Revealing Abuses of Channel Assignment Protocols in Multi-channel Wireless Networks: An Investigation Logic Approach

Qijun Gu[1], Kyle Jones[1], Wanyu Zang[2], Meng Yu[2], and Peng Liu[3]

[1] Texas State University, San Marcos, TX 78666, USA
[2] Virginia Commonwealth University, Richmond, VA 23284, USA
[3] Pennsylvania State University, University Park, PA 16802, USA

Abstract. This paper presents a novel specification-based investigation logic and applies it to tackle abuse of channel assignment protocols in multi-channel wireless networks. The investigation logic looks into malicious operations that violate the specification of channel assignment protocols. With logged operations, it reconstructs the process of channel assignment as an information flow that captures essential dependency relations among protocol-specific channel assignment operations. Then, it derives and applies reasoning rules to conduct consistency check over the logged operations and identify the source of abuse where the logged operations are inconsistent. Through simulation, the proposed investigation logic presents desired quality with zero false negative rate and very low false positive rate.

1 Introduction

Recent studies [5,15] have shown that using multiple non-interfering channels appropriately can significantly reduce interference among neighboring nodes and improve the overall capacity of a wireless network. Various channel assignment (CA) protocols have been developed for multi-channel wireless networks [16,2,28,8]. However, CA protocols are vulnerable to various new attacks [6,21,11,30], in addition to existing security threats in single channel wireless networks. These *new* threats can effectively ruin the "good will" of the CA protocols, reduce the network throughput, and downgrade the quality of network access. Examining these attacks, we found that it is very easy for attackers to abuse CA protocols to cause channel conflicts in multi-channel networks. Attackers can claim wrong, no-existing or incomplete channel information. Attackers can conduct mis-operations to provide different channel information to different nodes. All such abuses of CA protocols force or misguide the victims to change channels so that the network capacity is worse utilized or even the network access is disabled.

Unfortunately, securing multi-channel wireless networks is not as simple as adapting the existing defenses in single channel wireless networks. Authentication-based defense approaches [31,18] have been studied in multi-channel networks.

S. Foresti, M. Yung, and F. Martinelli (Eds.): ESORICS 2012, LNCS 7459, pp. 289–306, 2012.

They ensure the integrity of the node identity and the information carried in channel assignment packets. But, their defense capacity is limited to non-authentic information. For example, they cannot stop compromised nodes from providing authentic information with compromised credentials, or stop attackers from providing in-consistent yet authentic information to victims. Intrusion detection frameworks [25,9] have also been proposed in multi-channel networks. These frameworks still have their stands on the assumption that intrusive actions can be observed as in single channel wireless networks. Nevertheless, CA protocols intend to have nodes work on different channels and thus channel assignment operations (including attackers') on a particular channel may not be observed at all when intrusion detection agents are "cruising" on other channels.

Knowing the difficulties of securing multi-channel wireless networks, we explore new means of investigating abuses of CA protocols based on *evidence* (i.e. logged channel assignment operations). We propose a specification-based investigation logic to reveal abuses of CA protocols. Our proposal is built upon the fact that all good nodes will follow CA protocols in channel assignment. Any abuse of CA protocols must inherently have some operations or information that are either contradictory to the protocols or inconsistent inside the abuse itself. A key difference between the abuse investigation and existing intrusion detection in wireless networks is that it exploits the dependency among evidence. The investigation gathers both true and phony evidence in network, builds dependency among evidence using a generalized model of protocol specifications, and locates the inconsistency among evidence. The investigation helps the network administrators to track down the sources of abuse that inevitably results in contradictory evidence.

The investigation is challenging in several aspects though. First, CA protocols vary in their specifications. The operations and the causal relations among operations are specified by protocols, while the operations performed by nodes are stochastic and determined by the channel condition at the time being. Hence, it is neither simple to reconstruct the possible attack scenes nor straightforward to identify the relations among evidence. Second, nodes have only localized views on their channels, and may miss information due to packet loss. The evidence they can provide is always limited to what they can observe on their channels and in their neighborhood. The evidence provided by nodes may not directly include the information about attackers. The limited information is also an obstacle to justifying the operations of good nodes. Third, attackers can compromise deployed nodes to obtain credentials and provide seemingly authentic channel information. Evidence built upon such authentic but wrong information may deceive investigators. Attackers can "lie" in investigation by providing phony evidence that are not generated from their attack operations. Meanwhile, it is hard to identify phony evidence as attackers can claim that the phony evidence is established on information and operations missed by other nodes.

The proposed investigation logic contributes to the security of multi-channel wireless networks in the following aspects: **(1) An investigation model that captures essential dependency relations among protocol-specific**

channel assignment operations. CA protocols are complex in nature for managing multiple channels and multiple radio interfaces. We identify three types of dependency among operations. We show that such relations would enable one to link together channel assignment operations among nodes and rebuild the channel assignment process with the use of standard operation logs. Unlike static logic-based approaches that analyze security protocols or security configurations, the investigation is adapted to the dynamic evidence derived from channel assignment traffic and operations. **(2) Investigation reasoning rules that identify the source of abuse where the evidence is inconsistent.** Based on stochastic profiles of channel assignment operations, we identify two types of inconsistency that need different investigation mechanisms. Various investigation logic rules are derived to detect possible abuses with each category of evidence, and algorithms are developed to apply the investigation rules to locate inconsistency among evidence and identify the suspects involved with the suspicious evidence. Although the investigation does not directly locate the exact attacker, it enables the investigator to quickly narrow down the investigation onto a small range of suspicious operations and reduce investigation efforts and costs in orders of magnitude.

The rest of the paper is organized as follows. Section 2 states the assumptions, the threat model, and the investigation goal. Then, we propose the investigation model in Section 3, and the investigation reasoning in Section 4. Evaluation is given in Section 5. Related work is discussed in Section 6. Finally, we summarize our work in Section 7.

2 Preliminary

2.1 Network Model and Assumptions

Similar to many studies for multi-channel wireless networks, we assume a wireless network has multiple *non-interfering communication channels* (channels for short thereafter). The number of channels is specified by communication protocols. For example, IEEE802.11A has 13 channels, and IEEE802.11B has 3. To communicate with neighbors on different channels, a node is equipped with multiple *radio interfaces*. A node obtains its channels through the interaction with their neighbors following a CA protocol. Then, the node switches its radio interfaces onto the obtained channels. The gateway nodes of a network are the nodes statically connected to the wired network. They have pre-assigned channels. All other nodes obtain their channels after joining the network. We also assume good nodes in the network do not have the knowledge of the whole network. Each node can only observe network traffic on its working channels in its vicinity. If an attacker is communicating with a victim on the victim's channel while no other nodes are working on the victim's channel, no nodes other than the victim and the attacker can know the attacker's action.

2.2 Threat Model and Assumptions

For security, we assume proper authentication schemes have been applied. Meanwhile, we assume attackers can disseminate wrong but authentic channel information. For example, attackers can capture and compromise some deployed nodes to obtain credentials. We also assume attackers can lie during investigation. They can create phony evidence from wrong or non-existing operations, as long as they can authenticate the evidence. The only trusted nodes in network are the gateways, which are usually better secured than other nodes.

Multi-channel wireless networks are vulnerable to various attacks. Based on the common characteristics of attacks, we define two types of attacks that perform malicious actions in channel assignment to deceive the perception of good nodes on channels and make good nodes to switch to channels that are only "perceived" better. The formulation and implication of the two types of attacks will be discussed in detail in Section 3.4.

Definition 1. Type-I attack*: a malicious operation that violates the specification of a protocol and a good node does not perform.*

Definition 2. Type-II attack*: a malicious operation that follows the specification of a protocol but a good node may only perform with a low probability.*

2.3 Investigation Goal

Based on the above observations of the known threats, we define *good operations* as the operations that follow the specification of CA protocols, including using verifiable channel information, taking specified actions, and transmitting specified packets. Correspondingly, *malicious operations* do not use verifiable channel information, do not take specified actions, or do not transmit specified packets.

The objective of our investigation is not to identify which node or which operation triggers other nodes to change channels. Rather, *our investigation goal is to identify the malicious operations that violate the specification of CA protocols.* Because such malicious operations lead to the aforementioned attacks, our investigation can deter attackers from launching such attacks.

As malicious operations violate the specification of CA protocols, they must leave traces that in fact mismatch the actual status and audit data in good nodes and are not justifiable by attacker-provided information. Hence, if we could detect the mismatch in the traces that are supposed to hold consistent information, we could identify the malicious actions. Based on this rationale, we propose an investigation logic that can effectively investigate a possible attack.

3 Investigation Model

The foundation of our investigation is built upon the specification of CA protocols and the operations of nodes in channel assignment. This section presents the notations and the modeling used in the investigation logic.

3.1 Notations

To investigate the past operations of nodes, we first model the logged operations with a triple $\langle \mathbb{A}, \mathbb{M}, \mathbb{K} \rangle$ based on an earlier audit model in [13]. In this model, $\mathbb{A}$ is a set of nodes, $\mathbb{M}$ is a set of messages, and $\mathbb{K}$ is a set of actions. The investigation model captures the relation among the operations of nodes in channel with the three components:

Nodes could be honest (good) or dishonest (malicious). Gateways of a multi-channel network are assumed to be honest nodes in this paper, as they are usually better protected physically and logically than other nodes.

Messages include internally maintained information, externally transmitted packets, and so on. They are the information used in channel assignment, and are recorded for investigation. Fields of a message m is protocol-specific, and could include identities, assigned channels, traffic quality metrics, paths, sequence numbers (or time marks), neighbors' channel and traffic information, and so on.

Actions are operations performed by nodes in $\mathbb{A}$ over messages in $\mathbb{M}$. An action k is thus defined as $k = \{m \overset{a:\rho}{\rightarrow} r\}$, where $a \in \mathbb{A}$ is the acting node, $m \in \mathbb{M}$ is the input message, $r \in \mathbb{M}$ is the result message, and ρ is the operation. The action k means that a performs ρ on the input m and gets the result r. k is a *fixed* action if r is fixed given m. Otherwise, if r is random given m, k is a *random* action.

Actions can be *concatenated or combined*. Given two actions $k_1 = \{m_1 \overset{a:A}{\rightarrow} r_1\}$ and $k_2 = \{m_2 \overset{b:B}{\rightarrow} r_2\}$, the two actions are *concatenated* as $k_1 \| k_2$, if the result of k_1 is an input of k_2, i.e. $r_1 \subseteq m_2$. The two actions are *combined* as $k_1 \cup k_2$, if the results of k_1 and k_2 are the input of another action $k_3 = \{m_3 \overset{c:C}{\rightarrow} r_3\}$, i.e. $r_1 \cup r_2 \subseteq m_3$.

Actions are also categorized as external and internal. An action is *external* if either the input or the result of the action is a transmitted packet. For example, two external actions are involved when a sends m_a to b: $k_a = \{m_a \overset{a:S}{\rightarrow} p_{ab}\}$ and $k_b = \{p_{ab} \overset{b:R}{\rightarrow} m_a\}$, where p_{ab} is a transmitted packet containing m_a, S means sending, and R means receiving. The two external actions are concatenated as $k_a \| k_b$, and simplified as $k_{ab} = k_a \| k_b = \{m_a \overset{a:S}{\rightarrow} p_{ab}\} \| \{p_{ab} \overset{b:R}{\rightarrow} m_a\} = \{m_a \overset{ab:SR}{\rightarrow} m_a\}$. An action is *internal* if both the input and the result of the action are ϕ or internally kept information by the node. For example, the internal action for a's periodic timer is $k_a = \{m_a \overset{a:\sigma}{\rightarrow} m_a\}$, where σ is timer and m_a is the information kept in a. The timer action is simplified as σ. Another example of internal action is that a updates its neighbor information m_a with a received update m_b from its neighbor b. The internal action is $\{(m_b \cup m_a) \overset{a:U}{\rightarrow} m'_a\}$, where U means updating and m'_a is the neighbor information after update.

3.2 Dependency among Actions

In a sequence of actions, we observe that the input of an action is usually the output of an earlier action. Thereby, we can concatenate an action with an earlier

action. The concatenation of the two actions represents the dependency between the two actions as defined below. We show three types of dependent actions that capture all possible dependency among actions in processes.

Definition 3. *Action x depends on an earlier action y if they can be concatenated as $y\|x$. x is called a* dependent action *and y is the* dependency *of x.*

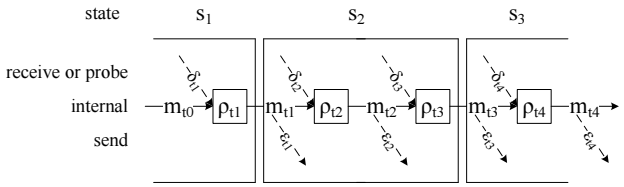
- **Dependent Internal Actions.** After a node obtains a channel, it needs to conduct periodic operations to keep update with its neighbors and select better channels. These operations form a sequence of internal actions inside each individual node. Three types of internal actions are involved: periodic timer action, update action, and channel selection action. The three types of internal actions are dependent internal actions and are concatenated with earlier internal or external actions.
- **Dependent External Actions.** The operations of individual nodes are not isolated. A node needs to send its latest channel to other nodes so that they can keep up-to-date information with each other. The sending and receiving actions involved in the information exchange are concatenated as dependent external actions. This type of dependent actions establish the connections between the operations of different nodes when they communicate to exchange information.
- **Leaf Actions.** CA protocols require nodes to conduct the channel assessment operation to assess the traffic condition of their channels. The assessment actions do not take any results from other actions as input. Hence, although they are internal actions, they do not depend on any earlier actions. The leaf actions cannot be concatenated to any earlier actions, but are always concatenated to later dependent internal actions.

3.3 Process Model

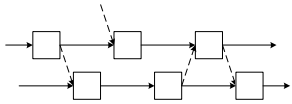
Although CA protocols vary, the operations of good nodes can always be represented as a process. A **process** is a triple $\langle \mathbb{S}, s_0, \Rightarrow \rangle$, where $\mathbb{S}$ is a set of states, s_0 is the starting state, and $\Rightarrow$ is a sequence of concatenated actions. In channel assignment, states are aligned with the assigned channels. For example, s_ϕ is the state without assigned channel and s_c is the state with an assigned channel c. If a node changes from s_1 to s_2 after action k, the process is $s_1 \overset{k}{\Rightarrow} s_2$.

As each node maintains the information of its neighbors and channel conditions, dependent internal actions in $\Rightarrow$ are concatenated to form an information flow of the process as illustrated in Figure 1(a). Let m_t be the information a node maintains and ρ_t be the internal operation at time t. The dependent internal actions are concatenated in the order of updating m_t.

The information flow of a node's process starts with $m_{t_0} = \phi$ in state s_{t_0} at t_0. Over a sequence of time points t_0, t_1, t_2, ..., the node conducts a sequence of operations ρ_{t_1}, ρ_{t_2}, ... Each operation ρ_{t_j} takes $m_{t_{j-1}}$ as input and outputs m_{t_j} to operation $\rho_{t_{j+1}}$. The information flow is segmented with different states. The last operation in each state segment is what the node performs to select new channels.



(a) An Example of Single Process



(b) An Example of Multiple Processes

Fig. 1. Process Model

ρ_{t_j} also take δ_{t_j} as input, which is a union of new information of channels received between t_{j-1} and t_j. $\delta_{t_j} = \phi$ indicates that the node didn't perceive any change in channels. Otherwise, $\delta_{t_j} = \cup m_{*a} \cup q_c \cup f_l$, which includes the messages received from neighbors and the results of channel and link assessments. Hence, δ_{t_j} is the results of combined dependent external actions and leaf actions. The output m_{t_j} of operation ρ_{t_j} may also be sent out as $\epsilon_{t_j} \subseteq m_{t_j}$ to neighbors through the sending action.

Processes of multiple nodes are further modeled as interconnected chains via δ_t and ϵ_t. Figure 1(b) shows two processes of two neighbor nodes. As they exchange information, their processes are interconnected. The connections of chains represent the dependency of dependent external actions.

The process model captures the dependency among actions of good nodes and represents the protocol specification as an information flow of processing m_t. Hence, abuse of CA protocols is in fact an attempt to change or break the information flow. The hypothesis of our investigation is that no attack presents to affect the channel assignment if the information flow can be verified.

3.4 Statistic Profile of Actions

As outlined in [12], the actions in the process are stochastic. $\mathbb{K}$ (the set of actions as discussed in the above investigation model) is a mixture of a good action set $\mathbb{G}$ and a malicious action set $\mathbb{B}$, i.e. $\mathbb{K} = \mathbb{G} \cup \mathbb{B}$. Following Definitions 1 and 2, the set of type-I attacks is $\mathbb{B}_{\mathbb{I}} = \overline{\mathbb{G}} \cap \mathbb{B}$ and the set of type-II attacks is $\mathbb{B}_{\mathbb{II}} = \mathbb{G} \cap \mathbb{B}$.

Let $G(t) \in \mathbb{G}$ be a possible good action at t, and let $B(t) \in \mathbb{B}$ be a possible malicious action at t. Let $k(t)$ be the logged action at time t. Then, $k(t) \in \mathbb{K}$ is either $G(t)$ or $B(t)$. Following the notations of [12], we define the following probabilities for good actions and malicious actions.

- $g(k(t)) = Pr\{G(t) = k(t)\}$ is the probability that the good action at t is $k(t)$.

– $b(k(t)) = Pr\{B(t) = k(t)\}$ is the probability that the malicious action at t is $k(t)$.

Let $k_g(t) \in \mathbb{G}$ be the logged good action at t, $k_I(t) \in \mathbb{B}_\mathbb{I}$ be the logged type-I attack action at t, $k_{II}(t) \in \mathbb{B}_\mathbb{II}$ be the logged type-II attack action at t. We have the following statistic profiles of good actions and malicious actions.

1. Statistic profile of good actions:
   – $g(k_g(t)) = 1$ for fixed good actions, because a fixed action always has only one specific result at t.
   – $0 < g(k_g(t)) < 1$ for random good actions, because a random action may end with different results at t.
   – $g(k_I(t)) = 0$ for good actions, because $k_I(t) \notin \mathbb{G}$ due to $\mathbb{B}_\mathbb{I} \cap \mathbb{G} = \phi$.
   – $0 < g(k_{II}(t)) < 1$ for random good actions, because $k_{II}(t) \in \mathbb{B}_\mathbb{II}$ and $\mathbb{B}_\mathbb{II} \subset \mathbb{G}$.
2. Statistic profile of malicious actions:
   – $b(k_g(t)) = 0$ for type-I attack actions, because $k_g(t) \notin \mathbb{B}_\mathbb{I}$ due to $\mathbb{B}_\mathbb{I} \cap \mathbb{G} = \phi$.
   – $0 < b(k_I(t)) \leq 1$ for type-I attack actions.
   – $0 < b(k_{II}(t)) \leq 1$ for type-II attack actions.

As discussed in the investigation models, actions can be classified as fixed and random. The dependent external actions and the dependent internal actions (excluding the last internal action at the end of each state in a process) are usually fixed actions, because their results are fixed given their inputs. The dependent internal action at the end of each state in a process is the channel selection action. It is fixed if the CA algorithm of the protocol is deterministic, e.g. the algorithm always selects the least used channel. Otherwise, it is random if the CA algorithm is non-deterministic. The leaf actions are usually random actions, because their results are the assessment of channel condition that always includes noise and errors.

4 Investigation Reasoning

For investigation, the actions are recorded by nodes and later retrieved to reconstruct the processes by investigators. A log entry includes the input, the result, the operation, and the sequence number of the action. Because the abuse actions violate the protocol specification, the log entries corresponding to the abuse will not fit into the information flow. The key of investigation is to locate such problematic log entries by conducting consistency checks on $\mathbb{M}$ and $\mathbb{K}$ following the information flow.

4.1 Consistency

As discussed in [12], the effectiveness of investigation is limited by the disparity of $b(k(t))$ and $g(k(t))$ for $k(t) \in \mathbb{B}_\mathbb{I} \cup \mathbb{B}_\mathbb{II}$. When the two probabilities are very different, the investigation can detect malicious actions with high confidence. When

the two probabilities are highly similar, the investigation cannot distinguish between good and malicious actions. In other words, when malicious actions follow the same statistic profile of good actions, attackers in fact behave as good nodes and thus do not threat channel assignment.

The investigation measures the disparity of the two probabilities as $d(k(t)) = |\log b(k(t)) - \log g(k(t))|$. Then, the overall disparity of all past actions is given in Eq.(1), where D_I is the disparity between type-I attack actions and good actions and D_{II} is the disparity between type-II attack actions and good actions. Note that actions in $\mathbb{G} \cap \overline{\mathbb{B}_{\mathbb{III}}}$ will not be performed by attackers, and thus $D_{\mathbb{G}\cap\overline{\mathbb{B}_{\mathbb{III}}}} = 0$.

$$D = \sum_t d(k(t)) = \sum_t |\log b(k(t)) - \log g(k(t))| = D_I + D_{II} \tag{1}$$

$D_I = \sum_t d(k_I(t))$. Because $g(k_I(t)) = 0$, $D_I = \sum_t |\log b(k_I(t)) - \log 0|$. $D_{II} = \sum_t d(k_{II}(t)) = \sum_t |\log b(k_{II}(t)) - \log g(k_{II}(t))|$. Because the goal of a good random action is to improve the channel usage, a good node will less likely to perform a type-II attack action that results in worse channel usage, and thus $b(k_{II}(t)) \geq g(k_{II}(t))$. Thereby, $D_{II} = \sum_t (\log b(k_{II}(t)) - \log g(k_{II}(t))) > 0$, and $\sum_t \log b(k_{II}(t)) > \sum_t \log g(k_{II}(t))$.

The above disparity analysis shows quantitatively and theoretically how attack actions are inconsistent with good actions. However, in practice, we do not know exactly $\mathbb{B}_{\mathbb{I}}$ and $\mathbb{B}_{\mathbb{III}}$, and their statistic profiles $b(k_I(t))$ and $b(k_{II}(t))$. Therefore, in the following, we propose reasoning rules to conduct consistency checks for investigating the two types of attacks based on the analysis of D_I and D_{II}.

4.2 Consistency Check on Type-I Attack

The disparity D_I shows that the inconsistency of a type-I attack action can be detected when a logged action should not occur but present in log. Hence, the consistency check on type-I attack is equivalent to the check on whether or not a logged action can fit into the processes built upon the log. The consistency check on type-I attack includes two steps: (i) rebuild processes from log and (ii) check dependency among actions.

Rebuild Processes. As discussed in Section 3.3, the process of a node is made of dependent and leaf actions that form an information flow. Upon investigation, the procedure in Algorithm 1 is executed to rebuild the processes for all nodes. If the information flow is not complete due to a missing earlier dependency action, inconsistency is detected. Lemma 1 states that such an inconsistent situation is caused by a type-I attack or a lie.

Lemma 1. *For a dependent action, if the search of an earlier dependency action in Algorithm 1 fails, (i) at least one of the two actions is a type-I attack action, or (ii) at least one of the owners of the two actions lies.*

Algorithm 1. Procedure of rebuilding processes

```
1: Collect log entries over an investigation period
2: for each node do
3:     Sort log entries according to their sequence numbers
4:     for each log entry do
5:         if the action is a dependent internal action then
6:             Search earlier dependent internal actions following its inputs
7:             Report inconsistency on the current action if search fails
8:             Add edges with the earlier actions that can be concatenated
9:         end if
10:        if the action is a sending action then
11:            Search earlier dependent internal action
12:            Report inconsistency on the current action if search fails
13:            Add an edge from the earlier internal action to the current action
14:        end if
15:        if the action is a receiving action then
16:            Search the sending action from the sending node
17:            Report inconsistency on the current action if search fails
18:            Add an edge from the sending action to the current action.
19:        end if
20:    end for
21: end for
```

Dependency Check on Dependent Actions. Although the information flow presents the dependency relations among logged actions, the actions may contain false information. For example, attackers provide bogus log entries in order to form a complete information flow in a rebuilt process. The bogus log entries will then cause inconsistency on the dependency among actions. To inspect dependent actions, investigation needs to check the *operation consistency* and the *dependency consistency* of the actions.

Definition 4. Operation consistency*: A logged action is operation consistent if the logged result matches the result of the action given the logged input.*

Definition 5. Dependency consistency*: Assume in the information flow of a process, a logged dependent action has edges with some earlier dependency actions. The dependent action is dependency consistent if the logged input of the action is the union of the logged results of the earlier dependency actions.*

To evade investigation, attacking nodes must lie to ensure that the logged actions can pass the two dependency checks. Because attacking nodes can only provide bogus actions in their own processes, they must provide actions with self-sustained operation consistency and dependency consistency in the processes that can prove each other. Lemma 2 and Lemma 3 can be established to ensure that no type-I attack can enable attackers to evade the investigation by lying on dependencies.

Lemma 2. *An attacking node cannot lie with a set of bogus type-I attack actions with self-sustained consistency to evade investigation.*

Lemma 3. *A set of colluding attacking nodes cannot lie with a set of bogus type-I attack actions with self-sustained consistency to evade investigation.*

Theorem 1. *A type-I attack is detectable with 100% guarantee.*

4.3 Consistency Check on Type-II Attack

The disparity D_{II} shows that the inconsistency of a type-II attack action can be detected when a logged action should occur less but present too often in log. Although we do not know the exact statistic profile of type-II attack $b(k_{II}(t))$, we can establish Lemma 4 stating that $\sum_t \log b(k_{II}(t))$ is an approximate of the entropy of type-II attack actions. Then, we can further establish Theorem 2 that type-II attack actions are detectable.

Lemma 4. *Let T be the number of type-II attack actions in log.* $\lim_{T\to\infty} -\sum_t \log b(k_{II}(t)) = H_T(\mathbb{B}_{\mathbb{II}})$*, where $H_T(\mathbb{B}_{\mathbb{II}})$ is the entropy of the T type-II attack actions.*

Theorem 2. *Given $H_T(\mathbb{B}_{\mathbb{II}})$, type-II attack actions are detectable with their cumulative information content as good random actions, i.e.* $-\lim_{T\to\infty} \sum_t \log g(k_{II}(t)) > H_T(\mathbb{B}_{\mathbb{II}})$.

Accordingly, the consistency check on a type-II attack action is to compare the information content of a logged action as a random good action with the entropy of the action. Since we do not have $H_T(\mathbb{B}_{\mathbb{II}})$, the detectability only works in theory. In practice, we develop heuristic consistency check on two types of random good actions: (i) leaf actions and (ii) channel selection actions.

Cross-Verification on Leaf Actions. Leaf actions are stochastic in that they are used to sample time-variant channel conditions. In particular, we see two types of leaf actions for all the protocols we have studied. Our investigation logic cross-verifies them with other log entries based on some heuristics.

- The *link failure action* occurs when a node changes its channel or leaves the network which results in the link failure. An attacking node may exploit this action to claim the change in its channel usage. The heuristic of our investigation is to compare the channel of the link before and after the failure. A logged link failure action is determined as a type-II attack action, if the channel of the link is not changed.
- The *channel assessment action* measures the channel usage condition. To verify a logged channel assessment action, our investigation compares the assessed channel condition with the channel usage information in the log entries of its neighbor nodes. A logged channel assessment action is

Algorithm 2. Procedure of Consistency Check

```
Start from a logged channel selection action.
Initiate an empty set INV
Add the starting action to INV
repeat
    Follow the information flow edges of the logged actions.
    Add the dependency actions and the edges to INV
until no more dependency actions in the investigation window
for each dependent action in INV do
    Check the operation consistency of the action
    Check the dependency consistency of the action with its dependency actions
    if the action does not satisfy the consistency then
        Report the action, its dependency action, and the owners of these logged actions
    end if
end for
for each leaf action and channel selection action in INV do
    Verify the action according to its type
    if the log entry does not satisfy the consistency then
        Report the log entry, the log entries used for verification, and the owners of these log entries as suspicious
    end if
end for
```

determined as a type-II attack action if the difference of the channel usage falls outside a reasonable statistical range.

Entropy Check on Channel Selection Actions. In most CA protocols, the channel selection action is a random action. When a node finds a set of better channels, the node can select a new channel or stay at its current channel with some probabilities. Following Theorem 2, we substitute $H_T(\mathbb{B}_{\mathrm{II}})$ with $H_T(\mathbb{G})$. This heuristic is based on (i) the simplification that $g(k_g(t))$ and $b(k_{II}(t)$ are symmetrically skewed, because a less performed good action will be more performed by attackers and vise verse; (ii) the information content of a less performed good action is larger than the entropy.

4.4 Procedure of Consistency Check

Algorithm 2 shows the procedure of applying the two consistency checks after the processes are rebuilt. It first identifies the log entries needed for investigation and extracts them out to form a reasoning graph for investigation. Then, it verifies the consistency among these log entries. It reports both the inconsistent log entries that cannot be justified and the supporting log entries that are directly involved in the verification of the inconsistent log entry.

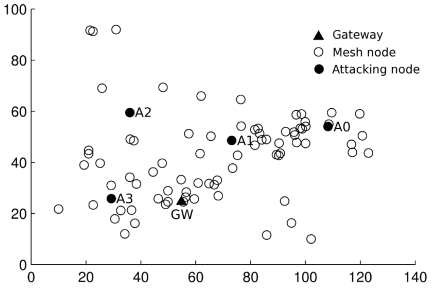


Fig. 2. Topology of Roofnet

5 Evaluation

5.1 Simulation Settings

The investigation logic has been implemented in the multi-channel wireless network simulation framework [14] in OMNET++/INET [1]. The investigation logic has been tested with two CA protocols MCR [15] and ROMA [8]. We use the topology of Roofnet to build the network for simulation. Roofnet is a testbed of wireless mesh network [3]. The coordinates of the mesh nodes in Roofnet are published online.

Figure 2 shows the reconstruction of the topology with 90 deployed good nodes in simulation. Each node is equipped with two radio interfaces. According to the specifications, 13 channels are used for IEEE802.11A and 3 channels for IEEE802.11B. The channel capacity is set to $2Mbps$ per channel. Good nodes are given random starting time points within the first 30 seconds, from which they start searching channels to join the network. Once they get channels, they broadcast their channel and load information every 10 seconds following the CA protocols, and send traffic to the gateway.

We also deploy 4 attacking nodes to inject false CA packets following the known attack techniques [21,11,30]. For type-I attack, the attacking nodes inject non-existing channel usage packets, and send different channel usage packets to different nodes. For type-II attack, the attacking nodes notify other nodes of newly assigned channels, which are less possibly selected by good nodes.

In simulation, we vary the traffic volume of good nodes so that the load of aggregated traffic at the gateway ranges from 10% to 80% of channel capacity at the gateway, i.e. $200Kbps$ to $1600Kbps$. We fixed the communication range of each router but adjusted the distance among routers proportionally to control the node degree, i.e. the number of neighbors per router. The average node degrees are set to 9, 14, and 27 for simulation with different network density. Each experiment ran 200 seconds and all data points are the averages over 10 repeats.

5.2 Quality of Investigation

The quality of investigation logic is measured by the false positive (*FP*) rate (the percentage of good actions that are determined as malicious) and the false negative (*FN*) rate (the percentage of attack actions that are determined as good). Note that the two rates do not indicate the quality of final judgment on good and malicious nodes. Our investigation goal is to mark suspicious attack actions so that the intrusion detection can zero onto the nodes involved with the marked actions.

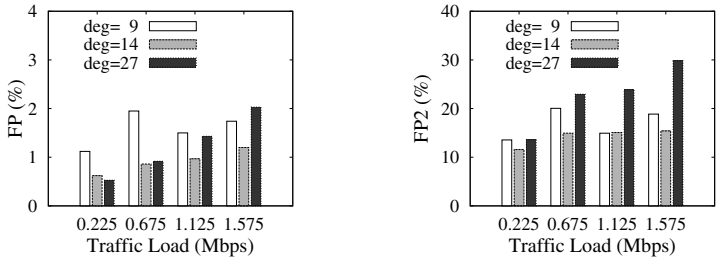
In simulation, the FN rates achieved by the investigation logic are 0 for two reasons. First, the type-I attack actions do not follow the specification. They cannot evade detection following Theorem 1. Second, the type-II attack actions in simulation always have lower probabilities than the good actions performed by good nodes. They can thus be detected using Theorem 2. This result indicates that the investigation logic will not miss any attack actions.

Figures 3(a) and 4(a) show the overall FP rates with MCR and ROMA. The overall FP rates are below 2% for MCR and below 1% for ROMA. With further examination, we find that the FP rate of type-I attack (i.e. a good action is determined as a type-I attack) is zero. Hence, the FP actions are in fact determined as type-II attack. We calculated the FP rates of type-II attack with only the good actions, which could be mis-classified, as illustrated in Figures 3(b) and 4(b). Because the number of such good actions is only a small portion (4% to 10%) of all good actions, the overall FP rates are low even though the FP rates of type-II attack could be as high as 30%.

With further examination of the FPs of type-II attack, we find the actions and the reasons that lead to the FPs. First, the leaf actions of channel assessment and link failure could be determined as type-II attack. The leaf action of channel assessment reports the channel condition perceived by nodes. Such perception can hardly be accurate and uniform among different nodes. A good node may perceive a channel condition that appears abnormal to other nodes and thus the leaf action of this channel assessment will be determined as a type-II attack action. Similar reason applies to the leaf action of link failure when a link fails due to temporary communication problems.

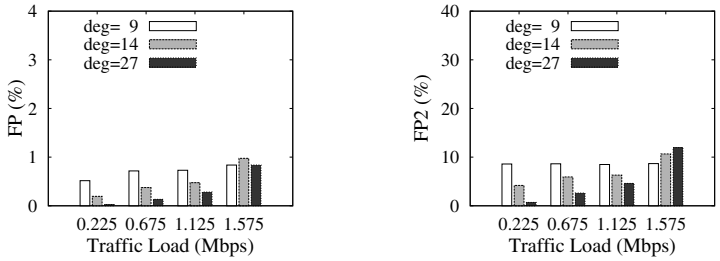
The other action that could be determined as type-II attack is the channel selection action. For MCR, when a node finds a better channel, it may switch to that channel at a probability of 60%. In contrast, for ROMA, the node will switch to the better channel. Hence, for MCR, when the node stays on its current channel, the channel selection action is more likely determined as type-II attack. It is also the reason that MCR's FP rate of type-II attack is significantly larger than ROMA's.

Another observation of Figures 3 and 4 is that the traffic load may influence the FP rate only when the network is dense. When the node degree is 27 (i.e. a node has 27 neighbors in average), the FP rates grow as the traffic load increases. This is because the network is less stable with the growing traffic load in a dense network. The instability of network contributes more into the uncertainty



(a) Overall False Positive (b) FP of Type-II Attack

Fig. 3. False Positive with MCR



(a) Overall False Positive (b) FP of Type-II Attack

Fig. 4. False Positive with ROMA

in channel assessment and link failure, which results in more wrong perceptions of channel and link conditions in good nodes.

6 Related Works

6.1 Logic Methodology

The taxonomy of secure protocol analysis methods includes two methodologies to handle not-so-sophisticated attacks, which are Protocol Logic and Model Checking [7,20,19], and two methodologies to handle sophisticated attacks, which are Poly-ti e calculus [17] and Symbolic methods (MSR) [4]. We choose to adopt the logic methodology due to its simplicity and efficiency.

BAN logic has been used for describing security protocols. It formulates the security protocol analysis problem as feeding "formal protocol" and "intruder model" as two inputs into an analysis tool, which finds errors. Compared with this problem formulation, a major difference is that our investigation logic does not require an explicit intruder model. Instead, our investigation logic is *specification-based*, that is, it is based on specification of expected protocol behavior. In this way, our investigation logic has the potential capability to identify unknown attacks.

Datalog is another logic-based approach used in security systems. The MulVAL [23,22] analyzes the vulnerabilities of a system. It adopts Datalog as the modeling language for system elements, which are configuration, privilege, reasoning rules, and so on, in analysis. The Datalog-based logic approaches are different from our investigation logic in that our objective is to track down the sources of protocol abuses instead of finding vulnerabilities in protocols or system configurations. In addition, they build reasoning graphs based on static information, while our reasoning graph adapts to evidence produced by dynamic channel assignment traffic and operations.

6.2 Node Compromise Detection

A number of node compromise detection schemes have been studied in wireless sensor and ad hoc network [29,24,27,26]. A compromised node may behave less trustable after compromise as it will conduct malicious operations to serve attackers. Hence, researchers proposed reputation-based trust management schemes [10,27], in which the reputation of each node is evaluated by other nodes or the system in accordance with its activities. A node with low reputation will be considered as untrustable. Various approaches can be used to evaluate the trust of a node, such as Bayesian-based [10] and entropy-based [27].

Attackers may insert malicious code into compromised nodes to conduct attack operations. Remote software-attestation schemes [29,24] were proposed with which the base station or the neighbors of a suspect node can detect the compromise via checking the change of the code image inside the node. In our study, attackers do not need to run extra malicious code in compromised node. They mainly need the credentials from the compromised nodes to abuse CA protocols in an authentic manner.

Our work is different from all the above works in that we look into the inherent properties of attacks rather than detecting if a node has been changed by attackers. We base our investigation logic on the fact that abusing CA protocols will violate the specification of the protocols, whether or not the attacking nodes are made from compromised nodes. The investigation logic tracks down to the attack activities directly without inspecting if the attack activities are caused by the change of nodes.

7 Conclusions

In this paper, we presented a specification-based investigation logic for revealing abuse of CA protocols in multi-channel wireless networks. We identified the key dependency relations among channel assignment operations and modeled the process of channel assignment as information flows with logged actions that are collected from nodes. We showed the fundamental differences among good actions and malicious actions based on their statistic profiles. We developed investigation reasoning approaches that inspect consistency among actions to locate the sources of abuse. We implemented the investigation logic and conducted

simulation with two CA protocols. The simulation shows that the investigation logic can achieve decent quality of zero false negative rate and low false positive rate.

Acknowledgments. This work was supported by the National Science Foundation under Grant No. 0915318, 1048339, and 0916469.

References

1. INET, `http://inet.omnetpp.org/`
2. Alicherry, M., Bhatia, R., Li, L.E.: Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks. In: Proc. of ACM MobiCom, pp. 58–72 (2005)
3. Bicket, J., Aguayo, D., Biswas, S., Morris, R.: Architecture and evaluation of an unplanned 802.11b mesh network. In: Proc. of ACM Mobicom, pp. 31–42 (2005)
4. Boreale, M., Buscemi, M.G.: A method for symbolic analysis of security protocols. Theor. Comput. Sci. 338(1-3), 393–425 (2005)
5. Chereddi, C., Kyasanur, P., Vaidya, N.H.: Design and implementation of a multi-channel multi-interface network. In: Proc. of International Workshop on Multi-Hop Ad Hoc Networks: from Theory to Reality, pp. 23–30 (2006)
6. Clark, S., Goodspeed, T., Metzger, P., Wasserman, Z., Xu, K., BBlaze, M.: Why (special agent) Johnny (still) can't encrypt: a security analysis of the APCO project 25 two-way radio system. In: Proc. of USENIX Security (2011)
7. Datta, A., Derek, A., Mitchell, J.C., Roy, A.: Protocol composition logic (pcl). Electr. Notes Theor. Comput. Sci. 172, 311–358 (2007)
8. Dhananjay, A., Zhang, H., Li, J., Subramanian, L.: Practical, distributed channel assignment and routing in dual-radio mesh networks. In: Proc. of ACM SIGCOMM, vol. 39, pp. 99–110 (2009)
9. Ferreira, E.W.T., de Oliveira, R., Carrijo, G.A., Bhargava, B.: Intrusion Detection in Wireless Mesh Networks Using a Hybrid Approach. In: Proc. of IEEE International Conference on Distributed Computing Systems Workshops, pp. 451–454 (2009)
10. Ganeriwal, S., Srivastava, M.B.: Reputation-based framework for high integrity sensor networks. In: SASN, pp. 66–77 (2004)
11. Gu, Q., Yu, M., Zang, W., Liu, P.: Lightweight Attacks against Channel Assignment Protocols in MIMC Wireless Networks. In: Proc. of IEEE ICC (2011)
12. Helman, P., Liepins, G.: Statistical Foundations of Audit Trail Analysis for the Detection of Computer Misuse. IEEE Transaction on Software Engeering 19(9), 886–901 (1993)
13. Jagadeesan, R., Jeffrey, A., Pitcher, C., Riely, J.: Towards a Theory of Accountability and Audit. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 152–167. Springer, Heidelberg (2009)
14. Kim, H., Gu, Q., Yu, M., Zang, W., Liu, P.: A Simulation Framework for Performance Analysis of Multi-Interface and Multi-Channel Wireless Networks in INET/OMNET++. In: Proc. of Communications and Networking Simulation Symposium (2010)
15. Kyasanur, P., Vaidya, N.H.: Routing and link-layer protocols for multi-channel multi-interface ad hoc wireless networks. SIGMOBILE Mob. Comput. Commun. Rev. 10(1), 31–43 (2006)

16. Lin, X., Rasool, S.: A Distributed Joint Channel-Assignment, Scheduling and Routing Algorithm for Multi-Channel Ad Hoc Wireless Networks. In: Proc. of IEEE INFOCOM, pp. 1118–1126 (2007)
17. Lincoln, P., Mitchell, J.C., Mitchell, M., Scedrov, A.: A probabilistic poly-time framework for protocol analysis. In: ACM Conference on Computer and Communications Security, pp. 112–121 (1998)
18. Martignon, F., Paris, S., Capone, A.: Design and implementation of MobiSEC: A complete security architecture for wireless mesh networks. Computer Networks 53(12), 2192–2207 (2009)
19. Mitchell, J.C.: Finite-State Analysis of Security Protocols. In: Vardi, M.Y. (ed.) CAV 1998. LNCS, vol. 1427, pp. 71–76. Springer, Heidelberg (1998)
20. Mitchell, J.C.: Security analysis of network protocols: logical and computational methods. In: PPDP, pp. 151–152 (2005)
21. Naveed, A., Kanhere, S.S.: Security Vulnerabilities in Channel Assignment of Multi-Radio Multi-Channel Wireless Mesh Networks. In: Proc. of IEEE GLOBECOM, pp. 1–5 (2006)
22. Ou, X., Boyer, W.F., McQueen, M.A.: A scalable approach to attack graph generation. In: ACM Conference on Computer and Communications Security, pp. 336–345 (2006)
23. Ou, X., Govindavajhala, S., Appel, A.W.: Mulval: A logic-based network security analyzer. In: 14th USENIX Security Symposium (2005)
24. Seshadri, A., Perrig, A., van Doorn, L., Khosla, P.: SWATT: SoftWare-based ATTestation for Embedded Devices. In: Proc. of IEEE Symposium on Security and Privacy, pp. 272–284 (2004)
25. Shin, D.-H., Bagchi, S.: Optimal monitoring in multi-channel multi-radio wireless mesh networks. In: Proc. of ACM MobiHoc, pp. 229–238 (2009)
26. Song, H., Xie, L., Zhu, S., Cao, G.: Sensor node compromise detection: the location perspective. In: IWCMC, pp. 242–247 (2007)
27. Sun, Y.L., Han, Z., Yu, W., Liu, K.J.R.: A trust evaluation framework in distributed networks: Vulnerability analysis and defense against attacks. In: INFOCOM (2006)
28. Xing, K., Cheng, X., Ma, L., Liang, Q.: Superimposed code based channel assignment in multi-radio multi-channel wireless mesh networks. In: Proc. of ACM MobiCom, pp. 15–26 (2007)
29. Yang, Y., Wang, X., Zhu, S., Cao, G.: Distributed software-based attestation for node compromise detection in sensor networks. In: SRDS, pp. 219–230 (2007)
30. Zang, W., Gu, Q., Yu, M., Liu, P.: An attack-resilient channel assignment mac protocol. In: NBiS, pp. 246–253 (2009)
31. Zhu, H., Lin, X., Lu, R., Ho, P.-H., Shen, X.: SLAB: A secure localized authentication and billing scheme for wireless mesh networks. IEEE Transactions on Wireless Communications 7(10), 3858–3868 (2008)