International Journal On Engineering Technology and Sciences – IJETS™
ISSN(P): 2349-3968, ISSN (O): 2349-3976
Volume III, Issue V, May- 2016

PRIVACY-PRESERVING DETECTION OF SENSITIVE DATA EXPOSURE

Mr.Muhammed Mansoor P.M
ME-CSE II YEAR
Department of CSE
Jay Shriram Group of Institutions
Tirupur
mansoorpunathil@gmail.com

Mr.T.SEENIVASAN
Assistant Professor
Department of CSE
Jay Shriram Group of Institutions
Tirupur
ktseenivasan@gmail.com

Dr.RAJALAKSHMI.S
HOD
Department of CSE
Jay Shriram Group of Institutions
Tirupur
mrajislm@gmail.com

ABSTRACT

Privacy preserving data-leak detection model for preventing inadvertent data leak in network traffic. Our model supports detection operation delegation and Internet Service Provider (ISP) can provide data-leak detection as an add-on service to their customers using our model. We design, implement, and evaluate an efficient technique, fuzzy fingerprint, for privacy-preserving data-leak detection. Fuzzy fingerprints are special sensitive data digests prepared by the data owner for release to the DLD provider. The advantage of our method is that it enables the data owner to safely delegate the detection operation to a semi honest provider without revealing the sensitive data to the provider. We describe how Internet service providers can offer their clients DLD as an add-on service with strong privacy guarantees. The evaluation results show that our proposed method can support accurate detection with very small number of false alarms under various data-leaks scenarios.

Keywords

WSN's, Privacy Preserving, Routing, Message Authentication.

1. INTRODUCTION

When a sensor network is deployed in an unattended or hostile environment, the attacker may capture and reprogram sensor nodes, or adds their own sensor nodes into the network and induce the network to accept them as legitimate nodes. Once in control of a few sensor nodes, the attacker can mount various attacks from inside the network. One common type of attack is targeted at message authenticity and integrity. If the sender and the receiver are not within the transmission range of each other, an intruder on the path connecting them can modify pass by messages or inject false messages. It appears to be a solution that the sender and the receiver share a secret key, and the shared key is used by the sender to generate message authentication code (MAC) for any outgoing message, and by the receivers to verify the authenticity and integrity of any incoming message. If a message is tampered en route, it will be detected by the receiver. However this method is not effective due to the following reasons: First of all, it cannot authenticate messages that are multicast because, if one of the receivers is compromised, the attacker can use the secret key held by the compromised receiver to fake MACs for messages modified or injected by it itself to cheat other receivers. Secondly, the method only allows end-to-end message authentication while en-route forwarding nodes cannot authenticate pass by messages; the intruder may launch denial-of-service attacks by repeatedly change messages or adding false messages to deplete the communication resources of intermediate forwarding nodes.

To thwart the above attacks, each message should be verifiable by both its final receivers and its intermediate forwarders. This may be simply implemented on top of the public key infrastructure; each message is sent along with a digital signature generated by the sender using its private key, and every intermediate forwarder or final receiver can authenticate the message using the public key of the sender. This approach may include high overhead in terms of computational cost and network bandwidth consumption. To mitigate the overhead, researchers have proposed low-cost schemes that use symmetric keys and hash functions. In these schemes, each symmetric authentication key is shared by a set of sensor nodes, and the keys can be captured by the intruder as sensor nodes are compromised. These schemes are not resilient to large number of node compromises. Utilizing a one-way key chain and delayed disclosure of keys, the TESLA schemes and its variants can achieve message authenticity in the presence of a large number of node compromises. These methods need synchronization among nodes. They introduce delay in message authentication and the delay increases as the network scales up.

2 RELATED WORKS

ROUTING PROTOCOLS AND THEIR REQUIREMENTS

We introduce the necessary concepts for describing the proper operations of different routing methods. First briefly review the different types of routing protocols in wireless networks. After that we establish a mathematical model of wireless networks. Using this mathematical model, we formally define three requirements that a properly operated routing protocol must satisfy.

A. Routing Protocols Types

A routing protocol consists of two components: a path calculation algorithm and a packet forwarding scheme. We review the most commonly used path calculation algorithms and packet forwarding schemes in wireless networks. By classifying routing protocols based on their path calculation algorithms and packet forwarding schemes, we can find the design guidelines for routing metrics of different types of routing protocols.

Path Calculation Algorithms

Verities of path calculation algorithms are appropriate for different networks. In this study, we find three path calculation algorithms: flooding-based route discovery, Dijkstra's algorithm and the Bellman-Ford algorithm, all used in wireless routing. In flooding-based route discovery, to search for a path to a destination node, a source node floods a route request message through the entire network to explore multiple paths simultaneously and the destination node selects a single path among all the searched paths as the path between the source node and the destination node. In Dijkstra's algorithm a source node collects network topology information through periodic message exchanges among nearest nodes. Based on the collected information, the source node calculates its paths to the other nodes.

Packet Forwarding Schemes

In wireless networks, two packet forwarding schemes, source routing and hop-by-hop routing, are often used in different routing protocols. A source node puts the entire path of a flow in its packet headers and intermediate nodes forward the packets accordingly. In hop-by-hop routing, a source node only puts the destination addresses in its

packet headers. An intermediate node forwards packet based on its routing table, which stores the next hops for reaching each destination address.

QOS ROUTING PROTOCOL

We first present our path selection method. It is based on the distance-vector mechanism. We give the necessary and enough condition to determine whether a path is not worthwhile to be advertised. We then describe our new isotonic path weight. We show that the routing protocol based on this new path weight satisfies the optimality requirement. We present our hop-by-hop packet forwarding method which satisfies the consistency requirement. We apply to estimate the available bandwidth of a path. To simplify our discussion we use "available bandwidth" instead of "estimated available bandwidth" when the context is clear. On the other hand, "widest path" refers to the path that has the maximum estimated available bandwidth.

Path Selection

We would like to develop a distance-vector based method. In the traditional distance-vector method, a node only has to advertise the information of its own best path to its neighbors. Every neighbor can then identify its own best path. We mentioned that if a node only advertises the widest path from its own perspective, its neighbors may not be able to find the widest path. To show this considers the network in Fig. 1 where the number of each link is the available bandwidth on the link.

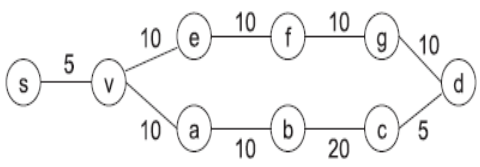

Fig. 1. An example of network topology.

ATTACKS ON ROUTING PROTOCOL

Many sensor network routing protocols were very simple and not developed as security in mind, so the attackers can launch various attacks in the network. Mainly network layer protocol suffers from many attacks like; IP spoofing or modifying the route information, selective jamming, sinkhole attack, wormhole attack, Injection attack, etc.

IP Spoofing, Modifying or replaying the route information

An adversary can launch the routing information corruption by IP spoofing, modifying or replying the routing information. By this an adversary can attracts or redirects the traffic, increases the latency, generate routing loops or creates false data etc.

Selective jamming attack

In the selective jamming attack, compromised node may refuse to forward certain packets and simply drop it. If an adversary drops the entire received packets, it behaves like a black hole attack. An adversary explicitly includes on the path of data flow to perform selective jamming.

Sinkhole and Wormhole attack

Basically, in the both sinkhole and wormhole attacks; the adversary tries to attract all the traffic from a particular area through a compromised node. Sinkhole attack mainly works by making a compromised node look attractive to the neighbor nodes to route the data packet and generally spoof, modify or drop the packet. Sinkhole attack give birth to many attacks like; selective forwarding, black hole, tempering the routing information etc. An adversary launch wormhole with two distant malicious nodes and try to attract the traffic by showing one hop distance to the

sink. Wormhole attack is very difficult to detect because it uses out-of-bound channel to route packets.

Sybil attack

In this attack a single node presents multiple identities to the other node in the network. It tries to mislead the node in neighbor detection, route formation and topology maintenance. The Sybil attack is a significant threat to many geographic and multi path routing protocols.

3 OUR WORK

We propose a new message authentication method to address the aforementioned limitations. Our method has following features: lightweight in terms of computation, communication and storage overhead; resilience to a large number of sensor node compromises; immediate authentication scalability; and non-repudiation. These features are attained by applying a number of novel techniques: we adopt polynomials for message authentication, which provides higher adaptability than existing authentication techniques based on multiple MACs and at the same time, keeps the advantage of immediate authentication held by those techniques. Next messages are authenticated and verified via evaluating polynomials, which incurs lower overhead than existing asymmetric cryptography-based authentication techniques such as digital signature. And then, independent and random factors are employed to perturb polynomial shares that preloaded to individual nodes, which significantly maximizes the complexity for the intruder to break the secret polynomial, and therefore renders the proposed approach to be resilient to node compromises. The proposed approach is the first one that applies the aforementioned techniques in message authentication for sensor networks, and also the first one that can achieve simultaneously the features of compromise-resiliency, flexible-time authentication, efficiency and non-repudiation without employing public key cryptography.

Performance

For every outgoing packet, the sender only needs to compute one HMAC function per packet per authentication chain, since the key chain can be pre-computed. We analyze the performance of our stream authentication scheme by measuring the number of packets per second that a sender can create. We suspect that an optimized C implementation might be at least twice as fast.

The communication overhead of our prototype is 24 bytes per authentication chain. Since we use 80 bit HMACMD5, both the disclosed key and the MAC are 10 bytes long. The remaining four bytes are used to send the interval index. The overhead of pre-computing the key chain is minimal. In our experiments we use an interval length of 1=10th of a second. To pre-compute a key chain long enough to authenticate packets for one hour, the sender precomputation time is only 36000=74626 _ 0:5 seconds. The computational overhead on the receiver side is the same as on the sender side, except that the receiver needs to recompute the key chain while the sender can pre-compute it. The overhead of computing the key chain is negligible, since it involves computing one HMAC functions in each time interval, and in practice only tens of intervals is used per second.

Algorithm used:

SHA1

- A symmetric key encryption algo. Invented by Ron Rivest.
- Normally uses 64 bit and 128 bit key sizes.
- Most popular implementation is in WEP for 802.11 wireless networks and in SSL.
- Cryptographically very strong yet very easy to implement.
- Consists of 2 parts: Key Scheduling Algorithm (KSA) & Pseudo-Random Generation Algorithm
- Using a secret key generate the RC4 keystream using the KSA and PRGA.
- Read the file and xor each byte of the file with the corresponding keystream byte.
- Write this encrypted output to a file.
- Transmit file over an insecure channel.

Results

The evaluation goal is to answer the following questions:

1) Can our solution accurately detect sensitive data leak in the traffic with low false positives and high true positives?
2) Does a using partial sensitive-data fingerprint reduce the detection accuracy in our system?
3) What is the performance advantage of our *fingerprint filter* over traditional Bloom filter with SHA-1?

3) How to choose a proper fuzzy length and make a balance between the privacy need and the number of alerts?

In the following subsection, we experimentally addressed and answered all the questions.

A. Accuracy Evaluation

We evaluate the detection accuracy in simple and complex leaking scenarios. First we test the detection rate and false positive rate in three simple experiments where the sensitive data is leaked in its original form or not leaked. Then we proposed accuracy evaluation on more critical leaking experiments to reproduce various real-world leaking detection scenarios.

1) Simple Leaking Scenarios: We test our prototype without partial disclosure in simple leaking scenarios, i.e., ¨S∗ = S∗. We generate 20,000 personal financial records as the sensitive data and store them in a text file. The data contains *person name*, *social security number*, *credit card number*, *credit card expiration date*, and *credit card CVV*.

To evaluate the accuracy of our strategy, we perform three separate experiments using the same sensitive dataset:

Exp.1 *True leak* A user leaks the entire set of sensitive data via FTP by uploading it to a remote FTP server.

Exp.2 *No leak* The non-related outbound HTTP traffic of 20 users is captured and given to the DLD server to analyze. No sensitive data should be confirmed.

Exp.3 *No leak* The Enron dataset as a virtual network traffic is given to the DLD server to analyze. Each virtual network packet reated is based on an email in the dataset.

No sensitive data should be confirmed by the data owner. Among the three experiments, the first one is designed to infer true positive rate. We manually check each packet and the DLD server detects *all* 651 real sensitive. The sensitivity value is less than one, because the high-layer headers in a packet are not sensitive. The next two experiments are designed to estimate the false positive rate. We found that none of the packets has a sensitivity value greater than 0.05. The results

indicate that our design performs as expected on plaintext.

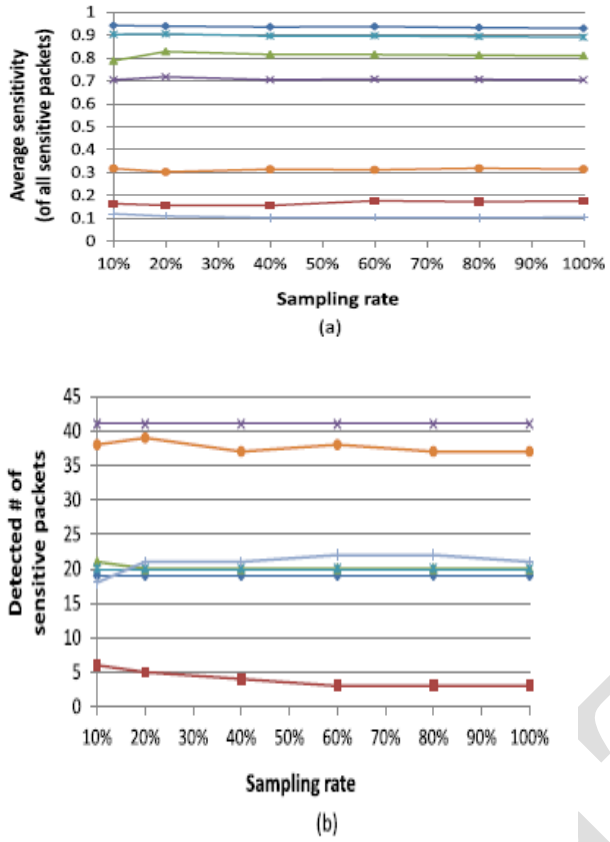

Fig. 2. Detection accuracy comparison in terms of (a) the averaged sensitivity and (b) the number of detected sensitive packets. X-axis is the partial disclosure rate, or the percentage of sensitive-data fingerprints revealed to the DLD server and used in the detection. [out] indicates outbound traffic only, while [all] means both outbound and inbound traffic captured and analyzed.

Fig. 2 shows the comparison of performance across various sizes of fingerprints used in the detection, in terms of the averaged sensitivity per packet in (a) and the number of detected sensitive packets in (b). These accuracy values reflect results of the POSTPROCESS operation.

The results show that the use of partial sensitive-data fingerprints does not much degrade the detection rate compared to the use of full sets of sensitive-data fingerprints. Extreme small sampling rates, e.g., 10%, may not provide sufficient numbers of fingerprints to describe the leaking characteristic of the traffic. The packet sensitivity estimation magnifies the signal as well as the noise produced by fingerprint sampling. The precision could be affected and drops, e.g., 6 packets with 10% vs. 3 packets with 100% for Keylogger in Fig. 2 (b).

The sensitivities of experiments vary due to different levels of modification by the leaking programs, which make it difficult to perform detection. WordPress substitutes spaces with +'s when sending the HTTP POST request. EZRecKb inserts function-key as labels into the original text. Typing typos and corrections also bring in modification to the original sensitive data. In results contain both outbound and inbound traffic and double the real number of sensitive packets in Blog and Wiki scenarios due to HTML fetching and displaying of the submitted data.

CONCLUSIONS

We proposed fuzzy fingerprint, a privacy-preserving data-leak detection model and present its realization. Using special digests, the exposure of the sensitive data is kept to a minimum during the detection. the proposed protocol has been identified to be not only capable of providing the conditional privacy preservation that is critically demanded in the VANET applications, but also able to improve efficiency in terms of the minimized anonymous keys storage at each OBU, fast verification on safety messages, and an efficient conditional privacy tracking mechanism. We have conducted extensive experiments to validate the accuracy, privacy, and efficiency of our solutions. In future we

plan to focus on designing a host-assisted mechanism for the complete data-leak detection for large-scale organizations.

REFERENCES

[1] X. Shu and D. Yao, "Data leak detection as a service," in Proc. 8th Int. Conf. Secur. Privacy Commun. Netw., 2012, pp. 222–240.

[2] Risk Based Security. (Feb. 2014). Data Breach Quick- View: An Executive's Guide to 2013 Data Breach Trends. [Online]. Available: https://www.riskbasedsecurity.com/reports/2013-DataBreachQuickView.pdf, accessed Oct. 2014.

[3] Ponemon Institute. (May 2013). 2013 Cost of Data Breach Study: Global Analysis. [Online]. Available: https://www4.symantec.com/mktginfo/whitepaper/053013_GL_NA_WP_Ponemon-2013-Cost-of-a-Data- Breach-Report_daiNA_cta72382.pdf, accessed Oct. 2014.

[4] Identity Finder. Discover Sensitive Data Prevent Breaches DLP Data Loss Prevention. [Online]. Available: http://www.identityfinder.com/, accessed Oct. 2014.

[5] K. Borders and A. Prakash, "Quantifying information leaks in outbound web traffic," in Proc. 30th IEEE Symp. Secur. Privacy, May 2009, pp. 129–140.

[6] H. Yin, D. Song, M. Egele, C. Kruegel, and E. Kirda, "Panorama: Capturing system-wide information flow for malware detection and analysis," in Proc. 14th ACM Conf. Comput. Commun. Secur., 2007, pp. 116–127.

[7] K. Borders, E. V. Weele, B. Lau, and A. Prakash, "Protecting confidential data on personal computers with storage capsules," in Proc. 18th USENIX Secur. Symp., 2009, pp. 367–382.

[8] A. Nadkarni and W. Enck, "Preventing accidental data disclosure in modern operating systems," in Proc. 20th ACM Conf. Comput. Commun.Secur., 2013, p. 1029–1042.

[9] A. Kapravelos, Y. Shoshitaishvili, M. Cova, C. Kruegel, and G. Vigna, "Revolver: An automated approach to the detection of evasiveweb-based malware," in Proc. 22nd USENIX Secur. Symp., 2013, pp. 637–652.

[10] X. Jiang, X. Wang, and D. Xu, "Stealthy malware detection and monitoring through VMM-based 'out-of-the-box' semantic view reconstruction," ACM Trans. Inf. Syst. Secur., vol. 13, no. 2, 2010, p. 12.

[11] G. Karjoth and M. Schunter, "A privacy policy model for enterprises," in Proc. 15th IEEE Comput. Secur. Found. Workshop, Jun. 2002, pp. 271–281.

[12] J. Jung, A. Sheth, B. Greenstein, D. Wetherall, G. Maganis, and T. Kohno, "Privacy oracle: A system for finding application leaks with black box differential testing," in Proc. 15th ACM Conf. Comput. Commun. Secur., 2008, pp. 279–288.

[13] Y. Jang, S. P. Chung, B. D. Payne, and W. Lee, "Gyrus: A framework for user-intent monitoring of text-based networked applications," in Proc. 23rd USENIX Secur. Symp., 2014, pp. 79–93.

[14] K. Xu, D. Yao, Q. Ma, and A. Crowell, "Detecting infection onset with behavior-based policies," in Proc. 5th Int. Conf. Netw. Syst. Secur., Sep. 2011, pp. 57–64.

[15] M. O. Rabin, "Fingerprinting by random polynomials," Dept. Math., Hebrew Univ. Jerusalem, Jerusalem, Israel, Tech. Rep. TR-15-81, 1981.