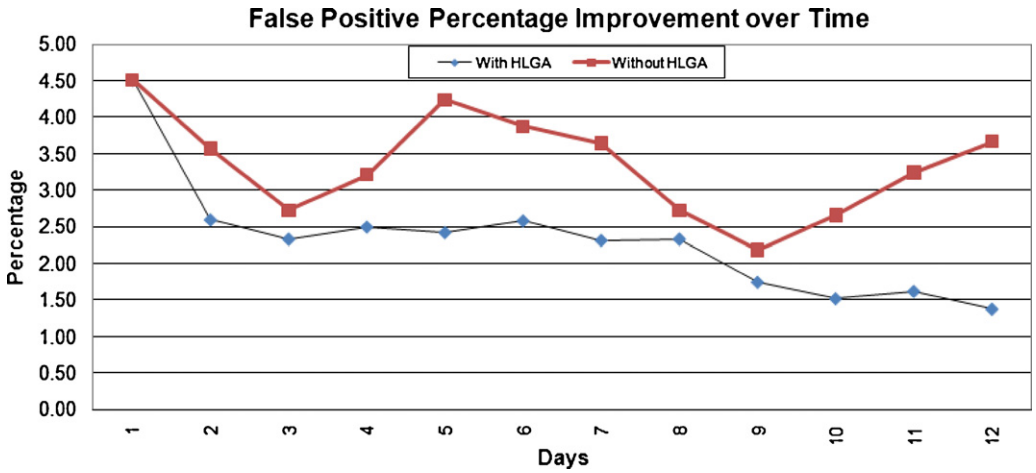

Fig. 15. False positive percentage improvement over time.

When the monitor agent complete one round of monitoring, the agent count reduces to 1 and after a delay of 60 s (configurable) second round of monitoring is started and again the agent count in the sub-networks increases as shown in Fig. 8.

Fig. 9(a) and (b) shows the violation pattern over the day captured using i-Detect and ABSAMN respectively, experimental results shows that maximum violations are performed between 7:00–9:00 AM and 3:00–6:00 PM. However, the pattern captured by ABSAMN is not correct because it shows Torrents is the most violated category from 7:00 to 9:00 whereas the correct most violated category is Chatting as shown in Fig. 9(b). Similarly, the violation type over sub-networks captured using ABSAMN shown in Fig. 10(b) is not correct because the system is unable to capture all the violations which leads to wrong statistics. Administrator of the network can view the violation type over sub-networks to observe which kind of violations are being performed on which sub-network as shown in Fig. 10(a), using this information it can change/update the restriction policy for the sub-network(s).

Fig. 11 shows the probability of violation detection with respect to varying monitor agent to network nodes ratio. Monitor agent takes about 15.98 s to scan a single PC, if monitor agent to network node is 1:4 monitor agent takes 63.92 to complete a single round. As we keep on increasing agent to network node ratio, probability of violation detection decreases because the scanning time to complete a single round increases and the violation performed on some other node during this period might not be captured as shown in Fig. 11.

Fig. 12(a) and (b) shows the number of violations per user and IP address captured using ABSAMN and i-Detect, experimental results shows that i-Detect captures more violation as compared to ABSAMN as the former has the ability to capture unknown illegal activities. This information can be used by the network administrator for securing the network by blocking/warning the users involved in these activities. Fig. 13 shows the number of violations captured per day using ABSAMN and i-detect simultaneously, the latter outperforms the former by capturing more violations performed over the network.

iDetect uses content (text, image, video) of the application/website to classify it as malicious or non-malicious whereas ABSAMN uses static rules for classification, in order to verify the accuracy of both classification algorithms, we calculated the false positive for each run (day) as shown in Fig. 14(a) and(b) respectively. The experimental result shows that for each run iDetect average false positive is 3.39 percent and ABSAMN average false positive is 5.72 percent; the iDetect maximum number of false positives captured for one run is 15 out of 387 activities out of which 11 were from Banned Websites category whereas for ABSAMN it is 24 out of 270 out of which 14 were from Banned Website category. Experimental result shows that iDetect classification algorithm is better than ABSAMN as the former false positive for each category

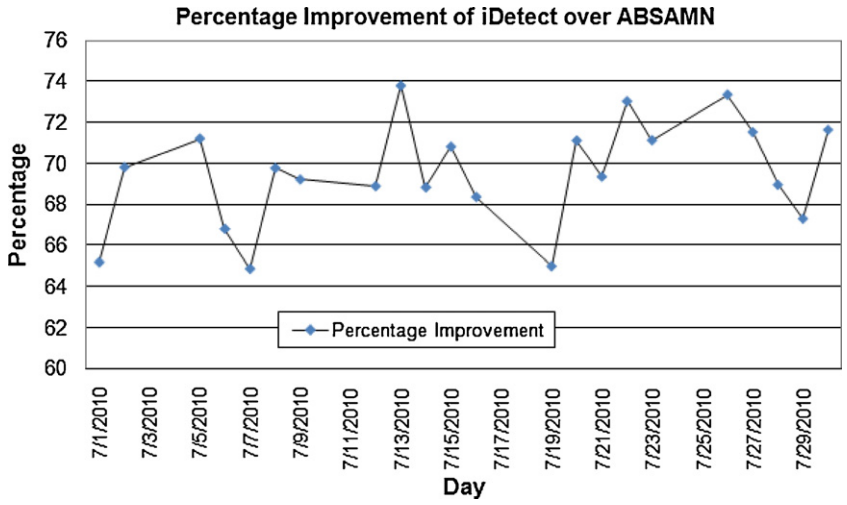

Fig. 16. Percentage improvement of iDetect over ABSAMN.

is less than the latter. Furthermore, the result shows that iDetect false positive of all categories except Banned Websites ranges from 0 to 1 percent whereas for Banned Websites it is between 0 and 3 percent. The reason behind the high false positive for Banned Website category is wrong classification of non-streaming websites (embedded streaming video shared inside webpage such as facebook) as streaming video websites. One possible way of detecting these kinds of non-streaming websites is to compare the video link with the website link; however, currently our framework fails to separate these kinds of websites.

Fig. 15 shows the improvement of false positives captured with and without High Level Guidance from Administer (HLGA) mechanism. Experimental result shows that iDetect HLGA mechanism significantly reduces the number of false positive over time as shown in Fig. 15. Fig. 16 shows the percentage violation capture improvement of iDetect over ABSAMN for various runs, result shows iDetect average violation capture improvement of 69.53 percent over ABSAMN.

5. Conclusion

In the last few decades, researchers have proposed many efficient methods for building large networks and efficient protocols for communication or information sharing. With the evolution in computer networks, many companies are trying to come up with automated monitoring tool(s) to make life easier for network administrator. In this paper we have proposed iDetect: Content Based Monitoring of Complex Network using Mobile Agents which uses the content/data (i.e. text, image and video) of the application to predict it as malicious or non-malicious. iDetect is fully autonomous and uses WordNet for text analysis and hybrid (skin and shape based) technique for malicious image detection. We have evaluated i-Detect on large test cases, results were very promising and shows an average improvement of 69.53 percent over ABSAMN with less than 5 percent false positive.

References

[1] P. Yolum, Design time analysis of multiagent protocols, Data & Knowledge Engineering 63 (1) (2007) 137–154.
[2] J.J. Jung, C.-M. Ou, T.N. Ngoc, G.K. Chong, Advances on agent-based network management, Journal of Network and Computer Applications 33 (November (6)) (2010) 631–632.
[3] L. Matthieu, W. Walter, Complex computer and communication networks, Computer Networks 52 (October (15)) (2008) 2817–2818.
[4] U. Manzoor, S. Nefti, An agent based system for activity monitoring on network – ABSAMN, Expert Systems with Applications 36 (October (8)) (2009) 10987–10994.
[5] Zend2 Online Anonymous Proxy, http://www.zend2.com, 2010.
[6] YouHide.com – Free Anonymous Proxy, http://www.youhide.com, 2010.
[7] U. Manzoor, S. Nefti, Autonomous agents: Smart Network Installer and Tester (SNIT), Expert System with Application (2010), doi:10.1016/j.eswa.2010.07.066.
[8] E. Costa-Montenegro, J.C. Burguillo-Rial, F. Gil-Castiñeira, F.J. González-Castañ, Implementation and analysis of the BitTorrent protocol with a multi-agent model, Journal of Network and Computer Applications (2010), doi:10.1016/j.jnca.2010.06.010.
[9] J. Wu, S. Yuan, S. Ji, G. Zhou, Y. Wang, Z. Wang, Multi-agent system design and evaluation for collaborative wireless sensor network in large structure health monitoring, Expert Systems with Applications 37 (3) (2010) 2028–2036.
[10] C. Garrigues, S. Robles, J. Borrell, N.-A. Guillermo, Promoting the development of secure mobile agent applications", Journal of Systems and Software 83 (6) (2010) 959–971.
[11] U. Manzoor, S. Nefti, Rezgui S Y., Autonomous malicious activity inspector – AMAI, in: International Conference on Applications of Natural Language to Information Systems, NLDB 2010, Cardiff, UK, June 23-25, Proceedings. Lecture Notes in Computer Science 6177 (2010) 204–215.
[12] How to Remove the Content Advisor Password in Internet Explorer or Fix Missing, Information, http://www.pchell.com/support/contentadvisor.shtml.
[13] B.D. Zarit, B.J. Super, F.K.H. Quek, Comparison of five color models in skin pixel classification, in: ICCV'99 Int'l Workshop on recognition, analysis and tracking of faces and gestures in Real-Time systems, 1999, pp. 58–63.
[14] Bill Green, "Canny Edge Detection Tutorial", http://www.pages.drexel.edu/∼weg22/can_tut.html, 2010.
[15] K9 Web Protection – Free Internet Filter and Parental Control Software, http://www1.k9webprotection.com/, 2010.
[16] Safe Eyes – Parental Control Software from Internet Safety, http://www.internetsafety.com/safe-eyes-parental-control-software-affiliate.php, 2010.
[17] Parental Controls, Internet Filter, Online Safety Software and Services, www.cyberpatrol.com/, 2010.
[18] U. Manzoor, S. Nefti, QUIET: a methodology for autonomous software deployment using mobile agents, Journal of Network and Computer Applications 33 (November (6)) (2010) 696–706.
[19] S. Venkatesan, C. Chellappan, P. Dhavachelvan, Performance analysis of mobile agent failure recovery in e-service applications, Computer Standards & Interfaces 32 (1–2) (2010) 38–43.
[20] Kim S C.O., Kwon S I.-H., C. Kwak, Multi-agent based distributed inventory control model, Expert Systems with Applications 37 (July (7)) (2010) 5186–5191.
[21] PC Tools Spyware Doctor, www.pctools.com/spyware-doctor/, 2010.
[22] Spyware Detector, http://www.spywaredetector.com/, 2010.
[23] Brintrup S A., Behaviour adaptation in the multi-agent, multi-objective and multi-role supply chain, Computers in Industry 61 (September (7)) (2010) 636–645.
[24] M.L. Sbodio, D. Martin, C. Moulin, Discovering semantic web services using SPARQL and intelligent agents, Web Semantics: Science, Services and Agents on the World Wide Web (2010), doi:10.1016/j.websem.2010.05.002.
[25] D.-H. Shih, H.-S. Chiang, B. Lin, Collaborative spam filtering with heterogeneous agents, Expert Systems with Applications 35 (November (4)) (2008) 1555–1566.
[26] Wu S L., Su S J., K. Su, X. Luo, Z. Yang, A concurrent dynamic logic of knowledge, belief and certainty for multi-agent systems, Knowledge-Based Systems 23 (2) (2010) 162–168.
[27] N.J.E Wijngaards, B.J. Overeinder, M. van Steen, F.M.T. Brazier, Supporting Internet-scale multi-agent systems, Data & Knowledge Engineering 41 (June (2–3)) (2002) 229–245.
[28] Q.-L. Guo, M. Zhang, An agent-oriented approach to resolve scheduling optimization in intelligent manufacturing, Robotics and Computer-Integrated Manufacturing 26 (February (1)) (2010) 39–45.
[29] M. Klusch, Information agent technology for the Internet: a survey, Data & Knowledge Engineering 36 (March (3)) (2001) 337–372.
[30] S.-H. Chen, Computationally intelligent agents in economics and finance, Information Sciences 177 (5) (2007) 1153–1168.
[31] J.P. Lage, A.S. da Silva, P.B. Golgher, A.H.F. Laender, Automatic generation of agents for collecting hidden Web pages for data extraction, Data & Knowledge Engineering 49 (May (2)) (2004) 177–196.
[32] S. Nefti, M. Oussalah, Y. Rezgui, A modified fuzzy clustering for documents retrieval: application to document categorization, Journal of the Operational Research Society 60 (March (3)) (2009) 384–394.
[33] WordNet, http://wordnet.princeton.edu/, 2010.
[34] Q.-F. Zheng, W. Zeng, G. Wen, W.-Q. Wang, Shape-based adult image detection, in: Proceedings of the Third International Conference on Image and Graphics, 2004, pp. 150–153.
[35] Java Agent Development Framework – JADE, http://jade.tilab.com/, 2001.