Vol.2018-IOT-42 No.6
2018/6/28

On a Finite State Machine and Input Fields for Incident Tracking System

Motoyuki OHMORI[1,a)] Masayuki HIGASHINO[1,b)] Toshiya KAWATO[1,c)]

Abstract: In order to appropriately and quickly handle a security incident, ones may need Incident Tracking System (ITS) that records facts: what happens, when happens, who handles and how. It may be, however, difficult for a person in charge of incident handling to input all detailed information to ITS, and ITS should have minimal but enough information for further incident handling. In addition, a person in charge should be able to operate ITS intuitively since an incident does not happen so often. It is, however, unclear what information ITS should hold and how ITS navigates a person in charge to complete incident handling. This paper discusses these issues, and introduces our implementation and usage of ITS using Redmine within Computer Security Incident Response Team (CSIRT).

1. Introduction

Computer security has been getting more attentions because a computer security incident may cause great damage on an organization. Since it is difficult to avoid all incidents to happen, a proper and quick response against an incident is important in order to mitigate or minimize damage. To this end, it is now becoming common that an organization forms Computer Security Incident Response Team (CSIRT).

In many cases, a malicious communication is detected by an external organization such as Japan Security Operation Center (JSOC) [1] operated by LAC Co., Ltd, National Institute of Informatics Security Operation Collaboration Services, the so-called NII-SOCS, operated by National Institute of Informatics (NII) [2], government organizations or others. A CSIRT in an organization then firstly recognizes a computer security event after receiving an alert of a suspicious communication from an external organization. The CSIRT then makes a triage decision whether the event should be handled as an incident or not. If the event is considered as an incident, the CSIRT then initiates an incident response. During an incident handling, it is important to record facts, e.g., what happens, when happens, who handles and how, and share them among all people involved in the handling.

To this end, we, Tottori University, had introduced an Incident Tracking System (ITS) using Redmine [3], which is one of famous Bug Tracking System (BTS), as an experiment within CSIRT on January 2017. We have been operating ITS in actual production environment within CSIRT since April 2017. ITS manages an incident as a *ticket*, and information about an incident is recorded into an associated ticket. We have then found that it was not really easy for CSIRT members to input or change a value of a field on ITS due to some reasons:

- CSIRT members were not familiar with ITS which is a tool for a software developer,
- there were many fields to input,
- it was difficult to remember an order of a field to input,
- CSIRT members never referred to ITS manual during an actual incident,
- CSIRT members never remembered all operations of ITS because an incident rarely occurs,
- CSIRT members could not practice enough in normal times for some reasons such as busyness for other tasks, no motivation and so on.

We have then realized that ITS should be easy to intuitively operate even without any practice or manual.

To this end, this paper will present how to design Finite State Machine (FSM) and input fields of ITS using Redmine. We have found that:

- status of a ticket of an incident on ITS can be combined with *handling*, *uncritical*, *ball*, i.e., who is in charge of, and *done*.
- this combined status can navigate CSIRT members to easily and intuitively change FSM,
- *workflow* of Redmine works well for ITS, and
- most of many fields are unnecessary to input in many security events because most of security events are not critical security incident and they should be hidden if unnecessary.

In this paper, let us assume that only CSIRT members use ITS while other organization members do not.

The rest of this paper is organized as follows. Section 2 presents what ITS is. Section 3 describes why the default status in Redmine is not suitable for ITS, and how we have improved the status. Section 4 proposes FSM for ITS. Section 5 proposes fields for ITS. Section 6 refers to related work. Section 7 finally concludes this paper.

[1] Tottori University, Koyama-minami, Tottori Japan, 680–8550 Japan
[a)] ohmori@tottori-u.ac.jp
[b)] higashino@tottori-u.ac.jp
[c)] t.kawato@tottori-u.ac.jp

2. Incident Tracking System (ITS)

ITS is in charge of sharing information among CSIRT, recording actions that CSIRT takes and observed phenomenon, and make an incident trackable. ITS must be able to:

(1) share information among CSIRT members involved in a security incident response,
(2) issue a ticket for an incident,
(3) differentiate *open* and *closed* issues.
(4) associate the similar incidents with a ticket,
(5) register CSIRT member in advance,
(6) notify CSIRT involved of updates of an incident,
(7) upload a file for an incident,
(8) automatically produce a final report of an incident, and
(9) automatically produce a summary of incidents during specified duration.

ITS can then be built using an exiting Bug Tracking System (BTS) or issue tracking system [3], [4], [5]. ITS, however, needs to assign an incident to a group of CSIRT members while BTS usually assigns to a one person. ITS is very different from BST or issue tracking system in this point. In this paper, we use Redmine [3] as ITS.

3. Status of a Ticket

This section presents what problems we faced regarding status of a ticket, and how we have solved.

We firstly faced the problem that CSIRT members did not *close a ticket* even after the incident handling was over. From the point of view of a software developer, it is extremely common to close a ticket after a bug or problem is solved. Most of CSIRT members, unfortunately, had not experienced to develop a information system from a scratch in real environment or in commercial use. They were, hence, not accustomed to close a ticket. They could not then close a ticket even our incident handling manual said to close a ticket after the incident handling finished.

We secondly faced the problem that it was unclear who was a person in charge and who should have been currently responsible to take an action. For example, let us assume that an external organization notify us of a suspicious communication. In this case, we need to compute a private IP address of a suspicious host from the notified global IP address because we adopts NAT or NAPT for all hosts in our campus network. In our organization, CSIRT is responsible to compute a private IP address from the global IP address. It was, however, difficult for CSIRT to notice at a glance whether this computation was required or not. We had then introduce new input field, *ball*, that indicated who, i.e., CSIRT, a department or a user, was in charge of an incident. This field was, however, not always updated because a person in charge could not notice that he or she should have updated the field. Even the field was properly updated, almost all CSIRT members did not check to see a *ball* field, and did not join an incident handling.

We thirdly faced the problem that CSIRT member could not understand when they could close a ticket. Redmine unfortunately cannot define a detailed condition onto each field by default when a ticket can be closed. Even such a detailed condition can be defined, it would be complicated and difficult for CSIRT members to understand which field should have what value.

Table 1 Status of a ticket.

Status
identification (CSIRT)
awaiting identification (department)
data breach investigation (CSIRT)
data breach investigation finished (CSIRT)
awaiting final report (department)
awaiting OS re-installation (student)
false positive (done)
uncritical (done)
confirmation operation (done)
the same host as other incident (done)
out of scope of CSIRT (done)
finished (done)

We have then solve these problems using *workflow* in Redmine. In order to adopt *workflow*, we firstly have modified and defined *status* of Redmine like below.

$$
\begin{aligned}
\mathit{status} &::= \mathit{type}\ \text{"("}\ \mathit{ball}\ \text{")"} \\
\mathit{type} &::= \mathit{handling} \mid \mathit{uncritical} \\
\mathit{ball} &::= \text{"CSIRT"} \mid \text{"department"} \mid \text{"user"} \mid \mathit{done} \\
\mathit{done} &::= \text{"done"}
\end{aligned}
$$

As ones can see in above definition, we have combined status with *handling*, *uncritical*, *ball* and *done*, i.e., finished status. We have actually defined status of a ticket as shown in **Table 1** in our Redmine. We have then instructed CSIRT members to go toward *done* state.

4. FSM for ITS

Using *combined status* as defined in 3, we define FSM of our ITS as shown in **Fig. 1**. In Fig. 1, each box and arrow represent status and an event, respectively. Blue boxes represent *open* status. On the other hand, green boxes represent *closed* status. All green status except for *finished (done)* can be moved from all status. As shown in Fig. 1, all events changes status toward to *closed* status, and there is no event that goes back toward initial status. In addition, all blue boxes have two or less arrows, that is, there are only two choices at maximum when status is changed except for *closed* status. As ones can also see in Fig. 1, lesser critical incident requires lesser status changes. While a really critical incident rarely happens, false positive detection often occurs in our environment. This nature decreases operations that CSIRT member must do on an incident handling. We have then implemented this FSM in Redmine using *workflow*.

5. Input Fields of a Ticket

When ones handle an incident, there are many things to interview, clarify and record. We define then information that ITS should hold as shown in **Table 2**. Note that boolean is not used in order to allow empty even though Redmine has a value type of boolean. Boolean values are listed as *list* in Table 2.

As shown in Table 2, there are currently 49 fields defined in our ITS while there is no unnecessary filed. We faced the problem that it was difficult for CSIRT member to find out which field should have been input. Even though there is no unnecessary

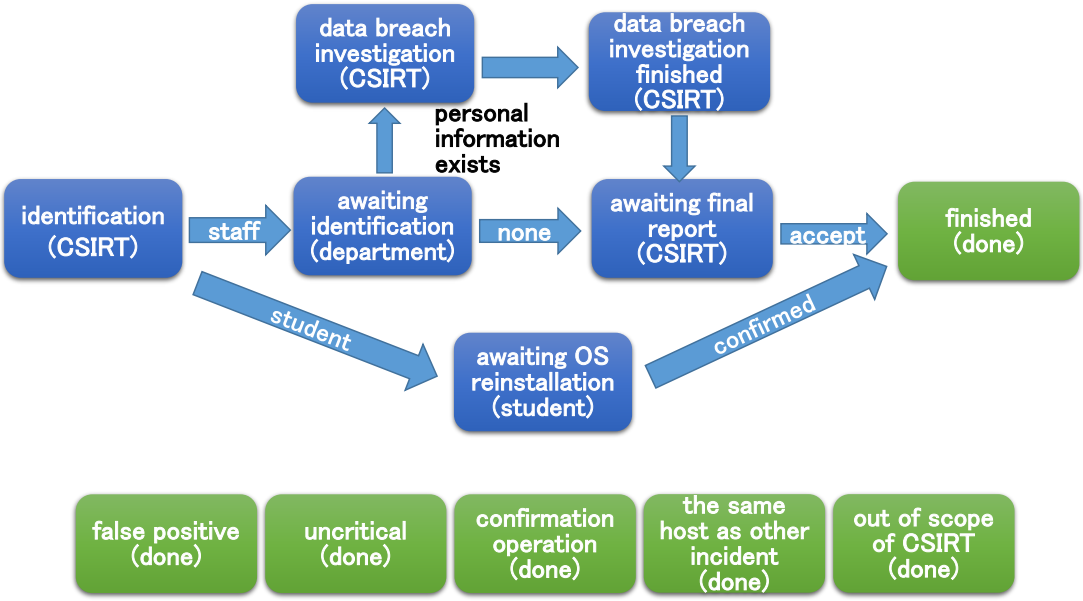

Fig. 1 FSM on ITS.

field, all fields are not *always* necessary. For example, let us assume that a PC gets infected with malware, and the PC does not contain any confidential information. In this case, ones do not need to preserve all data stored in the PC for digital forensic since there is no possibility of data breach. Ones do not then need input fields regarding digital forensic. As described above, it depends upon status which field should be input or not.

In order to reduce fields which are displayed in front of CSIRT member, we have utilized privilege control of Remine. **Fig. 3** shows our privilege control for each status and each field. In Fig. 3, "*" represents required field, and "-" represents read only field which is hidden. A blank means that the field is displayed on the status.

6. Related Work

Information Security Management System (ISMS) ISO/IEC-27001[7] briefly defines requirements of computer security incident responses. There are many security or network vendors such as TrendMicro, Paloalto, FireEye, Fortigate, Cisco, Alaxala and so on try to produce the best security solutions.

NAGAI, Y. et al. investigated and reported differences between ISMSs in national universities in Japan[8]. They also presented their own incident management system using trac[4]. They then reported that their system could record information of only about a half of all security events because some of those events were reported or discussed in meetings and their data was never input to the system.

HASEGAWA, H. et al. proposes the supporting system against an incident caused by targeted attacks [9]. Their system automatically suggests 9 types of access filtering across VLANs to an administrator in accordance with a severity of an incident when a network configuration is pre-defined and given. They, however, assumes only filtering across VLANs, and do not consider the case where there is a router run by a department, not a information infrastructure department that is in charge of a management of a campus wide network. In addition, they do not consider a mobile host that moves around while our proposal do.

Request Tracker for Incident Response (RTIR) [6] is a famous ITS written in Perl. There are also BTSs or ITSs such as trac[4] written in Python, mantis[5] written in PHP and so on. We will try to find the best system for our purpose.

7. Concluding Remarks

This paper has proposed FSM and input fields that would be suitable for ITS using Redmine. We have found that status of a ticket of an incident on ITS can be combined with *handling*, *uncritical*, *ball*, i.e., who is in charge of, and *done*, while there are usually only two types of status of a ticket, *open* and *closed*. This nature have simplified CSIRT operations, and enabled CSIRT members to close a ticket after an incident handling finishes. It is future work to deploy our ITS not only within CSIRT but also to an end user.

*1 automatically generated.
*2 Redmine built-in field.

References

[1] LAC Co., Ltd: Japan Security Operation Center(JSOC®) — Services and Products — LAC Co., Ltd., https://www.lac.co.jp/english/service/operation/jsoc.html (1995). Accessed: 2017/05/26.
[2] National Institute of Informatics: National Institute of Informatics, http://www.nii.ac.jp/ (2007). Accessed: 2017/05/26.
[3] Lang, J. P.: Overview - Redmine, http://www.redmine.org/ (2006). Accessed: 2017/05/19.
[4] Software, E.: The Trac Project, https://trac.edgewall.org/ (2003). Accessed: 2017/05/19.
[5] MantisBT Team: Mantis Bug Tracker, https://www.mantisbt.org/ (2000). Accessed: 2017/05/19.
[6] Best Practical Solutions, L.: RT for Incident Response, https://bestpractical.com/rtir/ (2002). Accessed: 2017/05/19.
[7] ISO/IEC: Information Security Management Systems Requirements (2013). ISO/IEC27001:2013.
[8] NAGAI, Y., TADAMURA, K. and OGAWARA, K.: Considering Incident Management Systems in Some National Universities, *SIG Technical Reports*, Vol. 2014-IS-127, No. 7, pp. 1–7 (2014).
[9] Hasegawa, H., Yamaguchi, Y., Shimada, H. and Takakura, H.: A Countermeasure Support System against Incidents caused by Targeted Attacks, *Journal of Information Processing*, Vol. 57, No. 3, pp. 836–848 (2016).

Table 2 Input fields of a ticket on ITS.

Field	Value Type	Description
ID[*1*2]	integer	monotonically increasing number.
created time[*1*2]	timestamp	created time.
updated time[*1*2]	timestamp	last updated time.
subject[*2]	text	a subject of an incident: suspicious malware infection, and so on.
description[*2]	long text	a description of an incident that SOC firstly reports.
priority[*2]	list	priority of this incident: low, medium, high, very high, extremely high.
a person in charge[*2]	list	a person in charge in CSIRT.
status[*2]	list	status of an incident defined in Table 1.
detection	list	detecting institute: commercial SOC, NII-SOCS, MEXT, police, user, CSIRT and other.
type	list	types of incidents: security, physical and contents.
threat	list	threat type: malware, phishing, XSS, defacing, unauthorized access, mail sending miss, DoS, account data breach.
malware name	text	a malware name.
malware type	list	types defined in STIX: adware, backdoor, bot, dropper, exploit-kit, key logger, ransomware, remote-access-trojan, resource-exploitation, rogues-security-software, rootkit, screen-capture, spyware, trojan, virus and warm.
external corresponding IP address	IP address	an IP address of a corresponding host.
internal global IP address	IP address	a global IP address of a suspicious host.
internal private IP address	IP address	a private IP address of a suspicious host.
MAC address	MAC address	a MAC address of a suspicious host
network category	list	a type of a network: education, research, secretariat, guest and other.
LAN type	list	types of media:, wireless or wired.
start time	timestamp	the time when malicious communication is started.
end time	timestamp	the end time when malicious communication is finished.
communication block	list	unapplied, firewall (IP address filtering), core switch (MAC address or IP address filtering), edge switch (port shutdown, MAC address or IP address filtering), wired or wireless LAN authentication (MAC address), wireless LAN controller (MAC address) and released.
host isolation	list	status of a suspicious host isolation: locating or isolating a host, recoverying from isolation and unapplied.
department	list	a department that the network belongs to.
division or section	text	a devision or section that the network belongs to.
user type	list	staff, student or other.
user ID	text	user ID of staff or student.
personal information	list	a suspicious host contains personal information or not.
encryption	list	confidential data is encrypted or not.
data breach	list	data breach is possible or impossible.
SOC ticket number	text	SOC ticket number.
SOC ticket status	text	open, SOC investigating, wating for SOC response, CSIRT investigating, closed, and so on.
SOC notification time	timestamp	the time when a SOC notifies.
OS and version	text	OS and its version of a suspicious host.
security software	text	security software name and version.
personal information types and amount	long text	personal information types such as phone number, name, e-mail address and etc. and theirs amount.
communication log investigation	list	done or not.
identifying infection source	list	done or not.
specimen collection	list	done or not.
static analysis	list	done or not.
dynamic analysis	list	done or not.
obtaining file list	list	done or not.
obtaining start up list	list	done or not.
obtaining task list	list	done or not.
obtaining task scheduling list	list	done or not.
obtaining registry	list	done or not.
forensic	list	done, deleted or not.
countermeasures to prevent recurrence	long text	a description of a countermeasures.
abstract	long text	a brief description of an incident to explain to board members.

Table 3 Visibility and permissions of input fields of a ticket on ITS.

Field	Identification (CSIRT)	Awaiting identification (department)	Data breach investigation (CSIRT)	Data breach investigation finished (CSIRT)	Awaiting final report (department)	Awaiting OS re-installation (student)	Abnormal (done)	Finished) (done)
ID *	*	*	*	*	*	*	*	*
created time *	*	*	*	*	*	*	*	*
updated time *	*	*	*	*	*	*	*	*
subject *	*	*	*	*	*	*	*	*
description *	*	*	*	*	*	*	*	*
priority *	*	*	*	*	*	*	*	*
a person in charge		*	*	*	*	*	*	*
detection	-	-	-	-	-	-		-
type	-	-	-	-	-	-		-
threat								
malware name								
malware type								
external corresponding								
IP address								
internal global IP address								
internal private IP address		*	*	*				*
MAC address		*	*	*				*
network category	*	*	*	*	*			*
LAN type	*	*	*	*	*			*
start time								
end time								
communication block	*	*	*	*	*			*
host isolation		*	*	*	*			*
department	*	*	*	*	*			*
division or section		*	*	*	*			*
user type		*	*	*	*			*
user ID		*	*	*	*			*
personal information		*	*	*	*		*	*
encryption		*	*	*	*		*	*
data breach			*	*	*		*	*
SOC ticket number								
SOC ticket status								
SOC notification time								
OS and version			*	*				
security software			*	*				
personal information types and amount	-	-	*	*	-	-	-	-
communication log	-	-		*	-	-	-	-
investigation				*	-	-	-	-
identifying infection source	-	-		*	-	-	-	-
specimen collection	-	-		*	-	-	-	-
static analysis	-	-		*	-	-	-	-
dynamic analysis	-	-		*	-	-	-	-
obtaining file list	-	-		*	-	-	-	-
obtaining start up list	-	-		*	-	-	-	-
obtaining task list	-	-		*	-	-	-	-
obtaining task scheduling list	-	-		*	-	-	-	-
obtaining registry	-	-		*	-	-	-	-
forensic	-	-		*	-	-	-	-
countermeasures to prevent recurrence	-	-						*
abstract	-	-					*	*