An Empirical study of clock skew behavior in modern mobile and hand-held devices.

Swati Sharma
Comp. Sc. & Engg. Dept.
Indian Institute of Technology,
Delhi, India
Email: sswati@cse.iitd.ernet.in

Huzur Saran
Comp. Sc. & Engg. Dept.
Indian Institute of Technology,
Delhi, India
Email: saran@cse.iitd.ernet.in

Sorav Bansal
Comp. Sc. & Engg. Dept.
Indian Institute of Technology,
Delhi, India
Email: sbansal@cse.iitd.ernet.in

***Abstract*—Host identification, today, can be done at many layers of the network protocol stack depending on the identifiable parameter used for classification. But, these generally include fields from TCP/IP/MAC packets; that can be spoofed or manipulated very easily to misguide the identification process or include intolerable error into. Identification on wireless networks can be done with better precision by using physical layer, machine-dependent characteristics. We provide an empirical study of another such parameter, the host's clock information, that may lead to accurate identification of a host, among other applications. It is resistant to the earlier mentioned methods of spoofing, as clock information is very specific to the oscillator that generates it. We provide a simplification into the measurement technique of an already investigated approach of remote identification, to achieve lower error rates. We also provide a detailed study of clock skew behavior on a LAN, consisting of wired, wireless nodes and modern mobile and hand-held devices. To our knowledge, this work is the first in the mobile and hand-held device domain to identify such devices definitively. Clock skew based host identification can be put to many applications, that may be specific to each Enterprise network. For instance, aiding the network administrator in monitoring the network, malicious activity flagging mechanism for IDS's/IPS's, isolating unknown or new machines, keeping count of the number of active machines at any time for the purpose of say IP address allocation, associating virtual machines to their corresponding physical machines and so on.**

I. Introduction & Related Work

The proliferation of mobile devices and liquidification of enterprise boundaries from the enterprise servers to personal devices, has made security in Enterprise networks questionable. To accommodate these devices, enterprise networks are shifting to wireless networks. Now, the high speed applications containing confidential business data demand absolute security for the data migration across heterogeneous device platforms. So, faultless and efficient functioning of wireless networks is of utmost importance. A network administrator has to be confident that any machine on the network is not forging the identity of multiple hosts and/or changing signatures very often and/or operating with many signatures simultaneously. A first and foremost step in this direction would be to establish the existence of every device on a network and flag any malicious activity for further scrutiny by the NIDS.

978-1-4244-8953-4/11/$26.00 2011 IEEE

Some of the existing host identification methods are sequence number based MAC-address spoof detection [2] [9], acknowledgement frame delay based identification [7], active behavioral fingerprinting [1], OS and NIC driver fingerprinting [11]. But, measures to spoof most of them exist too. Thus, there is a need for a deterministic method of unique classification for the hosts on a network. For a wireless network, one such measure is capturing the physical layer transmission signature [3] of a device under active communication and build its transmission signature that is characteristic of the electrical circuits in the device. But, proximity to the transmitting device is required in this case to be able to capture the transmitting signal from wireless device, the range of which is usually very small. An alternative to all this trouble is to extract timing information from a host and classify the clients based on it. Clocks that provide this timing information are characteristic of the oscillator (heart of the timing information in the electrical circuits).

This line of research was initiated by Paxson [10] and further worked on by Moon [6]. They demonstrated that packet delay and lost packets were taken care off in a network, but lack of synchronization between the end hosts is generally unaccounted for, thus, leading to overestimated accuracy from such measurements. They introduced algorithms to judge and remove the skews and offsets from delay measurements. While Moon worked on the removal of these clock skews from network measurements, Kohno [4] exploited this clock information to derive the microscopic deviations in the device hardware and hence identify the device uniquely.

The technical contributions of our work are threefold. Firstly, we provide a simplification of the measurement technique used by Kohno to achieve a finer precision with which a host identification could be done on a wired remote network. Secondly, we apply this technique to the wireless nodes (laptops and netbooks alike) to achieve the same accuracy level in skew calculations. This was a little tricky owing to the varying latency and errors reported in the wireless channels. Finally, we use this technique to evaluate the clock skew of modern mobile and hand-held devices (Samsung Galaxy-GT 9000 and Nokia N8 phones) and study their behavior in a network. To the best of our knowledge, this work is

the first such study done for smartphones to extracting clock information and doing precise identification for the purpose of enterprise network monitoring.

II. Background

Any fingerprinting method strives to achieve three main characteristics: Accuracy, Diversity and Stability (Spatial and Temporal).

A. Timestamp Sources

We calculate the clock skew by utilizing the ICMP and TCP timestamp information. They differ in the sources they extract time from, their propagation, detection and evasion. Timestamps are also present in the HTTP header and can be extracted to calculate the clock skew of the host. But, we do not emphasize much on this source of timing information.

B. ICMP Timestamps

Timestamps can be extracted by sending ICMP timestamp request messages (ICMP message type 13 and ICMP message code 0) to the target host, who then responds with ICMP Timestamp Reply messages. The ICMP timestamp reply message contains of three timestamps. First, **ORIGINATE** timestamp, is filled in by the measurer when the request is sent. The receiver, then, fills in **RECEIVE** timestamp when it actually receives the message and **TRANSMIT** timestamp when the packet is sent back. In the presence of a Time Synchronization software, like NTP [5], the timestamp values get updated according to the system time.

C. TCP Timestamps

A TCP flow will use the TCP timestamp option if the network stacks on both ends of the flow implement the option and if the initiator of the flow includes the option in the initial SYN packet. All modern operating system versions implement this today with the exception of Windows operating system. The timestamp request and timestamp reply fields are combined into a single packet contained in the options part of the TCP header. **TSVal** (Timestamp Value) field, contains the current value of the timestamp clock of the TCP that sends the option.

D. Clock Terminology

The nomenclature used in this paper is consistent with that used by Paxson [10] and Kohno [4].

III. Skew Extraction Methodology

Let us assume that an adversary has a **trace**, with packets containing timestamps (ICMP/TCP) captured from the network. If **T** be the number of packets in the trace, **s**, the clock skew that we have to estimate, **ti**, the time in seconds at which the measurer observes **ith** packet, **Ti**, the timestamp that was sent by the target, in the ith packet. We define the following intermediate quantities, M denotes `on measurer` and T denotes `from target`:

$$ArrivalTime(M) :: xi = ti - t1 \quad (1)$$

$$Timestamp(T) :: vi = Ti - T1 \quad (2)$$

$$InterruptTimerFrequency(T) :: Hz \quad (3)$$

$$Observedoffset(ithpacket, T) :: yi = wi - xi \quad (4)$$

$$OffsetSet :: OT = (xi, yi)|ibelongsto1, ...|T|. \quad (5)$$

IV. Experiments

The experiments we did can be sub-categorized into Desktops, laptops, netbooks and smartphones, hand-held devices.

A. Experimental Setup

The experimental setup included a measurer (in a client and server model), who captured traces to extract timestamps and calculate the clock skew for all different hosts found in the trace. The targets were Desktops, wirelessly connected laptops, Acer netbooks and smartphones. The target hosts were running Fedora core 10, 11, 12, freeBSD 5.2.1, Ubuntu version 8.04, 9.10, Windows XP (SP 2, 3), Windows Vista, Windows 7, Macintosh OS X version 10.5.8, 10.6.2 and 10.6.4, android OS version 2.1 for Samsung Galaxy GT-9000 android phones, android OS x86 for netbooks and Symbian $\hat{3}$ for Nokia N8 phones. Virtual machines also ran the same versions of Ubuntu, FreeBSD and Windows XP. The physical targets monitored were 89 in number and the virtual machines monitored were 32 in number. The duration of these traces lasted from a few minutes to several days and the measurements were taken in multiples of 1 sec, 1 min, 5 mins, 20 mins, 30 mins, 40 mins and 1 hour. Measurements were repeated after a few days and a month to make sure that the calculated clock skew values were the same as earlier and hence, consistent. We repeated the experiment when the target machines were synchronized using NTP and without NTP, when some of them were connected to the network wirelessly (in the last mile) and in the wired fashion, when the load on the machines varied (i.e. the temperature of the motherboard varied and hence the clock skew varied too) as well as when the network load varied (so as to observe the delay that may occur in timestamping the packets by the target), when the target machines were real machines or virtual machines, when the timestamp acquisition was from ICMP timestamps or TCP timestamps, when the target machine was a wired client, a wireless client or a smartphone. Extracting timestamp from the HTTP packets is also another source of time information (that offers a different timing resolution) but we limited our experimental domain to the TCP and ICMP timestamps only. TCP timestamps are captured from the TCP communication taking place between the main server and the clients (thus passive technique) while for ICMP timestamp collection the server explicitly sends ICMP timestamp request packets to the target making it an active fingerprinting technique.

B. Desktops, laptops and Netbooks

Apart from the ICMP and TCP timestamps and their time sources mentioned earlier, we also tried to capture time from other sources like jiffies variable from the kernel and the

hardware clock. But they were both not feasible for our requirement. Accessing their value included a privileged system call which was non-deterministic in terms of amount of time it would take to complete. So, we just concentrated on the ICMP and TCP timestamps. A single straight line in figure 1 illustrating the offset sets for a subset of our hosts, shows that the slope or the skew estimate would be a consistent value that will be constant over time for a single machine. Each host has an unique line suggesting unique slope, hence, a different skew value.

All skew estimates found by using ICMP as well as TCP timestamps were within a range of 0.3 to 0.4 ppm (1ppm = 1 microsecond per second). A plot showing the summarized skew for a few hosts is shown in figure 2. This is an improvement over the skew estimate (0.67 ppm for TCP timestamps) that was reported by Kohno.

There were some contrasting differences between our observations and those reported by Kohno. Firstly, the accuracy of skew estimates observed by us were better than those reported by Kohno (0.4 ppm as opposed to 0.67 ppm, respectively). Secondly, skew behavior of Virtual machines as presented by Kohno were contrastingly different from what we observed. We created 32 virtual machines which were equally spaced out on 3 physical VMWare servers. The skew estimates of these machines were not very large or inconsistent (as prostulated) but rather consistent and in the same range as the other desktops. Further, we observed that all virtual machines on a particular server reflected the skew estimates of that physical machine very closely. Such a change in Virtual machine timekeeping behavior [8] may be attributed to the latest virtualization implementations.

For the laptops and netbooks, the last mile being wireless introduced large network delays. So, to filter out this affect, we put a restriction on the percentage error allowed on the latency of the packets. Putting a bar on the packets, thus, utilized for calculation, eliminates the high delay packets, leading to a stable skew value with the same accuracy as that of Desktops. This threshold was set differently for each host from the trace. The value of this unique threshold for a particular host was set to about 200% of the average latency in a captured set. We thus did not use linear programming (used by Kohno) to pull out the anomalies from the offset set data, **OT**. rather, we just eliminated the variable network delay packets to calculate the clock skew. The rest of the observations for the laptops and netbooks were the same as the Desktops. The netbooks did not show a different skew behavior than the normal resource-abundant machines, as we had originally thought. This led us into the direction of using the same approach for modern hand-held devices and smartphones.

C. Modern Hand-helds & smartphones

We used Samsung Galaxy GT-9000 phones with android OS version-2.1, android OS x86 for netbooks and Nokia N8 phones with open source Symbian $\hat{3}$ operating system, for the purpose of our experiments. But, our first attempt to understand their network protocol stack, raised a very important concern. The smart phones are constrained in terms of resources like processing power, battery-backup and memory, etc. So, their operating systems are optimized with power saving measures. As soon as the user activity and the network activity on a smartphone becomes idle, it goes into sleep mode to conserve resources. This results in very large, unpredictable and variable delays for a server who is sending packets waiting for a quick response (ICMP Timestamp Reply packets for skew estimates). Hence, we customize our methodology for such resource-constrained devices. We first send a small train of packets to check if the phone is in busy or idle/sleep mode. If the phone is in idle mode, the small train of packets would wake it up and commence the network activity. The packets that follow would not experience unpredictable delays. But if the device is in the sleep mode, then any number of probes are unable to wake it. To filter out any additional variable delay packets that may have been left in the trace due to the wireless last mile, we send a burst of packets to the device and then wait. When the response to that burst arrives at the server, we apply the same Threshold based elimination approach mentioned earlier to acquire stable and consistent clock skew estimates.

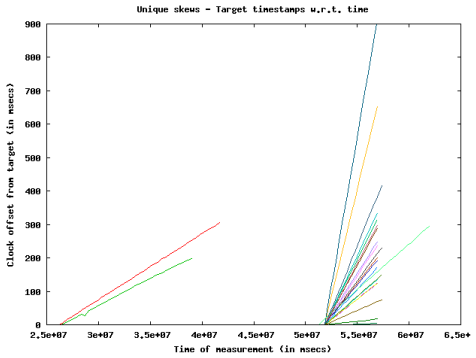

Fig. 1. Unique Slopes - Unique Skews

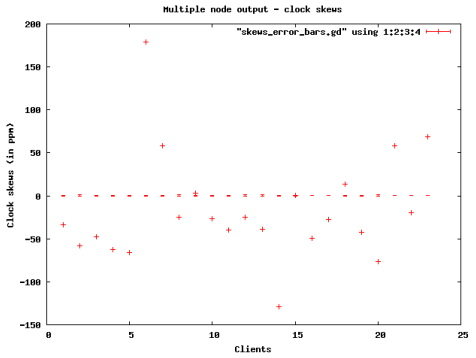

Fig. 2. Skew values with error bars

For the TCP timestamp based skew extraction, the measurer is a web server from whom the device requests a web page or small files to initiate a TCP connection. The TCP timestamp option, TSVal, is enabled by default on the android and Symbian OS. So, even a few packets are enough for the server to

extract timestamp and calculate the skew value. This becomes consistent with any further packets, transmitted between the device and server. But, there still exists uncertainty in this skew extraction technique as if the device goes into the idle mode and sleeps, without giving sufficient packets to the server, any number of packets sent in the one or more trains are unable to wake it up. Only the user having physical access to the device can wake up the wifi once it has entered the sleep mode. To remedy this situation, we build an android application that runs on the android device and requests very small information at regular intervals of time from the server. Since the transmitted packets are small, the device does not loose on important, scarce resources like power and lets the device sleep soon, to wake it up again when the desired interval is finished(to repeat the transmission).

The skew estimates for android smartphones, android OS x86 for netbooks and Symbian OS, using all the above mentioned approaches was again consistent within a range of 0.3 to 0.4 ppm. This is shown in figure 2.

- **Topology Changes** - For every smartphone, we shifted from the Wifi Network to EDGE based network. The measurer in this scenario was also multiple hops away. The skew estimates from both these versions were almost similar, hence, proving that all sorts of variability in terms of latencies have, thus, been removed.
- **Time Synchronization** - In the presence of automatic time synchronization enabled on the smartphones, their was no visible effect on the skew estimates. For the Desktops and laptops, presence of NTP, brought about a small change in skew due to continuous updation of system time by the NTP time server.
- **Trace duration** - Off all the trace durations we tried, once the threshold number of packets were collected, the duration of the capture after that did not affect the clock skew evaluation.
- **Temperature Changes** - A reboot or slight change in the machine temperature, shifted the skew range by about 0.1 to 0.3 ppm.
- **Load Changes** - Changes in the network load also failed to produce any perceptible changes in the skew estimates of the hosts. On the android phones, we used the network, made calls simultaneously. But, this did not bring about any variations in the skew estimates.
- **Stability** - The skew ranges of these hosts did not fluctuate by more than 0.1 ppm to 0.3 ppm when the measurements were repeated after a few days and even few months. This assures us that clock skews could be used as a method of identification for smartphones and handheld devices as they satisfy the desired 'Diversity and Spatial-Temporal Stability' characteristic of a fingerprint. Also, the similar values of skew when WiFi and EDGE were used as the underlying network on the smartphones, supports the Spatial Stability criteria.

V. Applications

This clock skew behavior is very distinctive of the host, so host identification is the most fundamental utilization of the skew estimates. Now, this identification could assist many motives. First of all, it could be used to identify a client in a big network trace, to help the network administrator manually or to a NIDS/NIPS to flag a malicious device behavior. Secondly, it could also be used to count the number of hosts from a network trace. Thirdly, it could be used to know the existence of vital servers on a network, for instance, the DHCP server or the DNS server or the gateway. Finally, skew behavior on virtual machines can be used to get an idea of the layout of the virtual-to-physical mapping.

VI. Conclusion

In this study, we analyzed the behavior of clock skew across different kinds of machines (Desktops, laptops, netbooks, smart phones) and operating systems (Windows, Linux, Macintosh, FreeBSD, Android, Symbian). We found out that each of these skews were unique even if the devices were of the same make and running the same operating system. The skews were constant over time and satisfied the Diversity, Spatial and Temporal Stability criteria. This can have many applications in the industry and the academia, like, becoming aware of the network around your device, or network administrator taking control of his network based on information regarding each and every device operating in it and so on. We made certain optimizations in the technique proposed by Paxson and Moon to boost its performance for wireless devices and resource-constrained devices like smartphones to achieve finer accuracy and confidence level with which each device can be chained to its unique clock skew with respect to a common measurer.

References

[1] S. Bratus, C. Cornelius, D. Kotz and D. Peebles. "Active Behavioral Fingerprinting of wireless devices. In Proc First ACM Conf. Wireless Network Sec, 2008".

[2] Fanglu Guo and Philippe Tzi cker Chiueh. "Sequence number-based mac address spoof detection. In Proceedings of 8th International Symposium on Recent Advances in Intrusion Detection, 2005".

[3] J. Hall, M. Barbeau and E. Kranakis."Detection of transient in radio frequency fingerprinting using signal phase. Wireless and Optical Communications, 2003".

[4] T. Kohno, A. Broido and KC Claffy."Remote physical device fingerprinting. IEEE transactions on Dependable and Secure Computing, 2005".

[5] D. Mills"Network Time Protocol (NTP),1985".

[6] S.B. Moon, P. Skelly and D.Towsley."Estimation and removal of Clock skew from network data measurements. IEEE Infocom, 1999".

[7] L.C.C. Desmon, C.C. Yuan, T.C. Pheng and R.S. Lee."Identifying unique devices through wireless fingerprinting. In Proceedings of the first ACM Conference on Wireless Network Security, 2008".

[8] V.M. Inc."Timekeeping in VMware virtual machines. Whitepaper, 2008".

[9] G. Lackner, U. Payer and P. Teufl."Combating Wireless LAN MAC-layer Address Spoofing with Fingerprinting Methods. International Journal of Network Security, 2009"

[10] Vern Paxson."On Caliberating measurements of packet transit times. Sigmetrics Perform. Eval. Rev., 1998".

[11] A.C. Judd, S. Reader and M.T. Shing."Improved Network Security and Disguising TCP/IP Fingerprint through Dyanamic Stack Modification, 2005".