Automatic Multi-Step Signature Derivation from Taint Graphs

Martin Ussath, Feng Cheng, Christoph Meinel
Hasso Plattner Institute (HPI)
University of Potsdam, 14482, Potsdam, Germany
Email: {martin.ussath, feng.cheng, christoph.meinel}@hpi.de

***Abstract*—An increasing number of attacks use advanced tactics, techniques and methods to compromise target systems and environments. Such multi-step attacks are often able to bypass existing prevention and detection systems, such as Intrusion Detection Systems (IDSs), firewalls and anti-virus solutions. These security systems either use an anomaly-based or a signature-based detection approach. For systems that utilize a signature-based approach, it is relevant to use precise detection signatures to identify attacks. The creation of signatures is often complex and time consuming, especially for multi-step attacks.**

In this paper, we propose a signature derivation approach that automatically creates multi-step detection signatures from taint graphs. The approach uses the recorded log events of an attack and the event attribute tainting approach to correlate the events and to create a taint graph. This graph, which provides comprehensive details about the attack, is then used to derive a precise multi-step detection signature. Therewith, this approach can reduce the needed time to create a multi-step signature as well as the complexity of this process. For the evaluation of the proposed approach, we simulated a multi-step attack with real world attack tools and methods. Based on the recorded log events and the implemented signature derivation system we automatically derived a multi-step detection signature that describes all relevant events and their relations.

I. Introduction

Attackers try to compromise single computer systems and whole network environments on a regular basis. The attacks have different objectives, such as stealing credentials or exfiltrate sensitive information [1]. The used tactics, techniques and methods of an attack often depend on the sophistication of the adversary. Various security systems, such as Intrusion Detection Systems (IDSs), firewalls and anti-virus solutions, are deployed to prevent and detect attacks. These systems usually make use of signature-based or anomaly-based approaches. Standard attacks, which use well known attack methods for targeting a high number of systems in an automated way, can most often be detected with the currently deployed systems and approaches. In contrast to this, more sophisticated attackers are able to bypass existing security systems and compromise even high profile targets, such as Lockheed Martin [2], RSA [3] and Bit9 [4], although these companies probably have stat-of-the-art security systems in place. To propose reasonable prevention and detection approaches for more advanced attacks, it is necessary to have a comprehensive understanding of the multi-step attack activities.

For the identification of attacks, the analysis of recorded log events plays an important role. The detection of standard attacks is most often possible with simple single-step signatures. The reason for this is that such attacks most often have certain characteristics that can be easily identified in a single log event. Relevant characteristics, such as IP addresses, user agents or URLs, can be used to create such detection signatures. Due to the fact that standard attacks are often highly automated and target a large number of systems, they can be easily identified and a corresponding signature can be created. In general, it is possible to create signatures for standard attacks in an efficient way. For multi-step attacks the creation of reasonable and precise signatures is more difficult, due to the fact that these attacks can not be detected with the help of a single log event. For the detection of such attacks, it is necessary to identify and correlate the different malicious steps. Only in the context of the different events, which recorded the single malicious activities, it is possible to comprehensively detect a multi-step attack. Therefore, the creation of multi-step signatures is very time consuming and difficult. To create an accurate signature, the relevant relations between the different events need to be identified and described in a signature. Furthermore, it needs to be considered that sophisticated attacks are often customized to compromise different target infrastructures. Therefore, multi-step signatures should also allow the detection of attacks that are slightly different. For the detection of sophisticated attacks, it is necessary to be able to efficiently create precise multi-step signatures.

In this paper we want to propose an approach that is able to create detection signatures for multi-step attacks in an automated manner. Therefore, we use the event attribute tainting approach to identify and trace all events that belong to a multi-step attack. Through the correlation of the different events and the provided context information, it is possible to derive a multi-step signature from a taint graph. To describe the complex relations between the different attack steps we use the Event Description Language (EDL).

The remainder of this paper is structured as follows. In Section II we give a short overview about related work that address correlation approaches, signature languages and existing signature creation and derivation approaches. In the following section, we describe the event attribute tainting approach that is used to correlate log events and to create taint graphs. Furthermore, we explain the signature derivation approach and the procedure to derive a valid EDL signature from a taint graph. In Section IV we create a multi-step signature for a simulated attack to evaluate the proposed signature derivation approach. Finally, in Section V we conclude our work and propose future work.

II. Related Work

A. Correlation

Through the usage of different approaches, correlation algorithms are able to identify relations between objects or entities. In the context of attack detection, these approaches can be used to identify the different malicious activities of multi-step attacks based on events. For a comprehensive detection of complex attacks, it is necessary that the correlation approach is capable to identify the various malicious activities of an attacker and correlate them. In some cases, it might not even be possible to detect an attack without correlating most of the performed activities. The reason for this is that the single actions of an attacker might look benign and only the correlated activities reveal the performed attack.

The usage of timing relations is one approach to correlate similar events [5]. The objective is to aggregate events that do not provide relevant additional information for the analysis. This relatively simple correlation approach can support the management and diagnosis of infrastructures, but for the detection of multi-step attacks it is not sufficient to correlate events that are similar and have a timing relation. The reason for this is that complex attacks can be performed over a longer period of time and the attackers sometimes also change their attack methods.

Other correlation approaches try to identify relations between IDS events to detect multi-step attacks [6], [7], [8], [9]. For these approaches it is relevant that an IDS is capable to identify at least some of the attack steps and provide the corresponding alerts to the correlation system. If a sophisticated attack can bypass the anomaly-based or signature-based IDS, the corresponding alerts will not be created and the correlation approaches will not work. Furthermore, these systems rely on patterns and rules that describe the different correlation scenarios. This shows, that these correlation approaches require a substantial amount of knowledge about an attack, to define the correlation rules and also the signatures as well as thresholds of the IDS. In general, such detailed information is most often not available for multi-step attacks.

B. Signature Languages

Detection systems often rely on signatures to detect attacks. Signatures describe relevant characteristics of an attack that can be used to identify the malicious activities of an attacker. It is relevant to use precise signatures, because otherwise the detection system might trigger a high number of false positive alerts. Nevertheless, a signature should also be flexible, so that a slightly modified attack can also be identified. These requirements show that the creation of reasonable signatures is challenging. Furthermore, it is also crucial to select a suitable language to describe a precise signature.

Various existing systems, such as Snort, Bro or Mod Security, are able to detect single-step attacks. The corresponding signature languages of these systems provide different possibilities to describe characteristics of an attack. Due to the fact that these languages can only be used to describe single-step attacks, the described signatures focus on single requests or network packets. Therefore, the creation of new signatures is relatively easy and also the detection systems can evaluate the signatures relatively fast. Nevertheless, more complex multi-step attacks, which can only be detected when multiple malicious events are considered within a signature, can not be described with these languages.

Signature languages that support the description of multi-step attacks need to provide additional capabilities and are therefore often more complex. Nevertheless, also signature languages, such as STATL [10], LAMDA [11], SHEDEL and EDL [12], provide different levels of expressiveness [13]. In general, it is often more complex to describe a precise signature for a multi-step attack, due to the multiple events and their relations that need to be considered in the signature. Furthermore, it requires more resources to evaluate multi-step signatures, because different states and other context information need to be available during the processing of the signature.

C. Signature Derivation and Creation

The creation of reasonable signatures is challenging, due to the fact that the described characteristics of an attack should allow a precise identification of the malicious activity. At the same time the description of the attack should also be flexible, so that a slightly modified versions of the attack can also be detected with the same signature. These requirements apply to single-step signatures as well as multi-step signatures, but the creation of signatures for multi-step attacks is more complex and therefore also often more time consuming.

To make the derivation and creation of signatures more efficient, different automatic and semiautomatic approaches were proposed that are capable to extract the characteristics of an attack and describe a corresponding signature [14], [15], [16]. In a first step these approaches need to identify the relevant characteristics of an attack that can be used within a signature. This requires that these systems know if a packet or event belongs to an attack or not. Therefore, they use data from honeypot systems or rely on correlated alerts. If a multi-step signature should be created, the approaches also need to describe the relations between the different steps in the signature. The automatic and semiautomatic approaches often focus on single-step signatures and if these systems support the creation of multi-step signatures then they are only able to describe relatively basic relations between the different steps, due to the used correlation approach.

III. Signature Derivation Approach

For the creation of multi-step signatures, it is necessary to use a suitable correlation approach that provides detailed information about the different attack steps and their relations.

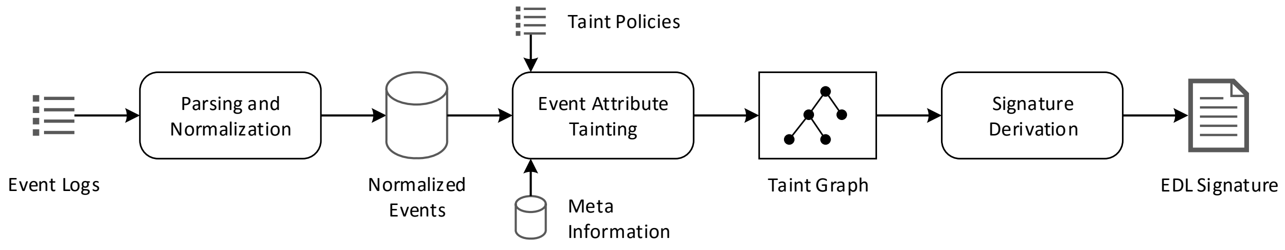


Fig. 1. Signature Derivation Steps

Therefore, our approach derives signatures from so-called taint graphs. These graphs provide comprehensive details about all log events that recorded the malicious activities of an attacker and the corresponding event relations. The event attribute tainting approach is used to create taint graphs from the identify events and their relations [17]. Due to the fact, that an automated correlation approach is used to provide all relevant characteristics of a multi-step attack, also the signature derivation approach can be automated. Figure 1 shows the different steps of the signature derivation approach. Subsequently, the different steps are described in more detail.

A. *Event Attribute Tainting*

1) Concept: For the correlation of log events it needs to be considered that all correlation approaches rely on the fact that most of the malicious activities of an attacker were recorded within events and the relations between these events are traceable. Furthermore, the events need to contain meaningful attributes that allow to identify relations between different events. Nevertheless, the log events do not provide all details about the performed activity and therefore only a limit amount of information can be used to identify relations between events, which leads to a certain inaccuracy.

The event attribute tainting approach uses the general concept of tainting [18] and applies it to event correlation, to provide an automated way to identify related events of a multi-step attack. For the identification of all relevant events, the correlation approach taints characteristic event attributes and propagates the taint to related events with same or similar attributes. With this procedure it is possible to identify all events that belong to a multi-step attack and the result of the correlation allows to trace an attack in a comprehensive way.

2) Requirements: For the tainting based correlation approach, it is necessary to parse and normalize the event logs that should be processed. Only if the correlation system knows the structure of the events and the semantic of the different event attributes, it is possible to identify relations between events. This is especially relevant for attributes that can be changed by an attacker, because these attributes might influence the propagation of the taint and also the result of the correlation system. Due to the normalized structure of the events, the tainting approach is also able to correlate different types of events that have diverse attributes and a different structure.

Next to the normalized events, the event attribute tainting approach also requires initial taint sources and taint policies. The initial taint sources are usually the indicators that are used as starting point for an investigation. These indicators describe characteristics of a potentially malicious activity that might belong to a multi-step attack. The taint policies describe the relevant attributes and the potential attribute relations in an abstract way. Based on this information taint is propagated between the different events. Therefore, taint policies are highly relevant for the correlation approach.

3) Taint Propagation Procedure: The event attribute tainting approach uses the following procedure to propagate the taint between related events. In a first step the initial taint sources are used to identify events that contain the described attributes of the initial indicators. These events get tainted and will be analyzed with the help of the taint policy. All event attributes that are specified in the policy will be extracted and added to the set of taint sources. Afterwards, the set of taint sources is used again to identify events that contain the described attributes and are therefore related with the already tainted events. These newly identified events will be analyzed again and also further relevant attributes, which can be used to identify relations, will be extracted and added to the set of taint sources. These steps of the procedure will be repeated until no new events and taint sources can be identified.

The different steps of the taint propagation can be represented in a graph. This so-called taint graph provides all correlation details that are necessary to retrace the identified relations between the different events. The nodes of the graph represent the tainted events and the edges between the nodes specify the attributes that were used to propagate the taint between the events. The taint graph clearly shows which taint source was extracted from which event and what relation could be identified with the help of the extracted taint source. Therefore, a taint graph makes the whole taint propagation process transparent and allows to easily identify potential imprecisions of the correlation result.

Although the general approach of identifying relations based on identical or similar event attributes is reasonable, in some cases this provide imprecise correlation results. The reason for this is that it might be necessary to use additional context information or more complex attribute relation descriptions to identify indirect or weak relations. Therefore, different enhancements can be used to improve the precision of the event attribute tainting approach. The usage of meta information can support the identification of indirect attribute relations. For example, the usage of passive DNS information could reveal a correlation between two events with different IP addresses. Furthermore, also the description of more complex taint sources, which consist of multiple different attributes, can improve the precision of the taint propagation. An additional method to improve the results of the correlation approach is to use taint labels from 1 to 100 to express the strength of a relation between two events. This also allows to describe uncertain event relations. To support this more fine granular taint labels, it is necessary to assign each event attribute a significance level and calculate the corresponding taint label for each event relation. In general, these enhancements need to be described within the used taint policy so that they can be evaluated during the propagation of the taint.

B. Signature Derivation from Taint Graphs

1) Signature Language: A taint graph describes all relevant log events and event relations of a multi-step attack that are necessary to create a detection signature. Due to the fact that the signature language needs to be capable to describe the multiple steps of an attack and the corresponding relations, our approach uses the Event Description Language (EDL) [12] to describe a multi-step signature.

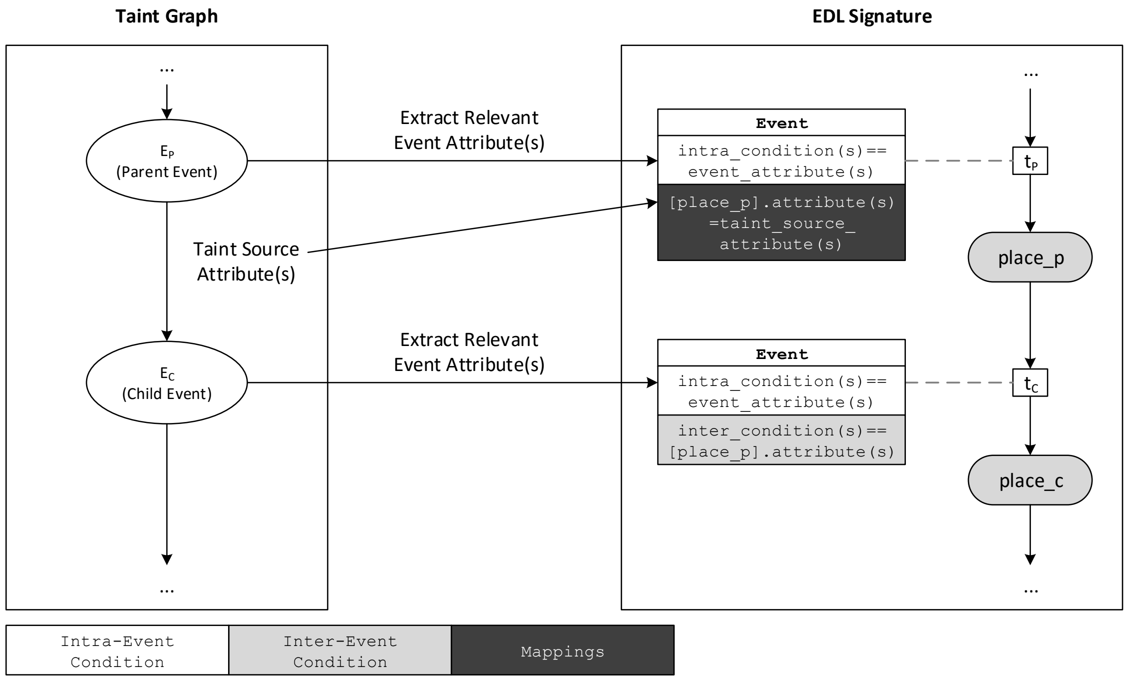
EDL represents multi-step attacks in a colored Petri net, where the system or signature state is represented as a place and the system events describe transitions between the places. This signature language is very expressive and therefore different intra-event conditions and inter-event conditions can be used to describe very precise signatures. The main parts of an EDL signature describe the existing places and the possible transitions. The description of a place consists of an optional type (e.g., initial or exit) and optional features that specify property mappings. To define a transition, it is necessary to specify a type, conditions, mappings and actions. Due to the expressiveness of EDL [12], [13], it is possible to describe complex multi-step signatures [19].

2) Signature Derivation Approach: The signature derivation approach uses the provided attack details of the taint graph to derive an EDL signature. Figure 2 shows the general approach that is used to derive an EDL signature from a taint graph. During the automatic derivation process, the approach tries to directly translate the taint graph into an EDL signature, to preserve the structure and to minimize the complexity of the signature. Nevertheless, it is necessary to add artificial transitions and places in an automated way to ensure the signature validity. The derivation approach maps events from the taint graph to transitions in the EDL signature and the places in the signature are used to describe the state after a certain event was observed. To describe the transitions, all relevant event attributes are used to specify intra-event conditions. These conditions need to ensure that only events with the correct attributes trigger the transition and move the state of the signature. Furthermore, the inter-event conditions of the transitions are used to describe the taint sources of the taint graph, which were used to propagate the taint to the corresponding event. The taint sources that are needed to describe the inter-event conditions are specified in the mapping section of the "parent" event transition and the extracted taint sources are stored within the place that describes the state after the "parent" event occurred. The stored values of the place can then be referenced in the conditions section of the "child" event. For the specification of the conditions, especially for the intra-event conditions, it is necessary to use only event attributes that are characteristic for a certain event. If nonrelevant attributes (e.g., timestamp) are used as conditions, the signature will only describe a concrete instance of an attack. This will result in false negatives, because such a signature cannot be used to detect other instances of the attack or slightly changed attacks. In cases where it is not possible to identify relevant attributes and specify the corresponding conditions, the signature will be imprecise and lead to false positives.

To create a valid and reasonable EDL signature it is necessary that the derivation approach automatically adds artificial transitions and places in three different cases. If one taint source leads to the tainting of multiple events, it is necessary to create an artificial transition and an artificial escape place. The reason for this is that the newly tainted events can most often occur in an arbitrary order and therefore it is not known which transition should be consuming and which should be non-consuming. To describe this case in an EDL signature, the approach sets all transitions of the newly tainted events to non-consuming and the artificial transition consumes the token of the "parent" event place (where the taint source was extracted) and it also has a non-consuming connections to all "child" event places. This artificial transition is triggered spontaneously, if all required tokens are available, and consumes the token of the "parent" event and moves it to the escape place. Therewith, it can be ensured that all "child" events can occur in an arbitrary order and the token of the "parent" event is also consumed.

Another case that needs to be considered during the derivation of the detection signature is, if multiple events and the corresponding transitions are identical. Due to the properties of EDL it is not possible to describe just the same transition multiple times. To describe multiple identical events, an artificial consuming transition has to describe the first occurrence of the event and change the state of the signature from the "parent" place to an artificial place. This additional place has another consuming transition with the conditions of the event and a counting variable. Another spontaneous transition changes the state from the artificial transition to the corresponding event places, if the event occurred as often as in the corresponding taint graph.

In the third case one artificial transition and place is needed to consume all tokens of the leaf places. Due to the fact



Taint Graph
EDL Signature
...
E_P (Parent Event)
E_C (Child Event)
Extract Relevant Event Attribute(s)
Taint Source Attribute(s)
Event
intra_condition(s)== event_attribute(s)
[place_p].attribute(s) =taint_source_ attribute(s)
inter_condition(s)== [place_p].attribute(s)
t_P
t_C
place_p
place_c
Intra-Event Condition
Inter-Event Condition
Mappings

Fig. 2. General Signature Derivation Approach

that the signature needs to ensure that all events of the taint graph were observed, it is necessary that an artificial transition is spontaneously triggered if the corresponding leaf places contain tokens. The artificial transition consumes all tokens of the leaf places and changes the state of the signature to the attack state, which indicates that all relevant events occurred and the signature matched these events.

3) Signature Abstraction: Due to the fact that different signatures may require different levels of abstractions, the automatic signature derivation approach might only provide a template for a concrete multi-step signature that need to be adjusted by a security analyst. The reason for this is that the automatically derived signatures precisely describe the correlated events and their relations, but in certain cases the system can not decide if an attribute or value is characteristic for an attack or can be changed dynamically. This could lead to signatures that are too generic or too concrete. To achieve a reasonable level of signature abstraction it might be necessary that a security analyst reviews the created signatures and adjust different parts of the description to ensure precise signatures.

IV. Evaluation

A. Approach

For the evaluation of the proposed automatic multi-step signature derivation approach, we simulated a successful attack against a web application that uses a vulnerable version of the Apache Struts framework. The recorded log events of this attack were normalized and stored in a database. Afterwards, these events and the attribute tainting based correlation approach were used to create a taint graph of the attack. To derive a valid multi-step EDL signature from the taint graph, we implemented a proof of concept system in Python that analyzes the taint graph and derives a multi-step detection signature.

B. Simulated Attack

To evaluate the signature derivation system, we simulated a successful multi-step attack against a web application. This application was deployed on a Tomcat server and used the Struts framework in version 2.3.15. This version of Struts has a known vulnerability (CVE-2013-2251) that allows remote command execution[1]. To show that our approach is able to describe signatures for real world multi-step attacks, we used an existing exploitation tool that is capable to exploit different vulnerabilities of the Struts framework [20]. Furthermore, the simulated attacker used the reDuh tunneling tool[2] [21] and the vulnerable web application to get access to the internal infrastructure of the target.

In the first step, the attacker exploits the vulnerability to gather relevant meta-information, such as active user and web path, from the target system to prepare further steps of the attack. This meta-information is then used to upload an additional POST-based upload script to the server. This additional upload script is necessary, due to the fact that the exploitation of the Struts vulnerability is performed over GET parameters,

[1]https://struts.apache.org/docs/s2-016.html
[2]https://github.com/sensepost/reDuh

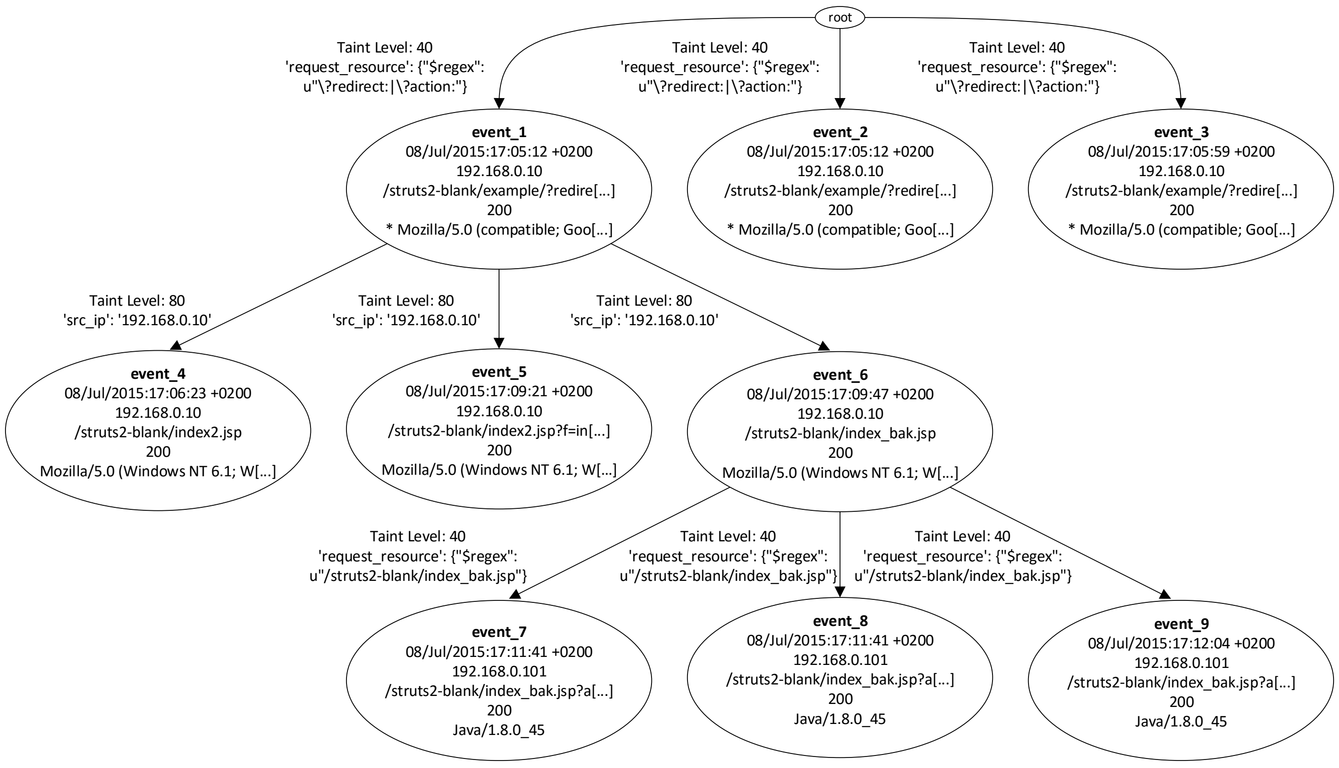

root
Taint Level: 40
'request_resource': {"$regex": u"\?redirect:|\?action:"}
Taint Level: 40
'request_resource': {"$regex": u"\?redirect:|\?action:"}
Taint Level: 40
'request_resource': {"$regex": u"\?redirect:|\?action:"}
event_1
08/Jul/2015:17:05:12 +0200
192.168.0.10
/struts2-blank/example/?redire[...]
200
* Mozilla/5.0 (compatible; Goo[...]
event_2
08/Jul/2015:17:05:12 +0200
192.168.0.10
/struts2-blank/example/?redire[...]
200
* Mozilla/5.0 (compatible; Goo[...]
event_3
08/Jul/2015:17:05:59 +0200
192.168.0.10
/struts2-blank/example/?redire[...]
200
* Mozilla/5.0 (compatible; Goo[...]
Taint Level: 80
'src_ip': '192.168.0.10'
Taint Level: 80
'src_ip': '192.168.0.10'
Taint Level: 80
'src_ip': '192.168.0.10'
event_4
08/Jul/2015:17:06:23 +0200
192.168.0.10
/struts2-blank/index2.jsp
200
Mozilla/5.0 (Windows NT 6.1; W[...]
event_5
08/Jul/2015:17:09:21 +0200
192.168.0.10
/struts2-blank/index2.jsp?f=in[...]
200
Mozilla/5.0 (Windows NT 6.1; W[...]
event_6
08/Jul/2015:17:09:47 +0200
192.168.0.10
/struts2-blank/index_bak.jsp
200
Mozilla/5.0 (Windows NT 6.1; W[...]
Taint Level: 40
'request_resource': {"$regex": u"/struts2-blank/index_bak.jsp"}
Taint Level: 40
'request_resource': {"$regex": u"/struts2-blank/index_bak.jsp"}
Taint Level: 40
'request_resource': {"$regex": u"/struts2-blank/index_bak.jsp"}
event_7
08/Jul/2015:17:11:41 +0200
192.168.0.101
/struts2-blank/index_bak.jsp?a[...]
200
Java/1.8.0_45
event_8
08/Jul/2015:17:11:41 +0200
192.168.0.101
/struts2-blank/index_bak.jsp?a[...]
200
Java/1.8.0_45
event_9
08/Jul/2015:17:12:04 +0200
192.168.0.101
/struts2-blank/index_bak.jsp?a[...]
200
Java/1.8.0_45

Fig. 3. Taint Graph of the Simulated Attack

which limits the size of the files that can be uploaded. In the third step of the attack, the upload script is used to upload the reDuh server component to the web server. Afterwards, the simulated attacker configures the server component to create a tunnel. In the last step of the attack, the created tunnel is used to get access to the internal infrastructure.

During the simulation of the attack, we also changed the source IP address of the attacker, because this is also common for real world adversaries and makes the correlation as well as the detection signature more complex. All the performed malicious activities were recorded in the web server logs, which are utilized to create the taint graph.

C. Taint Graph

For the creation of the taint graph it is first necessary to normalize the recorded web server log events and store them into a database. Afterwards, the attribute tainting based correlation approach can be used to correlate the events of the attack and create the taint graph. For the correlation of the simulated attack, the GET parameters *?redirect:* and *?action:* were used as initial taint sources. These two parameters can be used to exploit the remote command execution vulnerability in Struts. Furthermore, Listing 1 shows the taint policy that was utilized to identify the relations between the events.

The resulting taint graph of the tainting based correlation system is shown in Figure 3. To improve the readability of the graph we truncated all strings with more than 30 characters and also reduced the number of events that recorded the usage of the reDuh tunnel. The graph clearly shows which event attributes were extracted from the events and used as taint sources to identify related events. The events 1, 2 and 3 are correlated with the root node, because all three requests contain one of the GET parameters that were defined as initial taint sources. Afterwards, the source IP address of event 1 is added to the set of taint sources and leads to the identification of events 4, 5 and 6. The remaining events (7, 8 and 9) are related with event 6 due to the usage of the reDuh server component that is stored in the *index_bak.jsp* file.

Listing 1
TAINT POLICY

```
{   "significance_value":
        { "src_ip": 80,
          "user_agent": 20,
          "request_resource": 40 },
    "taint_attributes":
        [ "src_ip" ],
    "conditional_taint_attributes":
        { "request_resource": 50,
          "user_agent": 60 },
    "node_labels":
        [ "time_stamp",
          "src_ip",
          "request_resource",
          "status_code",
          "user_agent" ]
}
```

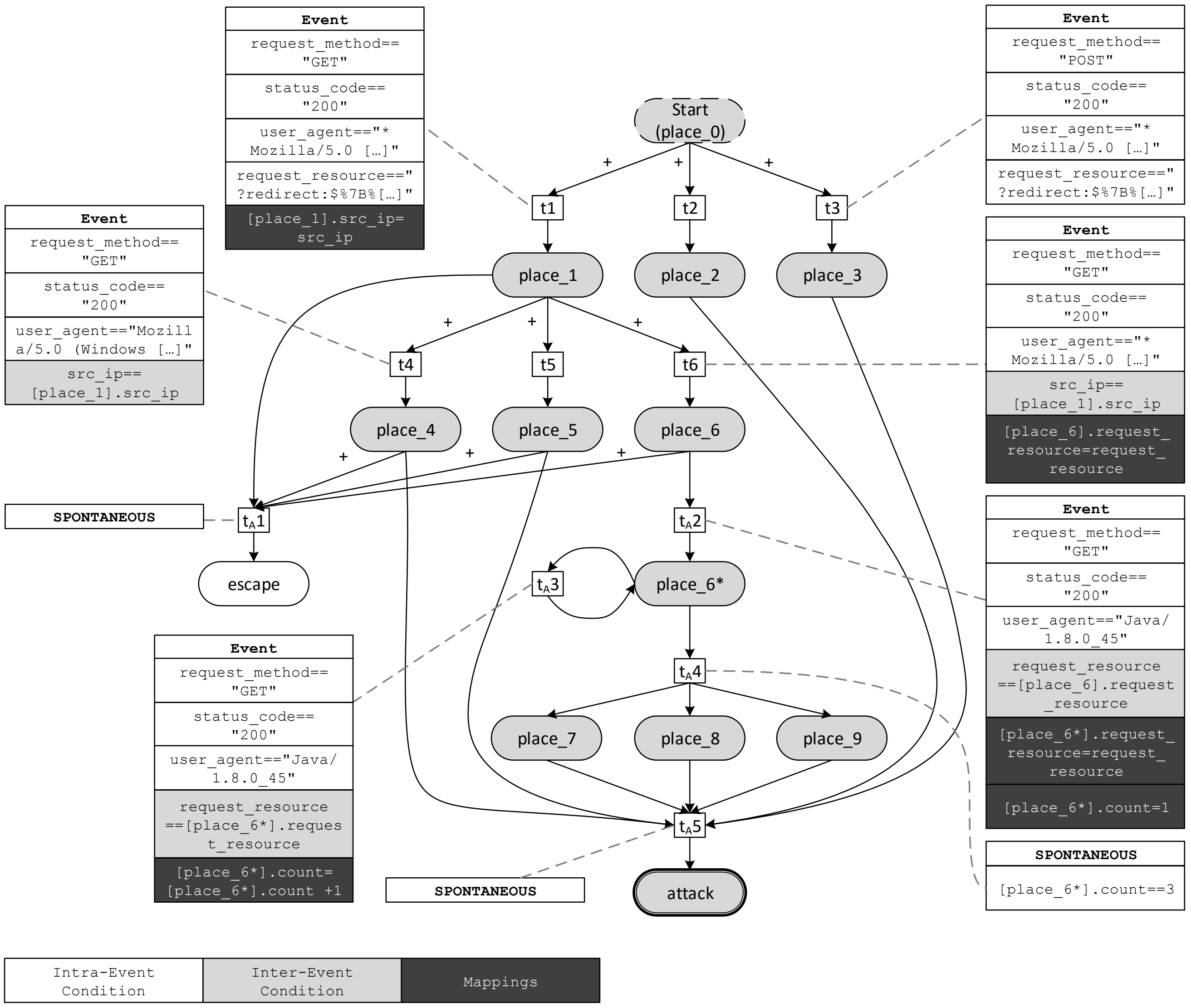


Fig. 4. EDL Signature Graph of the Simulated Attack

Further details of the event attribute tainting approach and the creation of taint graphs are described in [17].

D. Signature Derivation

The prototypical signature derivation system requires the serialized taint graph as input for the creation of the EDL signature. For the analysis of the graph the system deserializes the input and starts the derivation procedure. The general objective of the derivation system is to maintain the structure of the taint graph and only add artificial places and transitions to ensure the validity of the EDL signature. The derived signature graph for the simulated attack, including a selected number of transition descriptions, is shown in Figure 4. This signature describes the events and the identified relations of the taint graph within the corresponding transitions and places. Therefore, the derived signature precisely specifies all steps of the simulated attack. To ensure readability of the figure all overlong strings were truncated.

For the description of the EDL transitions, the derivation system defines the status code, the user agent and the request method of the events as *CONDITIONS* (intra-event conditions) of the transitions. The taint sources that were used to identify the relations between the events, are defined as *MAPPINGS* so that the specified attributes are extracted and can be used to describe inter-event conditions. For example, the transition *t1* describes the occurrence of event 1 and therefore it is necessary to specify a mapping to extract the source IP address of the request and assign the concrete value of it to the attribute of *place_1*. This value can then be used to identify the related events that recorded the further steps of the attacker. For each event node in the taint graph the signature has a corresponding place.

Due to the fact, that *place_1* has multiple successors and the corresponding log events can occur in an arbitrary order, an artificial transition ($t_A 1$) and the place (*escape*) are needed. This additional structure ensures that the signature still matches even if the attacker performs the activities in another order and the corresponding events 4, 5 and 6 (transitions *t4*, *t5* and *t6*) are then also recorded in this order.

The usage of the reDuh tunneling component is recorded in the events 7, 8 and 9. These events contain only very few relevant details about the performed activities. Therefore, the derivation system creates a structure for these identical events to count the occurrence. This requires three artificial transitions ($t_A 2$, $t_A 3$ and $t_A 4$) and one artificial place (*place_6**).

For the final detection of the simulated attack, it is necessary that all leaf places (places 2, 3, 4, 5, 7, 8 and 9) contain a token. This is necessary for transition $t_A 5$, which can change the state of the signature to the *attack* place. If the signature reaches this state, all described attack steps and the corresponding log events cloud be observed and detected.

V. Conclusion and Future Work

In this paper, we presented an automated approach that is able to derive multi-step detection signatures from taint graphs. In the first step, the event attribute tainting approach is used to automatically correlate log events from a multi-step attack. This correlation approach is able identify the relations between the different log events and the corresponding taint graph provides comprehensive details about all attack activities. In the second step, the taint graph is used to automatically derive a multi-step EDL signature that describes the attack in a precise way. The created detection signature describes all relations between the events and the relevant event attributes. This automated signature derivation approach allows to reduce the complexity and the needed time to create multi-step signatures.

For the evaluation of this approach, we implemented a prototypical signature derivation system and simulated a multi-step attack with real world attack methods and tools. The automatically derived EDL signature shows that the approach is able to describe complex attacks and create valid detection signatures. This signature does not only describe the concrete instance of the attack, but also similar attacks.

To the best of our knowledge, the described signature derivation approach is the first approach that is able to automatically create complex multi-step signatures from correlated log events, by using various event attributes to describe relevant relations between the different malicious activities of an attack.

For the future work it might be interesting to evaluate which attributes should be used to define intra-event conditions in EDL signatures. Therefore, the attributes of different event types need to be analyzed and a rating for each attribute should be calculated. With such a rating the proposed signature can be improved and the security analyst that verifies the signature can more easily decide which attributes should be used for the description of the intra-event conditions.

Another task for future work cloud be to describe multi-step signatures in a standardized cyber threat intelligence format to allow multiple parties to derive specific signature from the generic description of a multi-step attack. A widely used format for sharing threat intelligence is STIX (Structured Threat Information eXpression), which can support the description of attribute relations and multi-step attacks [22].

References

[1] Mandiant Conculting, "M-Trends 2016," *Special Report*, Feb. 2016.

[2] M. J. Schwartz, "Lockheed Martin Suffers Massive Cyberattack," *InformationWeek - DarkReading*, May 2011.

[3] J. Markoff, "SecurID Company Suffers a Breach of Data Security," *The New York Times*, Mar. 2011.

[4] B. Krebs, "Security Firm Bit9 Hacked, Used to Spread Malware," *Krebs on Security*, Feb. 2013.

[5] G. Liu, A. K. Mok, and E. J. Yang, "Composite events for network event correlation," in *Proceedings of the IEEE International Symposium on Integrated Network Management*, 1999.

[6] F. Cuppens and A. Miège, "Alert Correlation in a Cooperative Intrusion Detection Framework," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2002.

[7] C. Krügel, T. Toth, and C. Kerer, "Decentralized Event Correlation for Intrusion Detection," in *Proceedings of the International Conference Seoul on Information Security and Cryptology*, ser. ICISC '01. Springer, 2002.

[8] W. Li and S. Tian, "An ontology-based intrusion alerts correlation system," *Expert Systems with Applications*, vol. 37, no. 10, 2010.

[9] P. Ning, Y. Cui, and D. S. Reeves, "Constructing Attack Scenarios Through Correlation of Intrusion Alerts," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*. ACM, 2002.

[10] S. T. Eckmann, G. Vigna, and R. A. Kemmerer, "STATL: An Attack Language for State-based Intrusion Detection," *Journal of Compututer Security*, vol. 10, no. 1-2, Jul. 2002.

[11] F. Cuppens and R. Ortalo, "LAMBDA: A Language to Model a Database for Detection of Attacks," in *Proceedings of the International Workshop on Recent Advances in Intrusion Detection (RAID)*. Springer, 2000.

[12] U. Flegel and M. Meier, "Modeling and Describing Misuse Scenarios Using Signature-Nets and Event Description Language," *it - Information Technology*, vol. 54, no. 2, 2012.

[13] M. Meier, "A Model for the Semantics of Attack Signatures in Misuse Detection Systems," in *Information Security*. Springer, 2004.

[14] C. Kreibich and J. Crowcroft, "Honeycomb: Creating Intrusion Detection Signatures Using Honeypots," *SIGCOMM Computer Communication Review*, vol. 34, no. 1, Jan. 2004.

[15] U. Thakar, N. Dagdee, and S. Varma, "Pattern Analysis and Signature Extraction for Intrusion Attacks on Web Services," *International Journal of Network Security & its Applications (IJNSA)*, vol. 2, no. 3, 2010.

[16] L. Wang, A. Ghorbani, and Y. Li, "Automatic Multi-step Attack Pattern Discovering," *International Journal of Network Security*, vol. 10, no. 2, 2010.

[17] M. Ussath, F. Cheng, and C. Meinel, "Event Attribute Tainting: A New Approach for Attack Tracing and Event Correlation," in *Proceedings of the IEEE Network Operations and Management Symposium (NOMS)*, 2016.

[18] E. J. Schwartz, T. Avgerinos, and D. Brumley, "All You Ever Wanted to Know About Dynamic Taint Analysis and Forward Symbolic Execution (but might have been afraid to ask)," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2010.

[19] D. Jaeger, M. Ussath, F. Cheng, and C. Meinel, "Multi-step attack pattern detection on normalized event logs," in *Proceedings of the IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)*, Nov. 2015.

[20] N. Hayashi, "Chinese Underground Creates Tool Exploiting Apache Struts Vulnerability," Aug. 2013.

[21] SensePost, "Pushing the Camel through the Eye of a Needle," *Black Hat USA*, 2008.

[22] M. Ussath, D. Jaeger, F. Cheng, and C. Meinel, "Pushing the Limits of Cyber Threat Intelligence: Extending STIX to Support Complex Patterns," in *Proceedings of the International Conference on Information Technology : New Generations (ITNG)*, 2016.