Enabling Reproducible Cyber Research - Four Labeled Datasets

Tom Bowen[1], Alex Poylisher[1], Constantin Serban[1], Ritu Chadha[1], Cho-Yu Jason Chiang[1], and Lisa M. Marvel[2]

[1]Vencore Labs d/b/a/Applied Communication Sciences
Basking Ridge, NJ 07920

[2] U.S. Army Research Laboratory
APG, MD 21005

Abstract— **In this paper, we describe the design and creation of four publicly available datasets generated using a testbed with simulated benign users and a manual attacker. The datasets were created to provide examples of cyber exploitations and aid in the production of reproducible research that address cyber security challenges. The CyberVAN testbed provides sophisticated capabilities for high-fidelity cyber experimentation in strategic and tactical network environments. The representative network is sufficiently complex with synthetic users performing normal duties that generate traffic (webpage browsing, e-mail, etc.). Both network and host based facts/logs are included in the dataset along with a diagram of the network and a timeline of events. The four datasets encompass progressively complex scenarios: 1) malware infection injection via a phishing email attachment; 2) propagating botnet injection via phishing email attachment with a Single Fast Flux algorithm for bot master identification/communication; 3) propagating botnet injection via email link using a Domain Generation Algorithm for bot master identification/communication; 4) propagating botnet injection via corruption of a legitimate internal web server with Double Fast Flux for bot master identification/communication. The full datasets along with relevant documentation is available for public download. Additional datasets containing tactical network scenarios and environments will be added to the repository in the future with the goal of enabling reproducible cyber security research that will advance the science of cyber security.**

Keywords— Cyber Security, Computer Network Defense, Computer Network Operations

I. INTRODUCTION AND BACKGROUND

The importance of advancing the Army capabilities in cyber operations cannot be overemphasized. Dominance in this domain is vital to mission success on the battlefield now and in the future. To facilitate this endeavor, researchers are conducting basic and applied research exploring a variety of models and methods to enhance and enable cyber capabilities. However, a significant deficiency to forging ahead is the lack of well documented, realistic, complete datasets that exhibit currently relevant phenomenon. A methodology for constructing complete datasets will enable reproducible research results and reliable comparison with future models and techniques.

Currently cyber researchers may use self-generated datasets to validate their model or methodology. They may not publish adequate details of their datasets or make them available to other researchers to validate reported results and compare them to those from using competing techniques. Additionally, cyber datasets (consisting of host and network facts) collected from real networks may contain information that is private or restricted (e.g., passwords, PII, financial information, etc.). Most dataset owners are unwilling to release their datasets because the datasets may contain privacy-leaking information. It is well known that anonymization of this type of datasets is problematic [1]. Therefore, when datasets are released they may contain only partial information (e.g., flows/headers only).

An additional challenge to using real network/host data captures is the lack of metadata, such as those used for labeling traffic flows as benign or malicious. Providing metadata requires significant effort by analysts and the metadata may not be adequate. Many have attempted to utilize cyber exercise data as research datasets (CDX 2009 [2], iCTF [3]). However, detailed, accurate red team logs are rarely provided and ground truth cannot be ascertained without them. Ground truth is critical for research into intrusion detection and cyber security. Additionally, most cyber exercise data contain a disproportionate amount of attacks by nature and may not be indicative of typical networks. Currently, publicly available labeled datasets are dated (DARPA 1998-2000 [4]). For years these datasets were used because they were the ones with labels. As with any detection problem the labels of true positive and true negatives are essential to measure the performance of any detectors that are designed to categorize the events of interest. In 2014, Koch, et al, compared numerous public datasets in [5]. The only datasets with established ground truth were derived from DARPA-99 with only flow records or packet headers [6,7].

The work reported in this document/presentation was partially performed in connection with contract number W911NF-14-D-0006 with the U.S. Army Research Laboratory. The views and conclusions contained in this document are those of the authors and should not be interpreted as presenting the official policies or position, either expressed or implied, of the U.S. Army Research Laboratory, or the U.S. Government unless so designated by other authorized documents. Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

Cyber security datasets should be constructed using well thought out realistic scenarios. All assumptions should be clearly described. The actual datasets should consist of observations on the host- and network-related facts (traffic logs, system logs, firewall logs, etc.). A dataset should consist of all of these fact types collected during the malicious activities. Additionally, all facts should be labeled as benign or malicious to provide ground truth.

In this paper, we describe a data collection methodology using simulated users and a manual attacker on a testbed. This process alleviates the need to anonymize the data. Additionally, all details of the attacker's actions are known, enabling complete ground truth in data labeling. Realism of the datasets is achieved using the high-fidelity CyberVAN testbed [10].

The collection of these datasets is the intial step to enable the generation of reproducible results for cyber security experiments. The value of our approach is that attack scenarios are repeatable and the set of observations contained in the datasets are beneficial for future research efforts. The dynamic nature of cyber systems and their exploits requires not just offline analysis but also online examination. Online detection and reaction tools can be incorporated into the CyberVAN testbed to evaluate their capabilities in real time. As researchers suggest additional observations that are collectable through either standard utilities or newly developed tools, they can be incorporated into the testbed and the scenarios can be re-run to produce datasets with additional observations. Likewise, new scenarios can be added to the testbed to study different types of attacks. This process will be enhanced by human-in-the-loop experimentation, which will be added to CyberVAN in the near future.

The remainder of the paper is organized as follows. In Section II we first describe the CyberVan testbed used to generate the datasets. Then in Section III we provide a description of network and host configurations and the overall data collection process. This is followed in Section IV with specific details of the four event scenarios. Finally, we conclude the paper in Section V with a summary of our effort and a plan for future updates and extension.

II. CyberVAN Basics

In this section, we describe the CyberVAN (Cyber Virtual Assured Networks) testbed [10]. CyberVAN is a mature cyber experimentation testbed maintained by ACS and developed with funding from ARL, OSD, and US Army CERDEC [8][8]. CyberVAN provides sophisticated capabilities for high-fidelity cyber experimentation in strategic and tactical network environments. It enables arbitrary applications running on testbed nodes to communicate transparently via a simulated network implemented in a network simulator such as OPNET, QualNet, ns-2, or ns-3. Testbed nodes can be provisioned as Xen-based virtual machines, Linux containers, or simply "simulated hosts" whose behavior is governed by processes directly attached to the network simulation. This provides great flexibility for experiment designers to explore a wide spectrum of network topologies, sizes, and host fidelity. CyberVAN supports the use of a wide range of test scenarios covering enterprise, strategic, and tactical networks in the wired and wireless domains. It has been used for ongoing and recent DoD research programs including ARL Cyber Security Applied Research and Experimentation Partner (AREP), DARPA Wireless Network Defense (WND), and U.S. Army CERDEC CRUSHPROOF [9].

III. Dataset Descriptions

In this section we describe the host and network setup for the scenarios.

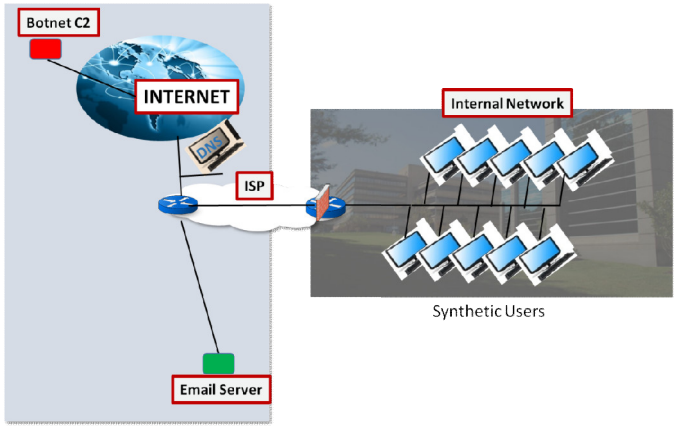

Figure 1. Cyber VAN Enterprise testbed

Figure 1 shows the notional topology of the testbed used for collecting the datasets. The internal network consists of workstations, and server nodes, and a firewall node. The work stations have outgoing Internet access and are used by synthetic users to perform email, ftp, and web browsing tasks that access both internal and external servers. The internal server nodes provide internal web sites and internal name service. The internal server nodes have outgoing Internet access and limited incoming Internet access: the internal web servers allow external access for web services, the internal name server has bi-directional connection to the public name server. The firewall node is located between the internal and public networks. The public network consists of a set of attacker nodes and a synthetic internet, which is a single node that simulates thousands of public Internet nodes including web sites, ftp sites, name servers, and email servers.

All of the datasets consist of an attack free period of at least one hour followed by a period in which an attacker, using an attack node in the public Internet, launches an attack against the internal nodes for the purpose of establishing a botnet that infects the majority of the internal nodes. The botnet used is synthetic, however it contains typical real world botnet features such as using stealth and migration to hide the location of the bot master and rendezvous points, using a process hiding rootkit for persistence on victims, and supporting communication between bot master and victims for executing commands and reporting status.

The datasets where collected in the context of 4 scenarios, and the scenarios differ in the methods used by the attacker to obtain initial entry in the internal network, and the characteristics under which the botnet functions once it is

implanted. The entry methods all require user assistance ranging from tricking a naïve user into executing an attachment in a spoofed email purporting to be from a colleague, to tricking a user into clicking a link on a frequently used internal web site that the attacker has corrupted. The varied botnet characteristics include how the victims discover and communicate with the bot master. Initial attack entry and subsequent propagation within the internal network exploit both real-world and synthetic vulnerabilities. The attacker is presumed to have some prior knowledge of the internal network, such as the email addresses of internal users, the IP address ranges of the internal network, and a rough idea of the types of internal web services offered. The attacker expands this knowledge using *nmap* [11] and *sqlmap* [12] utilities.

Observations are collected from the internal nodes and contain the following information:

- Network packets as captured by tcpdump
- File writes as captured by sysdig
- Pre and post tarballs of node state as shown in the /proc file system, and
- Pre and post tarballs of logfiles from /var/log

The following descriptions of the scenarios under which observations were collected describe at a high level the malicious activity, and how examples of that malicious activity appear in the dataset as shown by sample snippets of representative observations. The sample observations reported here cover only the malicious behavior visible in the datasets; however, the bulk of the datasets captures the benign activity from which a detection system would need to cull the malicious behavior. Page limit of this paper permits the report of only one representative observation for each scenario; however, the online datasets are accompanied by documentation with much more detailed, though not exhaustive, descriptions of benign and malicious behaviors manifest in the datasets.

A. Malware Infection via Phishing Email Attachment

For our first scenario we captured observations during an attack in which malware was implanted on a victim node through a naïve email phishing method, specifically, the attacker spoofed an email to a victim which appeared to be sent from a colleague. The spoofed email contained the malware as attachments, and the body of the email instructed the victim to download the attachments and execute one of them. The malware to be executed (called *mc*) was the client-side of the synthetic botnet application. The other attachments were supporting files. For scenario 1, *mc* was configured to be mostly dormant, that is, to not try to contact the bot master, and to not perform reconnaissance and propagation to other victims.

A sample file write observation captured when the victim was tricked into downloading the malware attachments in the spoofed email, is shown below:

```
13:26:49.824160725 thunderbird(9462) W
  4.00KB /tmp/mc.64
```

```
ELF>@@@8@@@@@@88@8@@@)) ..a.aU{
  ..a.aTT@T@DDPtdAAQtdRtd..a.a/lib64/ld
  -linux-x86-64.so.2GNU
  GNU\":h$_d[[\\)9?l[]U q>-hMNSH $ak
13:26:49.824174764 thunderbird(9462) W
  4.00KB /tmp/mc.64
0a 0a(0a00a80a@0aH0aP0aX0a
`0ah0ap0a
```

Only the first few lines are reported. This entry shows that thunderbird writes to the file /tmp/mc.64, which is the name under which botClient is implanted. The observation records file content, and since the file is a binary executable, attempts to render the contents in readable form. Most of the record is illegible; however, the string "ELF" is clearly visible on the first line of the write buffer. An ELF executable contains the string in their first line, so it is easy to deduce the download of an executable file.

B. Propagating Botnet with a Single Fast Flux Algorithm

For our second scenario we extended the first scenario by configuring the downloaded executable *mc* to both attempt to communicate with the bot master (being run by the attacker on an external host) and to perform reconnaissance to discover other victims and to propagate to those victims.

Communication to the bot mas ter used a Single Fast Flux (SFF) algorithm [13], in which the bot master registers the IP address of a rendezvous point node under a well-known name (*rpName*) in a public name server. The *mc* malware queries the name server for the IP address of the well-known name. If the name resolution is successful, it communicates with the bot master. Periodically the bot master migrates the rendezvous point to a different node and updates the public name server. Infected nodes with Internet access can resolve *rpName* and directly communicate with the rendezvous point. When the bot master migrates the rendezvous point, communication failure prompts *mc* to repeat the rendezvous point discovery algorithm. Infected nodes without Internet access are unable to resolve and/or connect to the rendezvous point. In which case, they communicate with the bot master through the node that infected them.

Reconnaissance to find other victims consists of using the *nmap* utility to discover other nodes that have one of two vulnerabilities, *p2paa* and/or *shellshock* [14]. *P2paa* is a vulnerability in a synthetic network daemon process running on each internal node. Specially constructed packets trigger built- in vulnerabilities in p2paa that are exploited to download and execute files. Shellshock is a bash vulnerability that, in conjunction with mis-configuration of a user account, allows file download and execution without authorization. The *nmap* scanning for the two vulnerabilities consists of finding nodes that have either port 22 (*ssh*) or 12345 (*p2paa*) open. When a vulnerability is found, it is exploited to propagate the malware on the current victim to the next victim. As the malware propagates, it is edited to update lists of already found victims to avoid redundant infection attempts.

While the botnet formed by the propagating malware allows the bot master to send arbitrary shell commands to the

victims, for scenario 2, botnet communication was limited to periodic status messages from the victims.

Observations associated with some of the malicious activity in scenario 2 can be seen in the dataset as the following examples show.

When the implanted malware runs *nmap* for reconnaissance, *nmap*'s stderr is redirected to /dev/null, and stdout is redirected to /tmp/out. Sysdig captures these file write observations as shown below.

```
16:09:48.374702889 nmap(28872) W 78B
  /dev/null
Starting Nmap 4.11 (
  http://www.insecure.org/nmap/ ) at
  2016-03-19 16:09 EDT

16:09:48.374778684 nmap(28872) W 149B
  /tmp/out
# Nmap 4.11 scan initiated Sat Mar 19
  16:09:48 2016 as: /usr/bin/nmap -n --
  send-ip --excludefile
  /tmp/excludeList -oG out -p 22,12345
  10.10.1.1-255
```

Only a fragment of the entry is repeated here.

C. *Propagating botnet via email link to a malicious website using a Domain Generation Algorithm for bot master identification*

For our third scenario we changed three aspects of Scenario 2. First, rather than using spoofed email in which malware is contained in the attachment, we used a spoofed email that contained a link to a malicious web site. Second, the *mc* malware propagated by the attack was configured to use the Domain Generation Algorithm (DGA) [13], instead of SFF, to allow victims to find the rendezvous point for communication with the bot master. Third, we configured *mc* to prefer communication through infecting node rather than over the public Internet.

The malicious web site is a publically accessible web site under the attacker's control (via unspecified attacks). This web site contains an html file (ffx.html) which exploits a vulnerability[1] in Firefox browsers. The vulnerability allows the website to inject an arbitrary javascript into the victim's browser and execute them. The injected javascript provides for back connection to the ffxServer process that the attacker is running on the web site server, receipt of commands from ffxServer, and execution of those commands via a shell interpreter, thus providing for the execution of arbitrary shell commands issued by the attacker on the victim node. The shell commands executed for this attack consist of *scp* commands to download the botnet malware from the web site (where the attacker has pre-positioned it) to the victim, and then execute that malware.

The malware downloaded to the victim executes exactly as in scenario 3 with two exceptions: 1) it uses DGA to find rendezvous points for communication with the bot master, and 2) it communicates through the infecting peer, if possible. With DGA, the bot master calculates a name for the rendezvous point using a wall clock time based algorithm and registers that name and the rendezvous point IP address with the public name server. The victims execute the same name calculation algorithm; however, they do multiple iterations using a time window around the current wall clock time. The victims try each name until one resolves and then use that to communicate with the bot master. Periodically the bot master migrates the rendezvous point, calculates a new name and registers that name and a new IP address. Migration causes communication failures that trigger DGA rediscovery. Communicating through the infecting peer simply means that instead of communicating through infecting peer as a fallback when Internet access is not available (as in scenario 2), a victim will always communicate through the node that infected it, if there is such a node. All victims except the first victim will have an infecting node, so only the first victim will execute the DGA and communicate over the public Internet to the bot master.

An example of observations associated with the malicious activity in scenario 3 that can be seen in the dataset include entries in the tcpdump file showing the interaction of the injected javascript with the malicious web site. These interactions occur over port 4444, and involve the web site sending commands to the javascript for execution. A few of these entries are shown below:

```
13:14:15.009377 IP 180.42.143.70.4444
  > 10.10.1.2.45043: P 1:162(161) ack 1
  win 227 <
nop,nop,timestamp 219637706 727203156>
E.....@.>.e..*.F

...\..bh..k0.......g.....
^M.g.+X=Tscp  -o
  "PasswordAuthentication=no" -o
  ConnectTimeout=10 -o
  "StrictHostKeyChecking=no"
  root@180.42.143.70:/tmp/mc.config
  /tmp/mc.config.180.42.143.70&&echo
  DONE

13:14:15.009391 IP 10.10.1.2.45043 >
  180.42.143.70.4444: . ack 162 win 237
  <nop,nop,timestamp 727203162
  219637706>
E..4.H@.@.+.
```

The above entry shows transmission of an scp command from malicious web site port 4444 to the injected javascript on i-dc-1 that downloads one of the malware files.

```
13:14:17.361500 IP 180.42.143.70.4444 >
  10.10.1.2.45043: P 798:832(34) ack 26 win 227
 <nop,nop,timestamp 219640050 727205501>
E..V..@.>.fI.*.F

...\..bh.9k0.......p.....
```

[1] https://community.rapid7.com/community/metasploit/blog/2015/03/23/r7-2015-04-disclosure-mozilla-firefox-proxy-prototype-rce-cve-2014-8636

```
^M.p.+XF}cd /tmp && ./mc.64.180.42.143.70&

13:14:17.361506 IP 10.10.1.2.45043 >
  180.42.143.70.4444: . ack 833 win 270
  <nop,nop,timestamp 727205513 219640050>
E..4.S@.@.+.

...*.F...\k0..bh.\....N.....
+XF.^M.p.
```

The above entry shows transmissions of a command from malicious web site port 444 to the injected javascript on i-dc-1 in Figure 1 that executes the previously downloaded malware.

D. *Scenario 4: Propagating botnet injection via corruption of a legitimate internal web server with a Double Fast Flux for bot master identification/communication*

For our last scenario, instead of tricking a user into visiting a malicious web site through an email, the attacker corrupts an otherwise legitimate internal web site typically visited by the victim such that the visitor is liable to exploitation and subsequent infection. Once the infection occurs, the *mc* malware executes on the victim exactly as in scenario 3 with the exception of using the Double Fast Flux (DFF) algorithm for finding the rendezvous point.

Two variants of Scenario 4 are provided. The variants comprise two different methods that the attacker uses to corrupt the internal web site.

In the first variant, known as *LAMP* (Linux, Apache, MySQL, PHP), the attacker exploits a SQL injection vulnerability in a postgres database used by the *SMF* (Simple Machines Forum) web site. The exploit allows download and execution of malware on the DB server node used by the SMF website. From the DB node, an additional attack is launched against the SMF website to corrupt the web pages on that node so that visitors are directed to the malicious ffx.html web page, which functions as in scenario 3 to infect the visitor. Essential to the corruption of the *SMF* website from the DB node is local-to-root escalation, since the sql injection attack occurs in the context of the *postgres*, not root, user. The escalation occurs through exploit of a known vulnerability [2] in the *delegate* utility which allows an unprivileged user to create files in arbitrary directories without regard to permissions. This vulnerability is exploited to place a file in the */etc/cron.hourly* directory such that on the hour the file will execute as a shell script run by user root. The file contains commands for downloading malware to the SMF web site, and providing the attacker with a root web shell to the SMF web site. Using the root shell, the attacker executes shell commands to install a malicious ffx web page and server program.

In the second variant of scenario 4, known as *node.js* [15], a similar strategy is used, however the mechanics are different. The nodejs variant uses a node.js web server called NodeGoat[3] to serve a retirement planning web site. The NodeGoat web server is vulnerable to javascript injection, due to poor input checking. Javascript can be injected and executed by entering it into certain input fields on the web page. The attacker injects javascript that downloads and executes malicious ffx web page and server program on the web server node. Unlike the LAMP variant the node.js web server is running as root, so privilege escalation is not required.

Visitors to either the SMF or the retirement planning web site are susceptible to infection if they follow the web site's instructions to click on a link to see the new version of the web page. The infection downloads and executes the *mc* malware which proceeds exactly as in scenario 3 except that it uses the DFF algorithm for rendezvous point discover.

The DFF algorithm operates as follows. The bot master maintains a set of private name servers and registers the current rendezvous point IP address under the name *rpName* on one of these name servers. Then the bot master registers the IP address of the private name server under the well-known name dnsName, with a public name server. Victims query the public name server for dnsName, and then query the name server on the resolved IP address for *rpName*. Periodically the bot master migrates the rendezvous point, and/or changes the private name server that can resolve the rendezvous point, updating the private and public name servers as needed. Migration of rendezvous point results in communication failures that trigger victims to execute the discovery algorithm again.

Example observations associated with the malicious activity in scenario 4 that can be seen in the dataset is presented below. The example shows one of the network packets used to convey the SQL injection attack that places a malicious file (mc.64) onto the node running the vulnerable *postgres* database.

```
2016-04-16 07:40:39.927163 IP
  10.10.1.13.38271 >
  10.10.1.14.postgres: P 4409:5098(689)
  ack 2622 win 282 <nop,nop,timestamp
  100343792 102126610>
E....B@.@.f..^M.....8r$....Z.....!....
  ........T.Q....
                                  INSERT
  INTO smf_sessions("session_id",
  "data", "last_update")
                                  VALUES

  ('eu7dkscmr185eigs55ki1dciv3',
  'session_value|s:32:"e338b1f1a41ae751
  43b4fd5722492b01";session_var|s:8:"dd
  d5d19e";mc|a:7:{s:4:"time";i:14608068
  39;s:2:"id";i:0;s:2:"gq";s:3:"0=1";s:
  2:"bq";s:3:"0=1";s:2:"ap";a:0:{}s:2:"
  mb";a:0:{}s:2:"mq";s:3:"0=1";}ban|a:5
  :{s:12:"last_checked";i:1460806839;s:
  9:"id_member";i:0;s:2:"ip";s:12:"14.1
  02.33.93";s:3:"ip2";s:12:"14.102.33.9
  3";s:5:"email";s:0:"";}log_time|i:146
  0806839;timeOnlineUpdated|i:146080683
  9;old_url|s:94:"http://192.168.2.1/fo
```

[2] http://www.vapidlabs.com/advisory.php?v=159
[3] https://github.com/OWASP/NodeGoat

```
rum/index.php?inline=select+lo_export
%28411348%2C+%27%2Ftmp%2Fmc.64%27%29"
;USER_AGENT|s:17:"Python-
urllib/3.3";', 1460806839)
```

IV. DATASET ACCESS

The datasets described in this paper are available for download at the ACS CyberVAN website, https://cybervan.appcomsci.com:9000.

V. CONCLUSION AND FUTURE WORK

We have presented a data collection methodology using a testbed and simulated users and a recorded manual attacker. This process alleviates the need to anonymize the data, provides ground truth labeling, as well as both host and network facts. Realism is enabled using the advances in the high-fidelity CyberVAN testbed.

These datasets are the initial step to enable reproducible research for cyber security experiments. The value of our approach is that attack scenarios are repeatable and the set of observations contained in the datasets are the starting point in this ongoing effort. The dynamic nature of cyber systems and exploits requires not just offline analysis but also online experimentation as well as nearly constructed datasets incorporating contemporary hosts, networks and attacks.

At this time we considered only strategic networks. In the future, we will expand this to include tactical network nodes and communication with relevant attack scenarios.

ACKNOWLEDGMENT

The effort described in this article was partially sponsored by the U.S. Army Research Laboratory Cyber Security Applied Research Experimental Partner under Contract Number W911NF-14-D-0006. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation hereon. Finally, we would like to thank Skaion for providing the Synthetic Users and Synthetic Internet tools to support this work.

VI. REFERENCES

[1] A. Narayanan and V. Shmatikov, "Robust De-anonymization of Large Sparse Datasets," IEEE Symposium on Security and Privacy 2008.

[2] B. Sangster, T. OConnor, T. Cook, R. Fanelli, E. Dean, W. Adams, C. Morrell, G. Conti; "Toward Instrumenting Network Warfare Competitions to Generate Labeled Datasets," 2nd Workshop on Cyber Security Experimentation and Test, Aug. 2009.

[3] Giovanni Vigna, Kevin Borgolte, Jacopo Corbetta, Adam Doupe, Yanick Fratantonio, Luca Invernizzi, Dhilung Kirat, Yan Shoshitaishvili, "Ten Years of iCTF: The Good, The Bad, and The Ugly," in Proceedings of the USENIX Summit on Gaming, Games and Gamification in Security Education (3GSE), San Diego, CA, August 2014.

[4] Lippmann, R., et al., "The 1999 DARPA Off-Line Intrusion Detection Evaluation," Computer Networks, 34(4), 2000.

[5] R. Koch, M. Golling, G. D.Rodosek, "Towards Comparability of Intrusion Detection Systems: New Data Sets," TERENA Networking Conference (TNC), 19-22 May 2014, Dublin, Ireland.

[6] S. Garcia, Stratosphere IPS, Malware Capture Facility Project, https://stratosphereips.org/category/dataset.html (access April 2016).

[7] NETRESEC Publicly available PCAP files, www.netresec.com, Sweden, (accessed April 2016).

[8] C. Jason Chiang, Alex Poylisher, Yitzchak Gottlieb and Constantin Serban. "Cyber Testing Tools and Methodologies." 30th Annual International Test and Evaluation Symposium, 2013, Washington, DC.

[9] Beitzel, et. al, "Recent Results Using CyDeF to Defend Large-Scale Simulated Tactical Wireless Networks", Technical Report delivered to U.S. Army CERDEC, 2014.

[10] Ritu Chadha, Tom Bowen, Cho-Yu J. Chiang, Yitzchak Gottlieb, Alex Poylisher, Angelo Sapello, Constantin Serban, Shridatt Sugrim and Gary Walther, "CyberVAN: A Cyber Security Virtual Assured Network Testbed", MILCOM 2016.

[11] Nmap. https://nmap.org/

[12] Sqlmap. http://sqlmap.org/

[13] T Holz, C. Gorecki, K. Rieck, F. Freiling, "Measuring and Detecting Fast-Flux Service Networks", NDSS 2008.

[14] National Vulnerability Database, CVE-2014-6271.

[15] Node.js, https://nodejs.org/