Augmenting Attack Graphs to Represent Data Link and Network Layer Vulnerabilities

Jaime C. Acosta
U.S. Army Research Laboratory
White Sands Missile Range, NM 88002
jaime.c.acosta.civ@mail.mil

Edgar Padilla
University of Texas at El Paso
El Paso, TX 79968

John Homer
Abilene Christian University
Abilene, TX 79601

***Abstract*—Attack graphs enable system stakeholders to understand the stepping stones or exploitation procedures that an adversary could potentially execute to impact the confidentiality, integrity, and availability of a network system. These graphs are used to assess risk and to determine components that, when hardened, contribute most to risk reduction.**

While these graphs are powerful and widely used in enterprise network systems they focus on application vulnerabilities; they currently do not incorporate weaknesses in the network backbone (e.g., routing) that could lead to traffic hijacking, spoofing, eavesdropping, and several others. In this paper, we describe our work in augmenting the MulVAL attack graph software to incorporate network layer misconfigurations. Through a case study, we show how our modular data pipeline, leveraging previous work in network layer attack impact prediction, can aid system stakeholders in identifying risk and deciding on risk reduction strategies.

I. Introduction

Analyzing the risk associated with a network system involves understanding the likelihood and impact of a threat acting on vulnerabilities. This is a critical process throughout product lifecycles. This risk analysis enables stakeholders to make key decisions about whether to mitigate (fix), transfer (e.g., purchase third-party insurance) or accept (do not fix and assume responsibility) vulnerabilities associated with a system. The risk analysis process is not a trivial task (risk metrics is a separate field of study); it is costly, intrusive, and requires expert-level knowledge associated with both the assessment process and system details (e.g., vulnerability and exploit data as well as user accounts and network configurations).

Assessments are executed throughout the system lifecycle process – early in development to identify design flaws and continuously as the system undergoes modification. These assessments can take different forms. Paper studies provide assessment based on open source intelligence (OSINT), working at the least intrusive level and involves collecting data from publicly available (or stakeholder-provided) resources. Penetration tests involve real analysts, or ethical hackers, connecting to real systems (usually in operational environments) to portray threat, acting as real adversaries in order to obtain realistic measures of likelihood and impact. Penetration tests are more intrusive and costly but are critical, especially for mid-scale to large-scale systems.

Attack graphs are a hybrid approach that can be used as a supplementary tool for paper studies and penetration tests. Attack graphs use a white-box perspective and provide a comprehensive view of all attacks that an adversary may execute in a network; showing which vulnerabilities can be exploited and used as pivots (or stages) during an attack in order to achieve an ultimate goal. For paper studies, attack graphs can show potentially *reachable* assets. For penetration tests, they can be used to show some untestable or theoretical attacks (this is true especially in the case where time is a constraint). These graphs are also useful in the mitigation process; e.g., it may be the case that fixing one vulnerability may eliminate a large number of potential attack paths. Attack graphs indicate the likelihood and potential impact of a successful attack against the system, which allows experts to fine-tune configuration for their specific environment. This is based on vulnerability scan data (from 3rd party tools such as Nessus and OVAL) and public resources such as the NIST National Vulnerability Database (NVD) and the associated Common Vulnerability Scoring System (CVSS).

Attack graphs may also be used alongside penetration tests. Analysts usually start with an incomplete (or incorrect) view of the system under test. As analysts learn more about the system (e.g., by pivoting into a remote subnet) they can update their knowledge base and regenerate the graph. These graphs also enable analysis of hypothetical scenarios (e.g., zero day attacks) and how fixing certain vulnerabilities contribute to risk mitigation.

While attack graphs are widely used with much success, they are incapable of describing risk associated with vulnerabilities or misconfigurations at the network and data link layer of the network stack. As a result, they ignore some well known attacks such as Address Resolution Protocol (ARP) Spoofing and route hijacking. These attacks can result in catastrophic impacts (in the case of route hijacking at a global scale [1]). In the case of network layer route hijacking, part of the reason is because route hijacking success or failure depends on specific implementations of routing protocols and on the operating environment. We tackle this issue by leveraging previous work in impact prediction for network layer attacks that have shown to be very effective for mobile ad-hoc networks (MANETs).

The contributions of this paper are the following:

- We provide a mechanism for integrating previous work in empirical-based network layer attack impact heuristics for network layer vulnerabilities.

- We describe a modular data pipeline that augments current attack graph capabilities by enabling representations of data link and network layer vulnerabilities and misconfigurations. The pipeline consists of pluggable components that can be interchanged depending on specific needs.
- We present a case study to demonstrate how this work can be used to investigate risk associated with hybrid wireless ad-hoc and wired infrastructure networks.

II. Related Work

Attack graphs identify and depict possible attack paths that can be performed by an intruder to compromise the victim's network and systems [2]–[4]. Attackers typically execute multi-stage attacks, accumulating additional privileges within the network by exploiting vulnerabilities repeatedly on key hosts until reaching a desired end-host [5]. An attack graph displays the identifiable steps (and paths over a series of such steps). Configuration and vulnerability data are modeled as preconditions in potential attack steps. Each access or privilege escalation potentially gained by a successful attacker is modeled as a postcondition, a consequence of the assembly of a sufficient set of preconditions. While much work has demonstrated improvements in performance and visualization [6], [7], the focus of attack graphs is mainly on application layer vulnerabilities, except for a few cases. For example, Rolando *et al.* [8] developed a formal approach, implemented using prolog, focused on strategic placement of network intrusion detection systems. However, this work does not show the multi-stage impacts of spoofing and does not incorporate empirical models for success/fail metrics.

Acosta *et al.* [9] developed such models by collecting a dataset from several emulation executions and training a classifier that (with some degree of certainty) will indicate whether a route attack is successful. Results show that these models are accurate for the protocols that were tested: wireless OLSR and OSPFMDRv3.

Related to data link layer modeling, lacking is an attack graph tool exists that models traffic hijacking at this layer.

III. Representing ARP Spoofing in MulVAL

One of our goals for this work was to augment an existing tool (not develop one from scratch) in order to leverage pre-existing risk analysis capabilities. We chose to use MulVAL as our platform because of the following features:

- Availability: open source
- Scalability: execution time grows $O(n^2)$ with the size of the network [6]
- Extensibility: its underlying reasoning engine is written in a logical programming language that enables users to extend functionality by writing custom rules
- Compatibility: leverages public vulnerability resources that are consistently being updated
- Broad acceptance: has been used by standardization organizations for large-scale risk analysis

MulVal takes as input a file with custom interaction rules and a file describing the scenario. The custom interaction rules enable extensible scenario descriptions. The scenario file models the network using primitives (unit clauses) to introduce facts, *e.g.*, specify available services, known vulnerabilities, user accounts, access on a system, and identifying nodes as web servers, network file servers, etc. MulVAL rules specify how knowledge is derived from known facts.

Listing 1 shows a simple interaction rule that states that if a principal's account on *Host* is compromised (*e.g.*, through stolen credentials) and *Host* is accessible (*e.g.*, through the network and a listening login service), an attacker may execute code on *Host* with the permission levels of the principal's account.

MulVAL generates the attack graph showing the primitives (facts) and derivations leading to the attack goal.

Listing 1. An execCode interaction rule.

```
execCode(Host, Perm) :-
  principalCompromised(Victim),
  hasAccount(Victim, Host, Perm),
  canAccessHost(Host).
```

In order to represent network layer, cross-subnet, hijacking with MulVAL, we started by focusing on a simpler (within-subnet) hijacking attack that shares many of the same required components: ARP Spoofing.

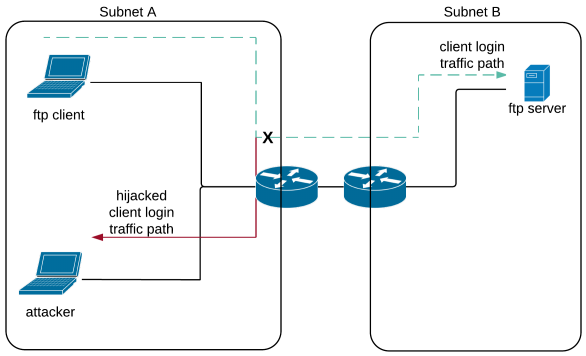

Fig. 1. ARP Spoofing scenario.

We began by developing an ARP Spoofing scenario (see Fig. 1). In this scenario, a client is communicating with an ftp server that is located in Subnet B. An attacker coexists within the client's subnet (Subnet A). All nodes in Subnet A are using the ARP to resolve MAC addresses. The attacker exploits an inherent vulnerability in the ARP (null authentication) that allows the attacker to spoof the gateway in Subnet A. All traffic that originates from the client and is destined for remote subnets will be funneled through the attacker. At this point, the attacker will be able to capture cleartext credentials (ftp vulnerability) that can then be used to log into the ftp server and execute code.

A. MulVAL custom interaction rules file

In this section, we describe the custom rules (see Listing 2) that we developed to represent ARP Spoofing.

Listing 2. Custom Rules for ARP Spoof.

```
1/** primitives **/
2gateway(_host).
3flowExists(_src, _dst, _protocol, _port,
    _user).
4/** interaction rules **/
5netAccess(H2, _protocol, _port) :-
6  gateway(H1),
7  advances(H1, H2),
8  netAccess(H1, _protocol, _port),
9  hacl(H1, H2, _protocol, _port).
10principalCompromised(Victim) :-
11  hasAccount(Victim, RemoteHost, User),
12  /* Arp spoof works only if the victim and
       attacker are in the same subnet*/
13  attackerLocated(Subnet),
14  hacl(Subnet, H, _anyProtocol, _anyPort),
15  /* Victim is using standard arp for address
       resolution*/
16  networkServiceInfo(H, arpd, _protocol,
       _port, _),
17  /* The standard arpd protocol is vulnerable
       to spoofing */
18  vulExists(H, arpSpoofVuln, arpd,
       remoteExploit, arpSpoof),
19  /* The User has an account on a login
       service on the remote host */
20  logInService(RemoteHost, Protocol, Port),
21  /* There is an active connection from the
       host to the remote machine */
22  flowExists(H, RemoteHost, Protocol, Port,
       User).
23logInService(H, Protocol, Port) :-
24  networkServiceInfo(H, ftpd, Protocol, Port,
       _).
```

1) Gateways: The gateway primitive (line 2) represents cross-subnet network access. For a node in one subnet to access a node in different subnet, the associated gateways must have connectivity through a series of hops. The execCode derivation rule (from Listing 1) requires that an attacker has network access (the canAccessHost rule) to the victim machine. Our overloaded definition of the netAccess rule (lines 5–9) will enable the derivation of canAccessHost. Subnets are specified in the scenario file (described in Section III-B2).

2) Traffic Flows: One important feature that was previously non-existent in MulVAL is the ability to describe a traffic flow. In order to represent traffic redirection it is important to describe active connections and any associated accounts. We added the flowExists primitive (line 3) to represent active traffic flows between two nodes (source and destination) communicating over a specific protocol, port, and associated with some user account.

3) ARP Spoofing and Eavesdropping: If a victim node that is colocated in the same subnet as the attacker is using the ARP service for address resolution, the attacker may redirect and read (or sniff) the victim node's traffic. To represent this, we overloaded the principalCompromised rule (lines 10–24).

Together, these rules enable representation of an attacker node obtaining a victim node's credentials if the victim node is using ARP and is using a login service on a remote subnet.

Listing 3. MulVAL input file for the ARP Spoofing Scenario

```
1/* attacker specification */
2attackerLocated(subnetA).
3attackGoal(
4   execCode(ftpServerHost,victimAccount)).
5/* topology */
6hacl(subnetA, ftpClientHost,_,_).
7hacl(subnetA, router1,_,_).
8hacl(router1, router2,_, _).
9hacl(router2, ftpServerHost, tcp , 21).
10/* cross-subnet comms through routers */
11gateway(router1).
12gateway(router2).
13/* client */
14networkServiceInfo(ftpClientHost, arpd,_,_,_).
15vulExists(ftpClientHost, arpSpoofVuln, arpd).
16vulProperty(arpSpoofVuln, remoteExploit,
     arpSpoof).
17/* ftp server */
18networkServiceInfo(ftpServerHost, ftpd, tcp,
     21, userLevel).
19hasAccount(victim, ftpServerHost,
     victimAccount).
20networkServiceInfo(ftpServerHost, arpd,
     _,_,_).
21/* comms */
22flowExists(ftpClientHost, ftpServerHost, tcp,
     21, victimAccount).
```

B. MulVAL ARP Spoofing scenario file

Listing 3 shows the ARP Spoofing scenario encoded as a MulVAL input file, specifying:

1) Modeled attack configuration: The attacker's expected location (within the ftp client's subnet) and goal (to execute code on the remote FTP server) are represented in lines 2–3.

2) Topology: Subnets are represented using the MulVAL hacl (host-access control list) rules. The ftp client and ftp server are configured in different subnets (lines 6–12); in order to communicate, traffic must hop through gateways, *i.e.*, router1 and router2.

3) Node configuration: Two nodes are actively communicating (an FTP client and an FTP Server), using ARP (lines 14–22).

C. MulVAL ARP Spoofing attack graph

MulVAL accepts these rules as input, considers potential multi-stage attacks, and produces an attack graph based on the provided network scenario (see Fig. 2). The attacker starts in Subnet A and executes ARP Spoof and eventually derives execCode as shown in Listing 1).

IV. Representing Route Hijacking in MulVAL

In order to generate attack graphs for network layer vulnerabilities, we needed to (1) develop MulVAL custom interaction rules to represent route hijacking, (2) incorporate the elements (e.g., node routes, attacker proximity, routing protocol used,

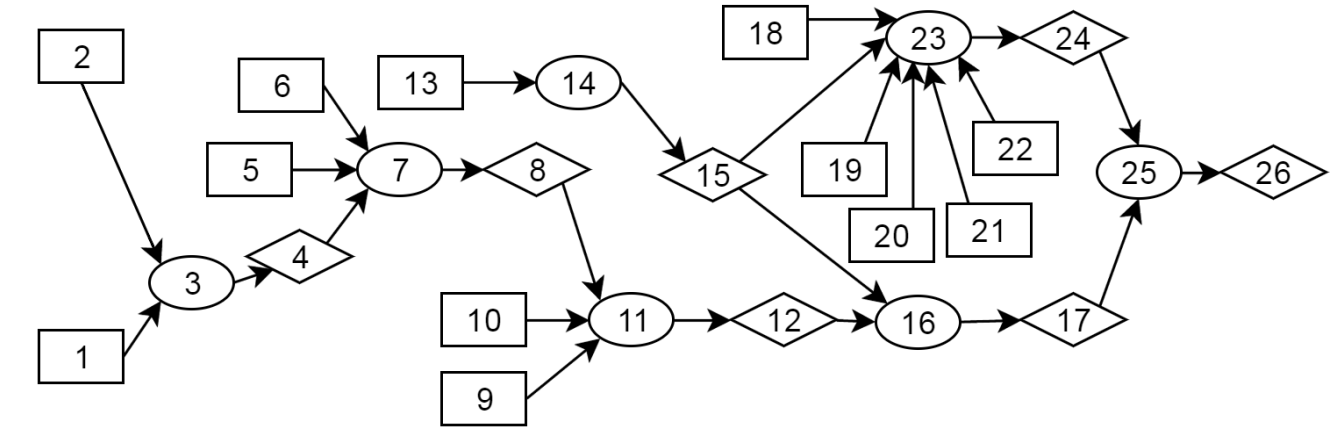

Fig. 2. Attack graph for ARP Spoofing.

traffic type) that are required to determine the success or failure of hijacking attacks, and (3) improve usability by automating the attack graph generation process. As a basis, we used the scenario in Fig. 3 which is similar to the ARP Spoofing scenario except that all nodes are gateways. OLSR is used for dynamic routing and the attacker is conducting an IP address spoofing attack (masquerading as the ftp server node) in order to hijack traffic from the ftp client.

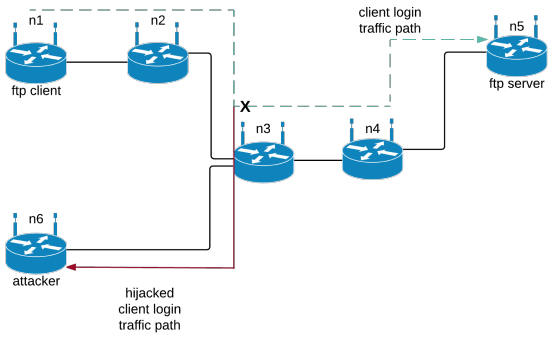

Fig. 3. Route hijack scenario.

A. Custom Rules for Route Hijacking

In addition to the rules used for ARP Spoofing, we overloaded principalCompromised to account for weak or null authentication in dynamic routing protocols (see Listing 4). This rule can be read as follows: *the principal is compromised if it has an account, an active connection to a login service on a node, and OLSR is misconfigured*. Unlike the ARP Spoofing rule, this principalCompromised rule does not contain the hacl and attackerLocated primitives. Route hijacking does not require the attacker to be located in the victim's subnet.

Listing 4. Custom Rule for Null or Weak Authentication in OLSR

```
principalCompromised(Victim) :-
  /* The victim has a user account on the
     remote host */
  hasAccount(Victim, RemoteHost, User),
  /* nrlolsr is being used */
  networkServiceInfo(H, nrlolsr, olsr,
     _no_port, _user),
  /* nrlolsr is misconfigured allowing
     traffic hijacking */
  vulExists(H, nrlolsrVul, nrlolsr,
     remoteExploit, nrlolsrHijack),
  /* The User has an account on a login
     service on the remote host */
  logInService(RemoteHost, Protocol, Port),
  /* There is an active connection from the
     host to the remote machine */
  flowExists(H, RemoteHost, Protocol, Port,
     User).
```

In order to determine if a flow in a scenario is hijackable, we leveraged previous work in network layer attack impact prediction for MANETs [9]. We chose this method because it is based on empirical data and has shown to be accurate for

OLSR and OSPFv3MDR routing protocols. As an alternative, a user has the option of using theory-based rules, such as [10]. These impact prediction rules are used to generate the MulVAL attack graph input file.

B. Data Pipeline for Generating Attack Graphs

To facilitate the development of these attack graphs, we created an automated and modular data pipeline (see Fig. 4) that only requires a user to enter network scenario data at the start of the pipeline, however, data can modified or substituted at any point in the pipeline.

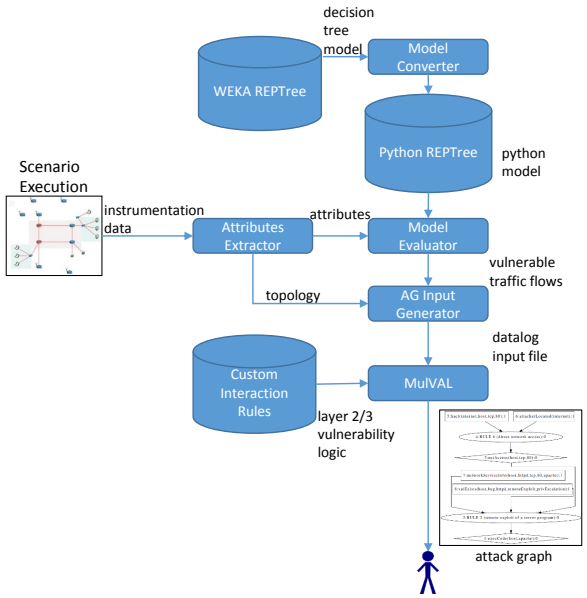

Fig. 4. Modular data pipeline for generating attack graphs.

All of the components in the pipeline read and write data in XML format. The *Attributes Extractor* reads information related to a network scenario including attack type, attacker position, routing protocol, traffic data, and route tables. These tables can be manually entered or extracted from emulation, simulation, or field data (e.g., using remote access scripts or Nessus plugins). For this scenario, we used the common open research emulator (CORE) [11] to configure the scenario. We executed the scenario, and after routes converged, we extracted this information from each node. This data is used to generate attributes to represent each flow in the scenario including attacker proximity, transport protocol, and others (see [9] for more details). The Model Evaluator uses the attributes to query the WEKA decision tree rules (that are converted into python scripts by the *Model Converter*) in order to determine which flows are vulnerable (i.e., could be hijacked).

For each vulnerable flow identified, the *AG Input Generator* will append a vulExists entry (associated with OLSR) to the MulVAL input file. The *AG Input Generator* also uses the data from the *Attributes Extractor* to encode network scenario information in the MulVAL file.

Listing 5 shows generated MulVAL input file. The topology information (lines 2–13) is obtained by the *Attributes Extractor* using the route information from the CORE execution (in our case). The network scenario configuration (lines 17–26) is provided as input by the user. Line 29 is generated using the results of the *Model Evaluator*.

Listing 5. MulVAL input file for Route Hijack Scenario

```
1 /* Network Topology Definitions:*/
2 hacl(ftpClientHost, n2, _proto, _port).
3 hacl(n2, n3, _proto, _port).
4 hacl(n3, n4, _proto, _port).
5 hacl(n4, ftpServerHost, _proto, _port).
6 hacl(n6, n3, _proto, _port).
7
8 gateway(ftpClientHost).
9 gateway(n2).
10 gateway(n3).
11 gateway(n4).
12 gateway(ftpServerHost).
13 gateway(n6).
14
15 /* Flow Definitions: */
16 /* Flow #1 :*/
17 flowExists(ftpClientHost, ftpServerHost,
       'TCP', 21, flow1Account).
18 hasAccount(flow1Principal, ftpServerHost,
       flow1Account).
19 networkServiceInfo(ftpClientHost, nrlolsr,
       olsr, _NA_port_layer3, _NA_perm_layer3).
20
21 /* Attack Configuration */
22 attackerLocated(n6).
23 attackGoal(execCode(ftpServerHost, _)).
24
25 /* A cleartext loginService exists on the
       remote machine */
26 networkServiceInfo(ftpServerHost, ftpd,
       'TCP', 21, flow1Account).
27
28 /* Vulnerable Flow */
29 vulExists(ftpClientHost, nrlolsrVul, nrlolsr,
       remoteExploit, nrlolsrHijack).
```

V. Case Study

Attack graphs are useful for identifying the multi-stage attacks that an adversary may use to accomplish a goal in a network. We developed a case study consisting of a notional heterogenous scenario with both tactical and strategic components (see Fig. 5). In this scenario, a 10 node wireless sensor network (that could also represent a MANET) is connected to a wired network containing login services and a workstation across two subnets. All routers use dynamic routing: wireless use OSPFv3MDR and wired use Quagga OSPFv3. There are three traffic flows: (1) node *n4* accesses the registration server on *n10* to gain access to the network (2) *n4* accesses a ftp server on node *n12* and (3) within the wired network, a workstation accesses a web server on a remote subnet.

In this scenario, we represented the attacker located on an arbitrary node in the wireless network having the attack goal

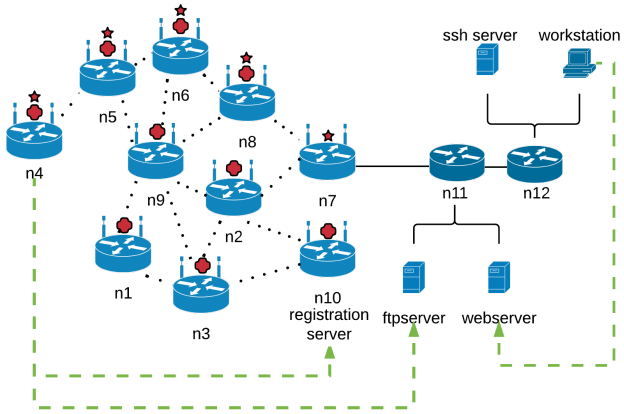

Fig. 5. Case study scenario.

to execute code on the ssh server. We used three different cases (each with a different spoofed victim). For each case, we executed our pipeline 10 times, each time varying only the attacker location.

First, we investigated the case when the attacker spoofs the IP address of the border node, *n7*. This is node is chosen because all traffic to and from the wired and wireless network will pass through this node. Using our hijacking rules and pipeline, we are able to determine the network nodes (indicated with red stars in Fig. 5) from which an attacker could achieve the predicted attack goal. Further investigation revealed that these nodes are along the traffic path; meaning that, in this scenario, route hijacking is unsuccessful.

Second, we investigated the case when the attacker spoofs the registration server. In our scenario, nodes must login to the registration server before becoming valid members of the network. The red pluses indicate locations where an attack could to be sourced to accomplish the attack goal. Hijacking is possible from any node except *n7*. To achieve the attack goal, the adversary first hijacks and sniffs credentials from *n4* to the registration server and then uses these to execute code on the ftpserver (same credentials). The attacker will use the ftpserver as a pivot (becoming an insider) to conduct an ARP Spoofing attack. The attacker will obtain the within-subnet web traffic that is forwarded from gateway *n12* to sniff credentials from the workstation. The workstation user also has an account (same credentials) on the ssh server. The attacker uses the sniffed credentials to execute code on the ssh server.

Third, we investigated the case when a wireless node spoofs the IP address of the the ftp server. The impact prediction rules that we used from [9] do not consider scenarios when spoofing occurs across mixed wireless and wired routing protocols. We developed and validated (using CORE) a trivial rule for our scenario that indicates a successful hijack if the attacker advertises a smaller metric. In the pipeline, we injected our rule in the *Model Converter* output. In this case, the route metrics among wireless nodes are much lower than the route metrics from wireless to wired subnets (at least by 10). Therefore, the attacker will always advertise a lower metric and successfully hijack the the ftp traffic and accomplish the attack goal.

VI. Conclusions and Future Work

In this paper we've provided a set of rules and a modular data pipeline that can be used to represent data link and network layer hijacking attacks. Our case study shows how our work augments existing attack graph capabilities; and how these capabilities can be used to develop network hardening strategies in hybrid wired and wireless networks.

There are several areas for future work. The pipeline should support MulVAL's third-party vulnerability import capability, *i.e.*, we need to encode route misconfigurations using a scoring system such as NIST CCSS. The impact prediction models used in this paper were trained with default values for routing protocol configurations (e.g., values for Pollrate, LinkQualityLevel, HystScaling, etc.). Custom configurations should also be incorporated when building attack graphs. Lastly, the attack graph rules should include confidence metrics associated with hijacking to enable broader analysis, e.g., aggregated vulnerability metrics [12].

References

[1] S. Goldberg, "Why is it taking so long to secure internet routing?" *Communications of the ACM*, vol. 57, no. 10, pp. 56–63, 2014.

[2] C. Phillips and L. P. Swiler, "A graph-based system for network-vulnerability analysis," in *Proceedings of the 1998 workshop on New security paradigms*. ACM, 1998, pp. 71–79.

[3] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs," in *Security and privacy, 2002. Proceedings. 2002 IEEE Symposium on*. IEEE, 2002, pp. 273–284.

[4] K. Ingols, R. Lippmann, and K. Piwowarski, "Practical attack graph generation for network defense," in *null*. IEEE, 2006, pp. 121–130.

[5] J. Dawkins and J. Hale, "A systematic approach to multi-stage network attack analysis," in *Information Assurance Workshop, 2004. Proceedings. Second IEEE International*. IEEE, 2004, pp. 48–56.

[6] X. Ou, W. F. Boyer, and M. A. McQueen, "A scalable approach to attack graph generation," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 336–345.

[7] S. Jajodia and S. Noel, "Advanced cyber attack modeling analysis and visualization," DTIC Document, Tech. Rep., 2010.

[8] M. Rolando, M. Rossi, N. Sanarico, and D. Mandrioli, "A formal approach to sensor placement and configuration in a network intrusion detection system," in *Proceedings of the 2006 international workshop on Software engineering for secure systems*. ACM, 2006, pp. 65–71.

[9] J. C. Acosta and B. G. Medina, "Survivability prediction of ad hoc networks under attack," in *MILITARY COMMUNICATIONS CONFERENCE, 2012-MILCOM 2012*. IEEE, 2012, pp. 1–6.

[10] V. Santiraveewan and Y. Permpoontanalarp, "A graph-based methodology for analyzing ip spoofing attack," in *Advanced Information Networking and Applications, 2004. AINA 2004. 18th International Conference on*, vol. 2. IEEE, 2004, pp. 227–230.

[11] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, "Core: A real-time network emulator," in *Military Communications Conference, 2008. MILCOM 2008. IEEE*. IEEE, 2008, pp. 1–7.

[12] J. Homer, S. Zhang, X. Ou, D. Schmidt, Y. Du, S. R. Rajagopalan, and A. Singhal, "Aggregating vulnerability metrics in enterprise networks using attack graphs," *Journal of Computer Security*, vol. 21, no. 4, pp. 561–597, 2013.