HoneyID : Unveiling Hidden Spywares by Generating Bogus Events

Jeheon Han, Jonghoon Kwon, Heejo Lee

Abstract A particular type of spyware which uses the user's events covertly, such as keyloggers and password stealers, has become a big threat to Internet users. Due to the prevalence of spywares, the user's private information can easily be exposed to an attacker. Conventional anti-spyware programs have used signatures to defend against spywares. Unfortunately, this mechanism cannot detect unknown spywares. In this paper, we propose a spyware detection mechanism, called HoneyID, which can detect unknown spywares using an enticement strategy. HoneyID generates bogus events to trigger the spyware's actions and then detects hidden spywares among running processes which operate abnormally. We implemented the HoneyID mechanism as a windows based, and evaluated it's effectiveness against 6 different known spywares(3 keyloggers and 3 ftp password sniffers). From this study, we show that the HoneyID can be effective to detect unknown spywares with high accuracy.

1 Introduction

Spyware has become one of the most serious Internet phenomena. A recent report stated that 9 out of 10 computers connected to the Internet are infected [6]. However, most users are unaware of the presence of spywares due to the evasive behavior. This makes it difficult for the user to prevent it from causing damage.

Today, many signature based anti-spyware solutions have been used to defence from spyware. However, these solutions cannot detect unknown spywares. Therefore, the users are exposed to the threat of spyware before the corresponding signature update. Behavior-based spyware detection has been studied in several ways such as Gatekeeper [5], Behavior-based Spyware Detection [3]. Although these behavior-based approaches provide good detection results for particular behavior,

J. Han, J. Kwon and H. Lee are with the Division of Computer and Communication Engineering, Korea University, Seoul 136-713, Korea. e-mail: {getroot, signalnine, heejo}@korea.ac.kr. This work was supported in part by the ITRC program of the Korea Ministry of Knowledge Economy.

they have less eppectiveness against spywares which are triggered by a user's activity. In addition, there are also flow based approaches to detect spywares which use a honeytoken. For example, Siren [1], SpyCon [2] were proposed, which induce a spyware to make additional network request. Unfortunately, these approaches cannot detect spywares which do not send the stolen user's information to a remote server and cannot obtain infected spywares information. For solving these problems, in this paper, we propose a novel spyware detection mechanism called HoneyID.

2 Detection Mechanism

HoneyID is a mechanism that detects dialog spyware processes actively in a local machine. The dialog spyware is one of the most significant spyware, since it works using specific user activity. To steal and handle the user's activities, it uses generated events. Thus, HoneyID causes dialog spywares to fall into a trap by generating specific bogus events.

HoneyID consists of the trap and bogus events, as shown in Fig. 1. The trap is a component which monitors the changes of each process and the bogus event is a mimic user event which can make the dialog spyware operate. To detect spyware processes, HoneyID sets up a trap in the operating system and generates bogus events. When these events induce a dialog spyware behavior, HoneyID can detect dialog spyware processes by checking the changes of the processes.

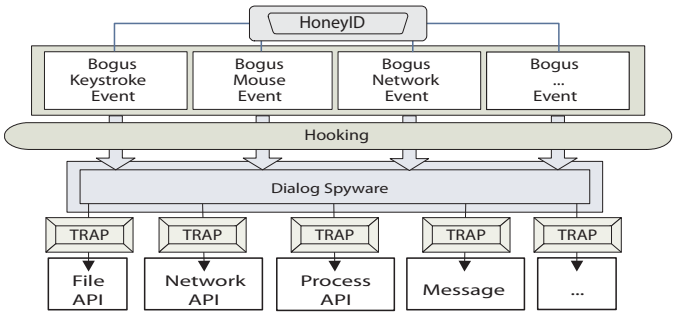

Fig. 1 The basic concept of HoneyID to detect dialog spywares.

If a process responds to bogus events whenever they are generated, HoneyID classifies it as a dialog spyware. Hence, dialog spyware can be classified by Eq. (1), where E is the number of bogus events generated per second, N is the period during which the bogus events occur continuously and T is the number of jobs of the process used for transacting bogus events. The job is a set of operations required for the target process.

$$T = E \cdot N \tag{1}$$

A dialog spyware steals and processes generated events immediately before the next event generation. Otherwise, it will fail to steal the next events. Therefore, the number of jobs of a dialog spyware process invoked by the bogus events and the number of generated bogus events is the same. So satisfying Eq. (1) means that the process can be regarded as a dialog spyware.

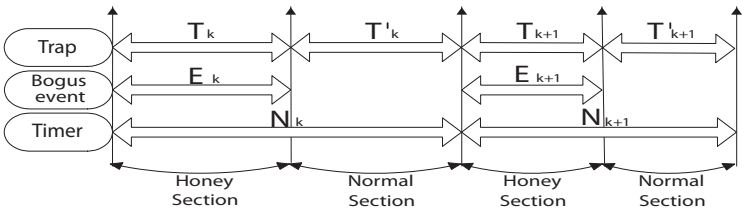

Fig. 2 Divided two sections with variables

In order to classify the jobs of normal processes and dialog spyware processes, HoneyID separates them into two sections. The first one is honey section with bogus event and other one is normal section. If HoneyID begins to generate bogus events and pauses repeatedly, and a process operates only in the honey section, HoneyID can check that the process operates as a result of the bogus events. If this method is repeated k times, the probability that a normal process runs in each honey section is $\frac{1}{2}^k$. Hence, the probability of a false alarm is very low.

Fig. 2 shows the two sections with three variables. The trap counter counts the number of jobs in both sections, the bogus event counter counts the number of bogus events per second, and the timer measures the time of each section. Eq. (1) can be represented by the divided section as the process reaction degree defined by Eq. (2).

$$R_k = \frac{T_k - T_k'}{E_k \cdot N_k / 2} \tag{2}$$

Eq. (2) measures the number of jobs induced by the bogus event. T_k include the number of malicious jobs and normal jobs while T_k' includes only normal jobs. $T_k - T_k'$ becomes the number of malicious operations because the normal jobs are distributed both sections. In the case of a dialog spyware process, R_k will become one and normal process. From this distinctiveness, HoneyID can determine the threshold to distinguish between normal processes and spywares to be 0.5.

There is some possibility of errors in R_k because a dialog spyware could use delayed processing. This error can be reduced by obtaining the average value of R_k after z times iterations of C_k. Therefore, HoneyID can detect the dialog spyware process with less error by calculating the reaction degree(Φ) using Eq. (3).

$$ReactionDegree(\Phi) = \frac{\sum_{k=0}^{z} R_k}{z} \quad (3)$$

3 Evaluation

The HoneyID architecture is composed of three modules, the trap manager, the bogus event generator and the spyware detector. The trap manager sets up traps in the operating system and gathers the number of jobs of processes. The bogus event generator generates bogus events in response to commands from the spyware detector. The spyware detector controls the other two modules and measures the process condition used to distinguish between normal processes and spyware processes.

We developed two types of bogus events, the virtual keystroke event using the SendInput API function and the virtual ftp login event modifying the CInternetSession of MFC, respectively. The trap is implemented by means of the Win32 hook and Paladin Hook technique [4], for monitoring at the kernel level.

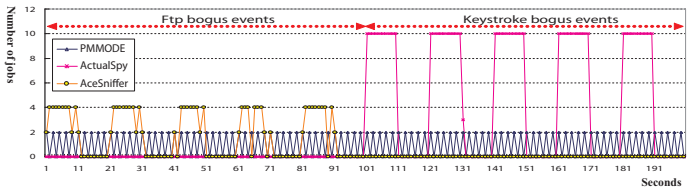

Fig. 3 The number of jobs of processes according to bogus events in a second

We collected three keyloggers and three ftp password stealers. Fig. 3 shows the fluctuations in the number of jobs of three processes : the normal process(PMMODE), the keylogger(ActualSpy) and the password stealer(AceSniffer). HoneyID generates ftp connection bogus events for the initial 100 seconds, and then generates keystroke events for the next 100 seconds. We can see that the PMMODE works irrespective of the bogus events, whereas the AceSniffer and the ActualSpy only works when relational bogus events are generated.

Table 1 show the results obtained from the experiment designed to detect keyloggers and password stealers, respectively. HoneyID operated for 100 seconds and was repeated five times. Bogus events for keyloggers are generated 10 times per second. The three keyloggers were identified. Bogus events for the password stealer are generated 3.6 times per second. Since an ftp connection event needs more resources, we generate ftp connection events as often as possible. The password stealers were also correctly identified.

When HoneyID was running, there were 36 processes. HoneyID found the three keyloggers that were installed and there were no false positives. One of the Internet messenger program is in close proximity to the threshold 0.5. This process may

Table 1 Password stealer detection (E=3.6 & N=100) and keylogger detection(E=10 & N=100).

Process	R_1	R_2	R_3	R_4	R_5	Φ	—	Process	R_1	R_2	R_3	R_4	R_5	Φ
Winsniffer	0.83	0.94	1	1	0.94	0.94	—	Family	1	1	1	1	1	1
Sniffpass	1	0.89	0.89	0.89	0.78	0.89	—	ActualSpy	1	0.97	1	1	1	0.99
AceSniffer	0.94	1	0.89	0.78	0.78	0.88	—	BPK	1	1	1	1	1	1

make use of keystroke events to notify friends of the user's status. When password stealers were found, the alg process is identified as the password stealer with no false negatives. However, the alg process is a core process for Windows. Since the alg process monitors ftp packets, HoneyID identifies the alg process as a password stealer. However these false positives can be solved by means of a white list filter.

4 Conclusion and Future Work

In this paper, we present a novel mechanism that detects spyware actively in a local machine. It causes spywares to exhibit malicious behavior by generating bogus events and it can detect dialog spyware processes by checking response to these bogus events. This approach is a powerful method of detecting dialog spywares that use the information of a user or PC. Future work will focus on extending our approach. We also plan to experiment on a large scale with a large number of spywares and normal applications installed.

References

1. Borders, K., Zhao, X., Prakash, A.: Siren: Catching evasive malware (short paper). In: Proceedings of IEEE Symposium on Security and Privacy (2006)
2. Chandrasekaran, M., Vidyaraman, S., Upadhyaya, S.: Spycon: Emulating user activities to detect evasive spyware. In: Proceedings of IEEE Int'l Conf. on Performance, Computing and Communications (IPCCC) (2007)
3. Kirda, E., Kruegel, C., Banks, G., Vigna, G., Kemmerer, R.A.: Behavior-based spyware detection. In: Proceedings of the 15th USENIX Security Symposium (2006)
4. pudn.com: HookAPI Source Code (2005). http://www.codeproject.com/system/Paladin.asp
5. Wang, Y.M., Roussev, R., Verbowski, C., Johnson, A., Wu, M.W., Huang, Y., Kuo, S.Y.: Gatekeeper: Monitoring auto-start extensibility points (aseps) for spyware management. In: Proceedings of Usenix Large Installation System Administration Conference(LISA) (2004)
6. Webroot Software, Inc.: Spyware info and facts that all internet users must know (2006). http://www.webroot.com/resources/spywareinfo