Leveraging Controlled Information Sharing for Botnet Activity Detection

Calvin Ardi
USC/Information Sciences Institute
calvin@isi.edu

John Heidemann
USC/Information Sciences Institute
johnh@isi.edu

ABSTRACT

Today's malware often relies on DNS to enable communication with command-and-control (C&C). As defenses that block C&C traffic improve, malware use sophisticated techniques to hide this traffic, including "fast flux" names and Domain-Generation Algorithms (DGAs). Detecting this kind of activity requires analysis of DNS queries in network traffic, yet these signals are sparse. As bot countermeasures grow in sophistication, detecting these signals increasingly requires the synthesis of information from multiple sites. Yet *sharing security information across organizational boundaries* to date has been infrequent and ad hoc because of unknown risks and uncertain benefits. In this paper, we take steps towards formalizing cross-site information sharing and quantifying the benefits of data sharing. We use a case study on DGA-based botnet detection to evaluate how sharing cybersecurity data can improve detection sensitivity and allow the discovery of malicious activity with greater precision.

CCS CONCEPTS

• **Security and privacy** → **Intrusion/anomaly detection and malware mitigation**; **Network security**; *Access control*; Information accountability and usage control; • **Information systems** → *Information retrieval query processing*; • **Applied computing** → *Network forensics*;

KEYWORDS

botnet detection, cybersecurity, data sharing, information sharing

ACM Reference Format:
Calvin Ardi and John Heidemann. 2018. Leveraging Controlled Information Sharing for Botnet Activity Detection. In *WTMC '18: Workshop on Traffic Measurements for Cybersecurity, August 20, 2018, Budapest, Hungary*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3229598.3229602

1 INTRODUCTION

Cybersecurity incidents continue to increase in size, with highly damaging economic and increasingly physical consequences. The consequences of these incidents include an enormous loss of private data on individuals (Anthem [1], OPM [8], Yahoo [10]) and corporations (Sony [5]), and the money spent cleaning up. Not limited to "simple" data loss, the damages are growing past the physical boundary, affecting critical infrastructure, from industrial systems (Stuxnet [12]), hospitals (ransomware [7]), and to our own homes (IoT malware, phish).

For organizations to improve and maintain their cybersecurity posture, they need to share data across and within organizations during the incident response process. Data and working processes are distributed and independent across and within organizations: each organization has its own unique and incomplete view of the Internet or local network. Data sharing during and after a security incident helps expedite the incident response process by collectively increasing the global knowledge and corresponding effort against an attack. The increased, shared knowledge and resulting collaboration helps lead to *forward progress*, defined as advances in research and understanding, in improved network security.

Data sharing today is difficult as many organizations share limited or no information *across* other organizations for several reasons. Organizations might not share their data because it contains highly sensitive and private information (competitive intelligence, proprietary data). Organizations also sometimes cannot share (if prohibited by law), or choose not to share due to fear, uncertainty, and doubt in the risks of data disclosure.

Even *within* an organization, different parts of the organization are often discouraged or prevented from sharing. Groups might be segmented in order to maintain independence and prevent conflicts of interest (for example, a logical "firewall" between investment groups to prevent insider trading, or the editorial and advertising groups of a publication), and establish security (accounting and IT have little to no visibility in the other's systems).

Organizations that share data across other organizations and within their own will accelerate progress in cybersecurity. Sharing across different organizations enables them to solve problems that are inherently distributed (stepping-stone attacks across many network boundaries), as each organization contributes a different view of the Internet. These benefits also apply to sharing within different parts of large organizations.

Our first contribution is to provide the controlled, cross-site data sharing mechanisms in Retro-Future (§3), a system that provides retrospective, post-event understanding with time travel. Using a query/response system for cross-site data sharing, we minimize the risks in data disclosure for the querier and responder, allowing both to manage the risks inherent in data sharing.

Our second contribution is to quantify the benefits of sharing using our controlled, cross-site data sharing mechanisms in Retro-Future through a case study (§4) in detecting botnet activity at USC/ISI, Los Alamos National Laboratory (LANL), and Colorado State University (CSU). We show that sharing improves diversity in network data, allowing us to detect bots in secure networks

Publication rights licensed to ACM. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of the United States government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.
WTMC '18, August 20, 2018, Budapest, Hungary
© 2018 Copyright held by the owner/author(s). Publication rights licensed to the Association for Computing Machinery.
ACM ISBN 978-1-4503-5910-8/18/08…$15.00
https://doi.org/10.1145/3229598.3229602

where few bots exist and show how sharing improves our detection algorithms' sensitivity, allowing us to detect botnet activity with greater precision. In addition to this case study, we are working on using Retro-Future with other applications like detecting malicious activity with DNS backscatter, and improving the security incident response process.

Our mechanisms shift the risk-benefit trade-offs in data sharing, showing that for this application, sharing makes sense. The Retro-Future framework and tools are open-sourced and are available online at https://ant.isi.edu/retrofuture. Our hope is that this example and these tools will promote broader sharing of security-related data in other applications.

2 PROBLEM STATEMENT

We next discuss why botnet activity detection without data sharing at individual sites is insufficient and how sharing data with Retro-Future addresses those insufficiencies. We also address Retro-Future's threat model with respects to sharing data, and how Retro-Future counters these threats.

2.1 Detection Sensitivity

Today, organizations can use software like BotDigger [20] to detect bot activity on a host by examining their DNS traffic for DNS access patterns that indicate botnet C&C traffic.

BotDigger works well for large, university-sized organizations (like Colorado State University, where it originated). However, it becomes much less sensitive for organizations that are not as large and diverse as CSU. Several kinds of diversity are relevant: in population and in richness. For example, diversity in population includes the volume of network traffic, number of users, applications, and hosts. Diversity in richness can include the number of network protocols, operating systems, and user types (sysadmins, casual users, etc.). For example, DNS captured might not reveal much suspect activity at USC/ISI since it has a low population (158 hosts) and low diversity (mostly Linux-based hosts). DNS captured at a larger campus like CSU will have both high population (20k hosts) and high diversity (Windows, Mac OS X, Linux) due to its relatively open and permissive network. Secure organizations like LANL (26k hosts) or those involved in banking and finance might have high population and low richness, because their network is relatively more locked down than others, restricting the types and amounts of permissible network traffic.

Our insight is that cross-site sharing with Retro-Future allows larger, more diverse organizations (CSU) to share sensitive data securely with smaller or less diverse organizations (USC/ISI and LANL, respectively) to help them detect malicious activity in their networks (§4.1). This sharing can also benefit the larger organization (§4.2 and §4.3). CSU can help because it has a much higher chance of malicious activity happening on its network: CSU (20k hosts running Windows/Mac OS X/Linux) has greater diversity in population and richness compared to USC/ISI (158 hosts, mostly Linux) and LANL (26k hosts).

Data collection and sharing with other, outside organizations poses privacy and legal risks. Collecting DNS and corresponding botnet activity (with BotDigger) data is privacy-sensitive since the data can include false positives and unvetted activity, and data collection involves IPs of end users. While each site will have its own specific sharing policy, in our case for CSU to be comfortable sharing this data with its collaborators, the data needs to be secured in transit and in use, and certain details (end-user IPs) must be omitted or minimized.

Retro-Future addresses these concerns about data sensitivity by providing the access controls and query system needed to securely share this data. With access controls, data is shared only with authorized users—the original, raw data remains at the origin site. The query system (combined with access controls) enables each site to selectively choose which details about the data are shareable based on their risk tolerance: in the following case study, end-user IPs are not shared by CSU and cannot be queried for. With the Retro-Future system in place, CSU is then comfortable with the controlled sharing of botnet when handled by Retro-Future.

2.2 Threat Model

Retro-Future's threat model (summarized in Table 1) looks at the primary threats to sharing data (via query-response) and their countermeasures when data is at rest, in motion, and in use. Retro-Future handles data in three stages in response to a query: we first pull data from archives ("at rest"), process and manipulate the data ("in use"), then transmit the results to the client ("in motion").

While some threats are general to any distributed system, collection and sharing of sensitive data with Retro-Future introduces particular risks that must be carefully addressed (shown by the '*' symbol in Table 1). Retro-Future is designed to address each of these threats and minimize these risks. For example, an active data breach (via many intrusive queries over a short period of time) can be mitigated with restricted query languages and "privacy budgets" which limit the amount and execution time of remote queries.

If the threats to data sharing are not properly mitigated, they can lead to unintended data disclosure or misuse, with further financial or physical consequences.

We also assume that the systems external to Retro-Future (to include the hardware and operating systems which Retro-Future operates and network data live on) are reasonably secured against general external and internal threats (for example, using NIST's framework [15]). For example, while we design Retro-Future to be robust to an eavesdropper on the network, we assume that the underlying Retro-Future system is trustworthy and has not been compromised by a malicious actor.

3 CONTROLLED SHARING FOR BOTNET ACTIVITY DETECTION

We next describe our approach and design decisions on how we enable controlled information sharing with cross-site queries.

The goal of controlled cross-site information sharing is to share usable data while balancing the privacy and exposure risks (§2.2) between query and response.

We achieve this goal in Retro-Future with the principled risk and privacy management needed to share data safely through trust and sharing policies and query management. Owners first set and modify accesses, even granting temporary escalated privileges, based on their trust relationships and sharing policies with other organizations (described in [9]). Queries to data are then *moderated* (§3.1)

Table 1: Threat Model Summary ('*' indicate threats introduced or caused by Retro-Future)

Data ...	Threat	Actor	Countermeasure
at rest	unauthorized access*	internal, external	secured data archive with data encryption and minimization (§3.3)
	abused access	internal	strong user authentication and authorization, data federation (§3.3)
in use	active data breach*	internal	ACLs, restricted query languages, query logs and audits, privacy and execution budgets (§3.1, §3.2)
	passive data leaks*	internal	remote and moderated queries, data anonymization and redaction, differential privacy (§3.1, §3.2)
in motion	eavesdropping	external	secure communication protocols (§3.3)
	wrong endpoint*	internal, external	public-key authentication, certificate/public-key pinning (§3.3)

such that more-specific queries are more likely to be answered than broad queries, and queries are always remotely processed to *control disclosure* (§3.2),

We emphasize owners' full control of the system and data using best common practices in system security (§3.3) while maintaining flexibility over data accessibility.

Finally, we note that our technical mechanisms complement legal or policy mechanisms in making data sharing possible. It is difficult for technical mechanisms to stop all information flow (consider side-channel attacks, for example), so policies may limit sharing and support auditing; these may be supported by formal or legal agreements. The broader legal and policy mechanisms are outside the scope of this paper.

3.1 Moderating Queries

Retro-Future enables cross-site information sharing with others using a query/response system as its foundation and additional risk and privacy management on top of the queries. After a remote client has been authenticated and authorized, the scope of queries is still limited, allowing fine-grained disclosure (described later in this section). Retro-Future thus moderates incoming queries from remote clients by sensitivity before processing them to further minimize risks to privacy in the response.

Objectives: We need to moderate queries on both responder and querier sides to control the query's sensitivity. Controlling the sensitivity will enable each party to preserve privacy while providing (or receiving) usable output. Our insight is that we can achieve a privacy balance by continuously adjusting the information trade-offs in the query and response. We can apply this insight to both the responder and querier below.

Responders moderate queries by dynamically adjusting what Retro-Future does in response to incoming queries, selecting between different levels or layers of sensitivity. For example, the querier is more likely to get the answers that they need by adding additional details in their query. Put another way, responders are more inclined to answer more specific questions ("did 10.0.0.2 visit `example.com`?") than broad ones ("who visited `example.com`?").

Similarly, we also need to balance the privacy needs of the querier as the information disclosed in their query presents a privacy risk. In the previous example query, the more specific question reveals that the querier is particularly interested in an IP (the broader query obscures that fact). Situations, such as an ongoing security incident, may require the querier to limit what they disclose, with a minimal initial query followed by additional details in subsequent queries or a post-mortem.

Mechanisms: Retro-Future has several mechanisms that support moderating queries that enable queriers to receive actionable information (enabling forward progress) and responders to protect privacy (managing risk): a privacy budget to control the level of sensitivity and contextual access to additional query levels.

Users are allocated and spend a certain amount of their privacy/token budget (that replenishes over time) with any given query. Although general budget allocation is still an unexplored area (and remains an unsolved problem in its roots of differential privacy), we assign rough values based on the query's attributes like the data being queried or the query's specificity. For example, packet payload inspection is much more sensitive than its headers and correspondingly has higher cost—both in privacy budget and disclosure by the querier. A compromise that lowers both privacy and disclosure costs might be a query that matches on the hash of its contents.

Another mechanism to support query moderation and, ultimately, forward progress is the contextual access to additional queries. For example, a positive response on an initial query about a vulnerability ("were you affected by X?") might be followed up with a more sensitive query ("which IPs were affected by X?"), a query that would otherwise be rejected as overly sensitive without the context of an ongoing vulnerability. Thus, this ability to upgrade queries in light of context and corresponding evidence supports flexible policies, and Retro-Future's time travel allows privacy decisions to be made after-the-fact. Abuse of upgraded queries would be limited with the mechanisms in §3.2.

Upgraded permissions can be handled manually (by a human) or semi-automatically, which can lead to a fully automated "query negotiation" process (where both querier and responder settle on a satisfactory cost). Retro-Future supports manual escalation and automated de-escalation ("downgrade-until-success") of a query. Similarly, Retro-Future provides the raw APIs (via RPCs and ACLs) that a query negotiation mechanism could use.

3.2 Controlling Data Disclosure

Problem and Objectives: Controlling data disclosure is the final step in the query process and part of how we balance risk and forward progress, protecting users' privacy in the data while providing

useful results. Our insight is that by controlling the level of detail in the results while the query is being processed, the querier can query on more sensitive attributes (returning more useful results) while the responder maintains their users' and organizational privacy. For example, it is sometimes sufficient to the querier to receive a simple yes/no response, in contrast to a more traditional reply of matching packets or log entries. The terse nature of the simple reply limits disclosure of more sensitive data.

Mechanisms: We control data disclosure to solve the problem of providing usable results while protecting user privacy with three types of techniques: data minimization, rate limiting, and query logging and auditing.

Data minimization controls what is being shared by obfuscating or removing sensitive attributes that contain PII. Retro-Future makes use of existing tools like dnsanon [18] and LANDER [19], which anonymizes or removes payloads in DNS and network packet capture data. These tools can be used to minimize and replace the original raw data in archives (minimizing the risk of disclosure) or used on-the-fly as the results are being processed (keeping the original preserved).

Rate limiting controls disclosure by giving users a strict budget to spend on queries and query processing time, limiting data throughput for unintended data disclosure: Retro-Future uses existing rate limiting techniques in the new context of data sharing for preserving privacy. For example, execution time limits places constraints on how long a query can take, ensuring that users can't monopolize available processing power or run data-intensive queries (a query that runs on all available data).

Finally, existing techniques in query logging and auditing allows operators to quantify exactly what information is being shared and assess whether the data disclosure controls are too strict or permissive and adjust accordingly.

3.3 Securing the Retro-Future System

Because organizations are now using Retro-Future to collect and store sensitive data (network traffic, system logs, etc.) that was previously discarded, Retro-Future must be and remain secure to prevent increasing existing threats or introducing new to data disclosure.

Objectives: To meet an organization's security needs, Retro-Future's system security emphasizes and builds on data owners' full control over the system and data. Full control over data (thus eschewing storing data at cooperatives, escrows, or the cloud), allows owners to manage the risks in data sharing. By storing and managing data locally, organizations control both its disclosure and the flexibility in choosing how it is shared.

Mechanisms: To secure Retro-Future and corresponding data access, we adhere to best practices, using standardized and widely deployed protocols for access control (client-side certificates and Kerberos, SSH/TLS transport). Securing local data archives is done with current best practices, including data encryption, aging (removing information over time), and anonymization.

Table 2: Number of domains and IPs detected as suspicious activity at each site independently (self) and with sharing.

Site	Domains	%	IPs	%
CSU	1845	100%	9	100%
self	1845	100%	9	100%
with sharing	—	—	—	—
LANL	52	100%	2	100%
self	10	19%	1	50%
with sharing	42	81%	1	50%
USC/ISI	30	100%	2	100%
self	0	0%	0	0%
with sharing	30	100%	2	100%

4 QUANTIFYING THE BENEFITS OF SHARING FOR BOTNET ACTIVITY DETECTION

To show the benefits of data sharing, we next look at how cross-site data sharing helps in support of detecting bots and botnet activity on local networks from their command and control (C&C) traffic. We earlier looked at why DGA-based botnet detection will benefit from cross-site data sharing (§2). We will show how Retro-Future's cross-site sharing in the context of DGA-based botnets is beneficial to participating sites by first evaluating each site's individual ability in detection (§4.1), showing how sites can leverage data sharing to improve detection (§4.2), and their detection sensitivity (§4.3).

4.1 Can sites detect malicious activity on their own?

We first ask if sites can detect malicious activity *on their own* with BotDigger (without Retro-Future's data sharing).

We run BotDigger over 30 days (2017-Feb-16 to 2017-Mar-16) at CSU (20k hosts, 5.2 B queries), USC/ISI (158 hosts, 46.2 M queries), and LANL (26k hosts, 3.3 B queries). Botdigger looks at local hosts' DNS queries, clustering together and labeling queried domains as suspect based on their linguistic features. BotDigger will filter out false positives by setting a minimum threshold of 10 domains resolving to the same IP (this threshold was found in [20] to have an optimal true positive rate). We then analyze the suspected C&C domains and IPs to evaluate BotDigger's efficacy (did BotDigger detect bots or botnet activity?) and look for commonalities across sites (did each BotDigger instance detect the same botnet activity?).

Table 2 shows us that sites can sometimes detect malicious activity on their own ("self"). CSU, a large organization with data diversity, is able to detect suspect malicious activity with 1845 suspected C&C domain names resolving to 9 IP addresses. LANL, sitting between CSU and USC/ISI in terms of data diversity, detects 10 suspected C&C domain names resolving to 1 IP address. BotDigger at USC/ISI detected 0 suspect domains and IP addresses. Each site has its own localized view of suspicious activity, resulting in no detected commonalities across sites.

The amount of network diversity at an organization affects the amount of malicious activity detected—greater diversity correlates with more malicious activity. Although we see that LANL has a

Table 3: Common suspicious activity seen at each site. ('*' denotes detection with CSU's data sharing)

Site	# Domains Seen	
	98.124.204.16	208.100.26.234
CSU	10	26
LANL*	0	42
USC/ISI*	27	3
Total (distinct)	**37**	**70**
CSU & USC/ISI (*overlap*)	*0*	*1*

Table 4: The sensitivity of BotDigger's detection is improved with controlled data sharing. ('*' denotes that entry passes the detection threshold)

Site	# Domains Detected		
(color in Figure 1)	72.52.4.119	69.172.201.153	209.197.2.10
CSU (green)	3	4	1
LANL (blue)	7	9	0
USC/ISI (red)	0	0	21*
Total	**10***	**13***	**22***

comparable number of hosts and queries to CSU, its network is a relatively unique, secure environment (similar to a network in the financial or health services sector) where attackers have greater difficulty in taking over hosts. Smaller organizations (like USC/ISI) may not have sufficient network data diversity to detect large-scale malicious activity: it is possible that USC/ISI has no bots on its network, or BotDigger's algorithm is not sensitive enough.

We next ask if sites can detect more malicious activity when they use Retro-Future to sharing BotDigger's output of detected malicious activity with each other.

4.2 Can sites detect more malicious activity when they share?

We next show how sites can detect more botnet activity when using Retro-Future to share sensitive data. Here we focus on the benefit of sharing data from a large organization with data diversity (CSU) with less diverse (LANL) and smaller (USC/ISI) organizations.

To test if sites can detect more malicious activity when they share, we take CSU's botnet activity lists and share it with LANL and USC/ISI using Retro-Future. Both sites then check whether its hosts have queried or interacted with hosts in CSU's lists, potentially revealing activity with undetected malicious hosts. Retro-Future is required to support sharing since CSU regards botnet detection data as sensitive and restricted for limited sharing only.

LANL and USC/ISI retrieve CSU's list of 1845 suspected C&C domains and 9 IP addresses found by BotDigger (§4.1, list published on 2017-Mar-16). We then check if either site has queried or interacted with these hosts during a period of [2017-Feb-14, 2017-Apr-15] (30 days before and after the list was published), potentially revealing activity with previously undetected malicious hosts.

Table 2 shows that sites ("with sharing") can detect more malicious activity when they share. After sharing CSU's active botnet list, LANL and USC/ISI are able to find 81% and 100% of their total suspect domains and 50% and 100% of their total suspect IPs, respectively.

Table 3 highlights that data sharing enables sites to discover and detect commonalities across multiple sites. LANL and USC/ISI, again using CSU's shared botnet activity list, find matches on two IP addresses with 71 additional distinct domain names (107 total), previously unknown to either site (1 domain was seen by both CSU and USC/ISI). For example, both LANL and USC/ISI saw in their respective DNS activity 42 and 3 domains resolving to IP 208.100.26.234—activity that had not been detected by their local BotDigger instances.

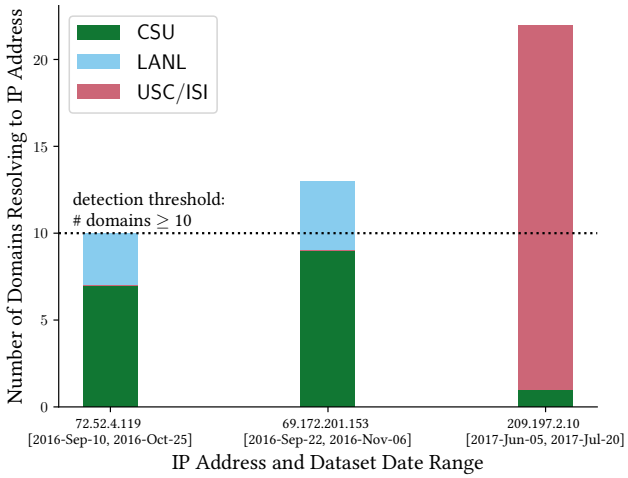

Figure 1: The sensitivity of BotDigger's detection is improved with controlled data sharing. All three domain/IP sets meet or pass the detection threshold.

Prior to sharing, a small organization like USC/ISI would not have been able to locally determine its interaction with CSU's detected malicious activity (we saw earlier that USC/ISI detected 0 suspect domains/IPs in §4.1). We next examine if the larger, more diverse site also benefits from sharing.

4.3 Can sites improve their detection sensitivity when they share?

We next demonstrate how sites can improve the sensitivity (true positive rate) of malicious activity detection when they share data with one another. False negatives can occur when potential botnet activity (C&C domains and IPs) as identified by BotDigger falls under the threshold of fewer than 10 IPs (as set in §4.1 and [20]) resolving to the same domain.

Sites can reduce false negatives by sharing and merging each others' BotDigger results and identifying any positives, suspect activity that meets the threshold. Each site exchanges botnet activity lists with C&C domain and IP pairs that fall below the threshold with a 45-day sliding window. They then combine the exchanged lists with their own results and check if the resulting output crosses the threshold, thus revealing previously undetected activity.

Table 4 and Figure 1 show that sites can improve their detection sensitivity when they share. Prior to sharing, each site (with the exception the 209.197.2.10 entry at USC/ISI) would have missed each corresponding domain/IP set as a false negative. With data sharing, the number of domains resolving to 3 particular IPs (72.52.4.119, 69.172.201.153, 209.197.2.10) reaches the minimum threshold of 10 domains per IP and are re-identified as possible C&C activity for further follow-up at each site.

Surprisingly, we see that even large, diverse organizations can benefit from sharing data. Prior to sharing, CSU would not have detected 209.197.2.10 as suspect, with 1 domain resolving to that IP. After exchanging botnet activity reports with USC/ISI, CSU combines USC/ISI's results with its own, 209.197.2.10 is now flagged as suspect, adding 1 additional IP/domain pair (+10% additional entries in their aggregated botnet activity list for the period of [2017-Jun-05, 2017-Jul-20]).

Organizations with all levels of diversity in their network traffic can benefit from sharing data by improving their detection sensitivity in botnet activity detection. In this final part of the case study, all sites benefited from sharing their respective BotDigger output, detecting more suspect activity that had previously fallen beneath the detection threshold. We plan on extending this case study in future work by analyzing detection performance over a longitudinal two-year period.

5 RELATED WORK

There have been many efforts and much work done in enabling and promoting Internet data sharing. We build on prior experience in information sharing frameworks and data collection.

Data Sharing Frameworks: Several logical frameworks for enabling and implementing data sharing have been proposed in prior work, outlining privacy, usability, and utility considerations in developing policies for data sharing.

Allman and Paxson, recognizing the prevalence of ad-hoc data sharing in the research community, proposed a set of high-level considerations for data sharing through "Acceptable Use" policies [2]. While they primarily consider Internet measurement data, these policies can be applied to cybersecurity incident data shared by a given organization. Retro-Future provides the mechanisms that can be used to implement and support data sharing in conjunction with these Acceptable Use policies, for example, by refusing queries from a requester who violates policy.

Kenneally and Claffy proposed a Privacy-Sensitive Sharing Framework (PS2) that seeks to balance risks that can occur with data sharing with privacy management [6]. Their framework enumerates the principles that a data sharing component should have, and challenges the assumption that the privacy risks of sharing data outweigh the benefits and they show that PS2 enables their organization, CAIDA [4], to realize utility goals in a risk-sensitive matter. Organizations can use PS2 as a guide to create policies and agreements with others, and use Retro-Future to enforce such policies in data sharing. We have also quantified the benefits of data sharing using Retro-Future in a case study of DGA-botnet detection.

We previously enumerated the privacy principles and corresponding engineering approaches for sharing cybersecurity data across organizational boundaries, recognizing the risk and benefit trade-off and the need to balance risks in disclosure with making forward progress in research and solving operational problems [9]. The Retro-Future system uses these principles and engineering techniques to implement a system for controlled information exchange across organizations and quantifies its benefits with a case study in malicious activity detection.

Data Collection, Storage, and Retrieval: There is much work in network data capture and collection, from the user-level [14] (used for capturing traffic at a specific node) to the network level [3, 11, 19]. Prior work has looked at both efficient (using deduplication or removing redundancy [17]) and secure (using encryption) capture and storage of network traffic for long-term storage and retrieval, especially in the context of intrusion detection [16] and network security analysis [13].

Retro-Future builds upon this work by looking at data capture and collection in the context and with the explicit purpose of sharing with other, outside organizations. Retro-Future encourages organizations to collect, archive, and use their network traffic and system log data, providing the mechanisms needed to share and use data with others in the context of *collaborative* (across organizations) intrusion detection and network security analysis.

6 CONCLUSIONS

This paper described our initial steps towards formalizing cross-site information sharing, and quantified the benefits of data sharing in the context of botnet detection. We enable controlled sharing by implementing cross-site queries through query moderation and controlled data disclosure in Retro-Future, a broader framework providing post-event understanding with time travel. We used the controlled sharing in Retro-Future in a case study on DGA-based botnet detection, showing how sharing cybersecurity data enabled sites to detect more malicious activity on their networks and improve the sensitivity of their detection algorithms.

Our future work entails further developing and evaluating trust relationship and data sharing policies between organizations, and greater longitudinal studies using Retro-Future and its data sharing in applications like DGA-based botnet detection, cybersecurity incident response, and network-wide malicious activity detection with DNS backscatter.

ACKNOWLEDGMENTS

The authors would like to thank Dimitrios Kounalakis, Christos Papadopoulos, and Han Zhang from Colorado State University and Patrick Avery, Paul Ferrell, Gina Fisk, Mike Fisk, Neale Pickett, and Shannon Steinfadt from Los Alamos National Laboratory for their contributions and help in supporting BotDigger and Retro-Future.

This material is based on research sponsored by Air Force Research Laboratory under agreements FA8750-17-2-0096 and FA8750-17-2-0280, and by the Department of Homeland Security Science & Technology Directorate, Cyber Security Division (DHS S&T/CSD) via contract number HHSP233201600010C. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

REFERENCES

[1] Reed Abelson and Matthew Goldstein. 2015. Millions of Anthem Customers Targeted in Cyberattack. *The New York Times* (5 Feb. 2015). https://nytimes.com/2015/02/05/business/hackers-breached-data-of-millions-insurer-says.html

[2] Mark Alllman and Vern Paxson. 2007. Issues and Etiquette Concerning Use of Shared Measurement Data. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (IMC '07)*. ACM, New York, NY, USA, 135–140. https://doi.org/10.1145/1298306.1298327

[3] C. J. Antonelli, M. Undy, and P. Honeyman. 1999. The Packet Vault: Secure Storage of Network Data. In *Proceedings of the 1st Conference on Workshop on Intrusion Detection and Network Monitoring - Volume 1 (ID'99)*. USENIX Association, Berkeley, CA, USA, 11–11. http://dl.acm.org/citation.cfm?id=1267880.1267891

[4] Center for Applied Internet Data Analysis. 2017. CAIDA. https://caida.org. (2017).

[5] Michael Cieply and Brooks Barnes. 2014. Sony Cyberattack, First a Nuisance, Swiftly Grew Into a Firestorm. *The New York Times* (30 Dec. 2014). https://www.nytimes.com/2014/12/31/business/media/sony-attack-first-a-nuisance-swiftly-grew-into-a-firestorm-.html

[6] K. Claffy and E. Kenneally. 2010. Dialing Privacy and Utility: A Proposed Data-Sharing Framework to Advance Internet Research. *IEEE Security Privacy* 8, 4 (July 2010), 31–39. https://doi.org/10.1109/MSP.2010.57

[7] Stacy Cowley and Liam Stack. 2016. Los Angeles Hospital Pays Hackers $17,000 After Attack. *The New York Times* (16 Feb. 2016). https://www.nytimes.com/2016/02/19/business/los-angeles-hospital-pays-hackers-17000-after-attack.html

[8] Julie Hirschfeld Davis. 2015. Hacking of Government Computers Exposed 21.5 Million People. *The New York Times* (9 July 2015). https://www.nytimes.com/2015/07/10/us/office-of-personnel-management-hackers-got-data-of-millions.html

[9] Gina Fisk, Calvin Ardi, Neale Pickett, John Heidemann, Mike Fisk, and Christos Papadopoulos. 2015. Privacy Principles for Sharing Cyber Security Data. In *Proceedings of the IEEE International Workshop on Privacy Engineering*. San Jose, California, USA. https://doi.org/10.1109/SPW.2015.23

[10] Vindu Goel and Nicole Perlroth. 2016. Yahoo Says 1 Billion User Accounts Were Hacked. *The New York Times* (14 Dec. 2016). https://www.nytimes.com/2016/12/14/technology/yahoo-hack.html

[11] Stefan Kornexl, Vern Paxson, Holger Dreger, Anja Feldmann, and Robin Sommer. 2005. Building a Time Machine for Efficient Recording and Retrieval of High-volume Network Traffic. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC '05)*. USENIX Association, Berkeley, CA, USA, 23–23.

[12] Ralph Langner. 2011. Stuxnet: Dissecting a Cyberwarfare Weapon. *IEEE Security & Privacy* 9, 3 (2011), 49–51.

[13] Gregor Maier, Robin Sommer, Holger Dreger, Anja Feldmann, Vern Paxson, and Fabian Schneider. 2008. Enriching Network Security Analysis with Time Travel. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication (SIGCOMM '08)*. ACM, New York, NY, USA, 183–194. https://doi.org/10.1145/1402958.1402980

[14] Steven McCanne and Van Jacobson. 1993. The BSD Packet Filter: A New Architecture for User-level Packet Capture. In *Proceedings of the USENIX Winter 1993 Conference Proceedings on USENIX Winter 1993 Conference Proceedings (USENIX'93)*. USENIX Association, Berkeley, CA, USA, 2–2.

[15] National Institute of Standards and Technology. 2017. Framework for Improving Critical Infrastructure Cybersecurity Version 1.1 Draft 2. (Dec. 2017). https://www.nist.gov/framework

[16] Vern Paxson. 1998. Bro: A System for Detecting Network Intruders in Real-time. In *Proceedings of the 7th Conference on USENIX Security Symposium - Volume 7 (SSYM'98)*. USENIX Association, Berkeley, CA, USA, 3–3.

[17] Lin Quan and John Heidemann. 2010. On the Characteristics and Reasons of Long-lived Internet Flows. In *Proceedings of the ACM Internet Measurement Conference*. ACM, Melbourne, Australia, 444–450.

[18] USC/ISI. 2016. dnsanon. (2016). https://ant.isi.edu/software/dnsanon/

[19] USC/ISI. 2017. LANDER Trace Capture Software. (2017). https://ant.isi.edu/software/lander/

[20] Han Zhang, Manaf Gharaibeh, Spiros Thanasoulas, and Christos Papadopoulos. 2016. BotDigger: Detecting DGA Bots in a Single Network. In *Proceedings of the International Workshop on Traffic Monitoring and Analysis (TMA '16)*. IFIP.