Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 920563, 5 pages
http://dx.doi.org/10.1155/2013/920563

Research Article

Immune Embedded Linux Core System with Multiple Sensors

Tao Gong[1,2,3]

[1] *College of Information S. & T., Donghua University, Shanghai 201620, China*
[2] *Engineering Research Center of Digitized Textile & Fashion Technology, Ministry of Education, Donghua University, Shanghai 201620, China*
[3] *Department of Computer Science, Purdue University, West Lafayette, IN 47907, USA*

Correspondence should be addressed to Tao Gong; taogong@dhu.edu.cn

Received 14 November 2012; Revised 28 March 2013; Accepted 28 March 2013

Academic Editor: Sabah Mohammed

The security of the embedded Linux core system is more important than before because this system is becoming more useful to solve the security problem of the computer system than the Windows operating system. It is firstly necessary to identify the normal state of the embedded Linux core system with a model. In this paper, the normal model of this normal system is represented with the space-time property set of all the components in this system. For example, the space property of a file in this system is the absolute pathname of this file, and the time property of this file is its last revision time. This immune embedded Linux core system uses multiple sensors to detect some important information, such as the disaster features. This antidisaster system will be tested on the ARM11 chips.

1. Introduction

As we know, an operating system such as Windows is often vulnerable to some viruses and attacks. This operating system must be updated with almost daily security patches, which cause new vulnerability due to the viruses or faults in the patches. It is true that the Linux is more secure than the Windows because the Linux is open at its source codes and can be enhanced in security by adding the security programs and customizing the source codes of the embedded Linux core system for special applications.

In fact, unknown viruses are difficult to be detected by traditional approaches [1] because the traditional security programs often detect the viruses with the feature matching of the viruses rather than those of the normal components. If these viruses are not prevented in the network, the network will be destroyed by the viruses in a short time [2].

To find the solution to the unknown viruses, we can seek inspiration from nature [3]. The human immune system really has advanced security mechanism to discriminate the selfs from the nonselfs and protect the body against the viruses, which are sometimes unknown [4]. The human immune system can detect the selfs first with immune tolerance to the selfs. We can build a normal model for the selfs to make the immune tolerance.

In this paper, by building an immune mechanism based on the normal model, an immune embedded Linux core system with multiple sensors is presented. In this system, the sensors are used to collect the data about the environment information of the disaster.

2. Related Work

In the histories of computers and networks, the viruses caused an increasing number of economic losses. On November 2nd, 1988, the Morris worms infected over 6000 Internet servers and the losses of these paralyzed servers were more than ten million dollars [5]. On July 19th, 2001, the CodeRed virus occurred and then caused the losses of more than 2 billion dollars. Moreover, the variants of this virus were more powerful and the variant CodeRed II caused the losses of more than 1.2 billion dollars. On September 18th, 2001, the Nimda virus occurred and then caused the losses of about 2.6 billion dollars.

To decrease the damage of the viruses, some network techniques were used to detect the viruses and to stop the

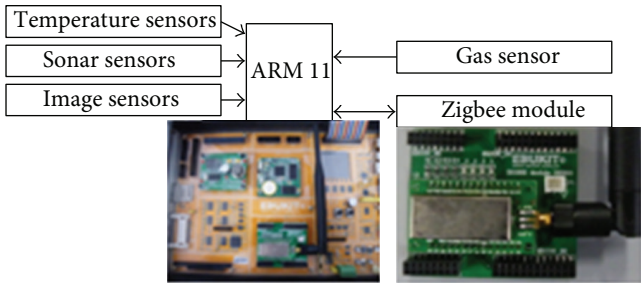

Figure 1: Frame of embedded Linux core system with ARM11 and multiple sensors.

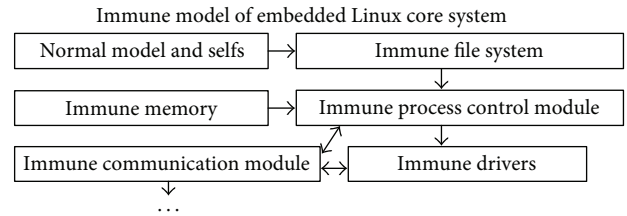

Figure 2: Immune model of embedded Linux core system.

spread [6–8]. The features of the viruses were analyzed and some models were built to describe the viruses and their spread. However, most of the traditional approaches cannot detect unknown viruses in real time, and the detection rates are not high enough. On the other hand, the variants of old viruses are continuously spread, and new viruses were increasingly designed. To overcome this bottleneck, it is necessary to design an immune embedded Linux core system based on the normal model of this core and more accurate self/nonself detection.

3. Embedded Linux Core System with Multiple Sensors

Suppose that this embedded Linux core system is composed of n_s antidisaster sensors, n_a ($n_s > n_a$) ARM11 chips, and n_a Zigbee modules, as shown in Figure 1. The antidisaster sensors include the temperature sensor, the sonar sensor, the image sensor (digit camera), and gas sensor and so on. The antidisaster sensors are used to collect all kinds of information about the disaster, and the Zigbee modules are used to transfer this information between an ARM11 node and another one.

This embedded Linux core system is built on the ARM11 chip, and the disaster environment information is collected by some sensors. The Zigbee modules make these ARM11 chips connected in a local wireless network. If a victim calls for help through an embedded system in Figure 1, the other embedded Linux core system will receive such information and activate the rescue procedure. If a software fault occurs in the embedded Linux core system, this system cannot be used to call for help or activate the rescue procedure. So it is very necessary and crucial to establish an intelligent mechanism to detect the faults and repair this damaged system in time. The immune system is one of the useful solutions for this problem.

4. Immune Model of Embedded Linux Core System

As we know, a complex standard Linux has too many components, so this operating system cannot be installed into the embedded system. Because the embedded system has limited memory to run programs, it does not need all the components of the standard Linux. So the standard Linux is trimmed into an embedded Linux, and the Linux core should be revised. The embedded Linux core system has necessary components such as files and directories. As shown in Figure 2, the immune model of this embedded Linux core system is built on its functional modules, which include file system, process control module, communication module, drivers, and memory management module.

5. Immune Algorithms for This Embedded Linux Core system

After analyzing the characteristics of the disaster samples of images, gas and temperatures, the real-coding affinity measure was used [9], and an immune algorithm was designed for recognizing the disaster on this embedded Linux core system with multiple sensors.

Input: N training samples $x_1, x_2, \ldots, x_N$ and N_o disaster feature vectors, whose information is collected from the antidisaster sensors and denoted as o_α, $\alpha = 1, 2, \ldots, N_o$.

Output: The recognition types of the disaster feature vectors $\{o_\alpha\}$.

Step 1. Initialize the parameters of the feature space, the feature-acquiring operator of each sensor, and the operator for searching the most similar sample.

Step 2. Build the feature space R^d with the feature vectors of the samples $x_1, x_2, \ldots, x_N$ and represent the antibody vectors Ab_m, $1 \le m \le N$.

Step 3. Collect the feature information from the antidisaster sensors, build the feature vector o_α with this feature information, and design the antigen Ag_α.

Step 4. Search the most similar sample of the object o_α with the real-coding clonal selection algorithm, by matching the antigen Ag_α with any antibody Ab_m, $m = 1, 2, \ldots, N$ according to that affinity measure.

Step 5. Calculate with uncertain reasoning and output the recognition type of the disaster feature vector o_α.

Step 6. Transfer the recognition result of the disaster feature vector o_α to other ARM11 chips.

```
temperature4_warning!
Ok!The value of combustible gas is:     1.0145
temperature1_warning!
temperature2_warning!
temperature3_warning!
temperature4_warning!
Ok!The value of combustible gas is:     1.0137
temperature1_warning!
temperature2_warning!
temperature3_warning!
temperature4_warning!
Ok!The value of combustible gas is:     1.0105
```

Figure 3: Abnormal temperature detection of the antidisaster sensor on the embedded Linux core system.

Moreover, another immune algorithm was also designed to build an initial normal model of the embedded Linux core system.

Step 1. Initialize the self-database.

Step 2. Make the backup system of the initial normal embedded Linux core system S and provide the root directory of the backup system.

Step 3. Read the root directory of the system S and search all the files in the root directory of this system.

Step 4. If there is at least one unread file or unread subdirectory in the current directory, then select one and read its absolute pathname and its last revision time; otherwise, set the ending condition satisfied and go to Step 7.

Step 5. Append the absolute pathname and the last revision time of this file or subdirectory as an immune record object into the self database.

Step 6. If this object is a file, then close the pointer of this file; otherwise, recursively build the normal model of the subsystem for this subdirectory.

Step 7. If the ending condition is satisfied, then end; otherwise, go to Step 3.

Based on the initial normal model and the rules for transforming a normal state to another one, the normal model of the immune embedded Linux core system can be updated dynamically.

With this normal model and the rules for transforming the normal states, the viruses and software faults can be detected by detecting the normal components (selfs) first in this embedded Linux core system. This detection algorithm is based on the self/nonself discrimination via the normal model.

Step 1. Initialize the infection database.

Step 2. Read the root directory of the system S and search all the files in the root directory of this system.

Step 3. If there is at least one unread file or unread subdirectory in the current directory, then select one and read its absolute pathname and its last revision time; otherwise, set the ending condition satisfied and go to Step 7. If the absolute pathname and last revision time of a file or directory are not matched with those of any record in the self database, then this file or directory is not a self; that is, this is a non-self, which may be a virus or software fault. Then this virus or software fault will be recognized and eliminated.

Step 4. Append the absolute pathname and backup pathname of the infected file or the infected subdirectory into the infection database.

Step 5. Recognize the viruses and software faults with feature matching and immune learning.

Step 6. Eliminate the viruses and software faults and then repair the damaged system.

Step 7. If the ending condition is satisfied, then end; otherwise, go to Step 3.

6. Experimental Results

Some experiments were conducted to test the antidisaster detection of the embedded Linux core system with antidisaster sensors. All the antidisaster sensors were tested on the ARM11 chips, and the data were analyzed on the PC machine, as shown in Figure 3.

From the monitor data of the antidisaster sensor, the temperature curve with the abnormal temperature is shown in Figure 4, and the gas voltage curve with the harmful gas from the sensor is shown in Figure 5.

The disaster recognition was built on the data from the multiple antidisaster sensors and the immune algorithm with clonal selection and real-coding affinity measure. The results of the recognition and the self/non-self detection were transferred between one ARM chip and another one, as shown in Figure 6. The received data showed the data source first and then the transferred data. When the victims need help in a disaster, the embedded Linux core system with the multiple antidisaster sensors were used to call for help and share the information about the disaster.

In these experiments, the immune Linux core was compiled with some ordinary compiling methods, and the step for compiling immune programs was started after building the normal model of this embedded Linux core system. After these immune programs were compiled, the files Makefile and Konfig were regenerated. The mirror of the Linux core used the zImage mirror, and this mirror was downloaded

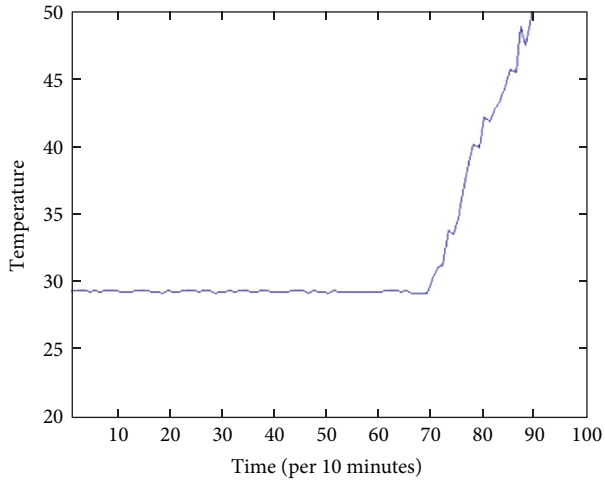

Figure 4: Temperature curve with abnormal temperature from the sensor.

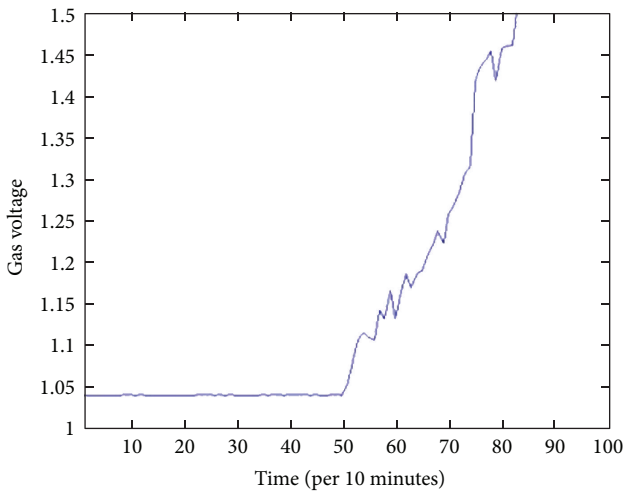

Figure 5: Gas voltage curve with harmful gas from the sensor.

```
T1 : 28.9375C

T2 : 28.9375C

T3 : 28.9375C

T4 : 28.9375C

Ok!The value of combustible gas is:     1.0645

T1 : 28.9375C

eceive data from :  0001
he data is :  123456789
T2 : 28.8750C

T3 : 28.8750C

T4 : 28.8750C

Ok!The value of combustible gas is:     1.0621
```

Figure 6: Data transferring between the ARM11 chips via the Zigbee modules.

```
Now building Normal Model…/home/core/rootfs/var/lib
Epoch Time is 1339382533
Now building Normal Model…/home/core/rootfs/home
Epoch Time is 1339382466
Now building Normal Model…/home/core/rootfs/opt
Epoch Time is 1345520051
Now building Normal Model…/home/core/rootfs/sys
Epoch Time is 1339382466

Normal Model has been built successfully!
File count is 672.
```

Figure 7: Building the normal model of this embedded Linux core system.

```
Now, start self/nonself detection...
Nonselfs detected are printed on the screen.

Nonself detected: /home/core/rootfs/etc/init.d
Nonself detected: /home/core/rootfs/etc/init.d/rcS~
Nonself detected: /home/core/rootfs/etc/init.d/rcS
Self/Nonself detection results are:
Selfs: 669, Nonselfs: 3
```

Figure 8: Self/nonself detection of Linux core system with immune programs.

```
Start system self-repairing...
cp: /home/core/backup/rootfs/etc /home/core/rootfs/etc
cp: omitting directory ‘/home/core/backup/rootfs/etc’
cp: /home/core/backup/rootfs/etc/group~ /home/core/rootfs/etc/group~
cp: /home/core/backup/rootfs/etc/group /home/core/rootfs/etc/group
cp: /home/core/backup/rootfs/etc/init.d /home/core/rootfs/etc/init.d
cp: omitting directory ‘/home/core/backup/rootfs/etc/init.d’
cp: /home/core/backup/rootfs/etc/init.d/rcS~ /home/core/rootfs/etc/init.d/rcS~
cp: /home/core/backup/rootfs/etc/init.d/rcS /home/core/rootfs/etc/init.d/rcS
cp: /home/core/backup/rootfs/etc/passwd /home/core/rootfs/etc/passwd
cp: /home/core/backup/rootfs/etc/passwd~ /home/core/rootfs/etc/passwd~
```

Figure 9: System repairing based on the normal model and immune programs.

online into the development board to test. These immune programs were tested to build a normal model for this immune Linux core system, as shown in Figure 7.

After this embedded Linux core system was changed by some nonselfs, the nonselfs were detected by the immune programs, as shown in Figure 8.

All the nonselfs were eliminated, and then the damaged Linux core system was repaired with the normal model and the immune programs, as shown in Figure 9.

7. Conclusions

With the increasing requirements about the security of the operation system and network, the embedded Linux core system is more crucial to the security applications. Inspired by the nature and beyond the traditional security approaches, the model of immune computation becomes a new effective tool to enhance the security of the Linux operatiing system from the core to some applications. In this research, the immunization mechanism is embedded into the Linux core

system to keep its functional modules, such as the file system, immune, and secure.

Acknowledgments

This work was supported in part by grants from the National Natural Science Foundation of China (61271114), the Natural Science Foundation of Shanghai (08ZR1400400 and 11ZR1401300), the Shanghai Postgraduate Education Creative Plan (SHGS-KC-2012OO3), the Shanghai Educational Development Foundation (2007CG42 and 12CG35), and the Fundamental Research Funds for the Central Universities at Donghua University (13D110414). The author thanks his graduate student Changxing Du for the help in programming.

References

[1] O. Sukwong, H. S. Kim, and J. C. Hoe, "Commercial antivirus software effectiveness: an empirical study," *IEEE Computer*, vol. 44, no. 3, Article ID 5506074, pp. 63–70, 2011.

[2] J. Balthrop, S. Forrest, M. E. J. Newman, and M. M. Williamson, "Technological networks and the spread of computer viruses," *Science*, vol. 304, no. 5670, pp. 527–529, 2004.

[3] T. Gong and C. X. Du, "Immune computation of secure embedded linux core against viruses and software faults," *International Journal of Security and its Applications*, vol. 6, no. 2, pp. 167–172, 2012.

[4] R. Medzhitov and C. A. Janeway, "Decoding the patterns of self and nonself by the innate immune system," *Science*, vol. 296, no. 5566, pp. 298–300, 2002.

[5] H. Orman, "The morris worm: a fifteen-year perspective," *IEEE Security and Privacy*, vol. 1, no. 5, pp. 35–43, 2003.

[6] E. Levy, "Worm propagation and generic attacks," *IEEE Security and Privacy*, vol. 3, no. 2, pp. 63–65, 2005.

[7] B. Madhusudan and J. W. Lockwood, "A hardware-accelerated system for real-time worm detection," *IEEE Micro*, vol. 25, no. 1, pp. 60–69, 2005.

[8] I. Arce and E. Levy, "An analysis of the slapper worm," *IEEE Security and Privacy*, vol. 1, no. 1, pp. 82–87, 2003.

[9] T. Gong, "High-precision immune computation for secure face recognition," *International Journal of Security and Its Applications*, vol. 6, no. 2, pp. 293–298, 2012.