Can Cybersecurity Be Proactive? A Big Data Approach and Challenges

Hong-Mei Chen
Shidler College of Business
Univ. of Hawaii at Manoa
Honolulu, USA
hmchen@hawaii.edu

Rick Kazman
Univ. of Hawaii at Manoa
& SEI/CMU
Pittsburgh, USA
kazman@hawaii.edu

Ira Monarch
Independent Consultant
Pittsburgh, USA
iramonarch@gmail.com

Ping Wang
University of Maryland
College Park, USA
pwang@umd.edu

Abstract

The cybersecurity community typically reacts to attacks after they occur. Being reactive is costly and can be fatal where attacks threaten lives, important data, or mission success. But can cybersecurity be done proactively? Our research capitalizes on the Germination Period—the time lag between hacker communities discussing software flaw types and flaws actually being exploited—where proactive measures can be taken. We argue for a novel proactive approach, utilizing big data, for (I) identifying potential attacks before they come to fruition; and based on this identification, (II) developing preventive counter-measures. The big data approach resulted in our vision of the Proactive Cybersecurity System (PCS), a layered, modular service platform that applies big data collection and processing tools to a wide variety of unstructured data sources to predict vulnerabilities and develop countermeasures. Our exploratory study is the first to show the promise of this novel proactive approach and illuminates challenges that need to be addressed.

1. Introduction

The number and variety of cyber-attacks is rapidly increasing, and the rate of new software vulnerabilities is also rising dramatically. According to a recent study "the compound annual growth rate (CAGR) of detected security incidents has increased 66% year-over-year since 2009" [21]. But the software security community is typically reacting to attacks *after* they occur. Being reactive is costly and can be fatal, where attacks threaten lives, important data, or mission success. Unfortunately, existing research on cybersecurity has focused almost exclusively on reactive strategies. Some attempts to be "proactive", such as in the guidelines published by IEEE Center for Secure Design [14], have been outlined, but these are limited to the scope of software design and are rather abstract.

Predictive analytics, an emerging tool being used to identify potential cyber threats against organizations, has the capability to be proactive but currently it is not. The emerging predictive analytics used in the security industry attempts to build a specific response to a specific cybersecurity threat [26]. As attackers find new ways to avoid detection, predictive analytics helps security professionals find unknown malware wherever it may be hiding. Bit predictive analytics, as it is currently practiced, doesn't mean seeing an attack before it occurs [26], which is what we mean by being "proactive". For instance, the analytics software company FICO, although still not "proactive" by our definition, used predictive real-time analytics to respond to data breaches faster than before [12]. The traditional approach of gathering data on a compromise, developing a threat's "signature" and then using that signature to protect against future threats, results in massive time delays. FICO, in contrast, identifies threats as they come on the scene by identifying anomalous patterns using real-time analytics. This identification has to build on the profiling of attacks that are currently known. But by this time considerable damage has already been done.

Can we be truly proactive about cybersecurity, in the sense that we can prevent the attack *before* it occurs? Can we predict what concepts that are emerging in the hacker community will eventually evolve into a successful exploit or an attack? These are our research questions. Our research is on analogy with the medical industry's use of predictive analytics to proactively prevent disease outbreaks. To be proactive, the disease has to be recognized *before*, not after, it becomes widespread.

In addition, continuing the medical analogy, to be proactive, we not only look out for external attacks, e.g., disease outbreaks, but we also need to look internally, in terms of an individual's predisposition to a disease. It has been shown [20] that a majority of security bugs—nearly two thirds—are "foundational"; that is, they have existed for many years in a system's legacy code. Many of them are, in fact, 0-day vulnerabilities, which give no

URI: http://hdl.handle.net/10125/41885
ISBN: 978-0-9981331-0-2
CC-BY-NC-ND

time to plan any mitigation against their exploitation once the flaws become known. To be proactive, organizations must take security assurance steps after a software product has been released, but before the broad hacker community discovers its vulnerabilities [7].

The proactive approach, if proven feasible, would be a game-changer for the cybersecurity community. Our research is motivated by the enormous potential benefits of this approach. The proactive approach is appealing, but questions remain whether it is feasible. Our exploratory study is the first to show the promise of this novel proactive approach, utilizing big data, and illuminates challenges that need to be addressed.

In what follows, we discuss the "Germination Period", a time lag between hacker communities discussing software flaw types and those flaws being exploited. Our definition of Germination Period includes the previously identified "Honeymoon period", which occurs after the release of a system, but before the identification of its first vulnerability [7]. Both of these periods represent opportunities where proactive counter-measures can be advantageously taken.

The rest of this paper is structured as follows: In Section 3, we present our vision of a Proactive Cybersecurity System (PCS), based on the big data approach. In Section 4, we describe our research framework and discuss challenges and directions for realizing PCS. Section 5 presents our exploratory study. Section 6 concludes with remarks for future work.

2. The Germination Period and Big Data

We first conducted a literature review on the patterns of past cyber-attacks to help answer our research questions. We found that the 'black hat' (offensive hacker) community is a *learning community* with unique ecologic properties, and we found there's a time lag that we called the Germination Period. This is the time between the emergence of a vulnerability concept in the hacker community and the creation of successful attacks. It is during this Germination Period that we can be proactive. For example, in May 2005, Robert Seacord, a security specialist at the Software Engineering Institute, published the first edition of Secure Coding in C and C++. On page 156 of his book, he cautioned about "referencing freed memory" [23]. In 2007, researchers from WatchFire reported a "Dangling Pointer" vulnerability in Microsoft IIS [1] and Justin Ferguson gave a talk at the Blackhat conference reporting one of the first valid exploits of what became known as Use-After-Free (UAF) [9]. Blogs and tutorials related to the concepts of UAF began to appear frequently around 2010. Figure 1 below shows the reported number of common vulnerabilities and exposures (CVEs), by year, for UAF entries. Successful UAF attacks can have serious consequences: corruption of data, and the execution of arbitrary code.

Clearly the offensive hacker community learned (about UAFs) and just as clearly it takes time, from the initial discovery of a vulnerability until it becomes a significant and viable threat to the "white hat" community. This time lag between 2006 to 2010, the Germination Period, during which offensive communities are gaining understanding and expertise and planning exploits, represents an opportunity for proactive counter-measures. But such counter-measures can only be applied if the potential threat is determined early enough.

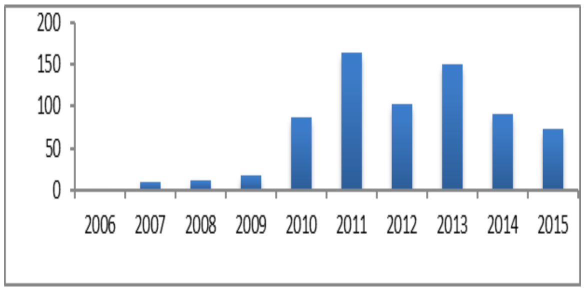

Figure 1. Use-After-Free Common Vulnerabilities and Exposures (CVEs)

How does the offensive community gain understanding and expertise in planning exploits? What traces do they leave? In our exploratory study, we have identified two main categories of data sources containing information leading to emerging concepts describing vulnerabilities that are likely to be targeted: (1) hacker communities, and (2) public security databases. Both types of data sources discuss vulnerabilities, PoC (Proof of Concept) exploits, and attacks. Interestingly, we have noted that both source types recognize occurrences of PoC exploits, attacks, and vulnerabilities at the same time. It has been shown that there are time delays, e.g., the Germination Period, both between the identification of vulnerabilities and the production of PoC exploits and also between hostile attacks targeting these vulnerabilities and the corresponding PoC exploits [6][23]. We intend to broaden these data sources, thus taking a big data approach, to show something more general, by identifying as many contexts as possible and determining whether the time delays are different in different contexts.

Hackers form communities. Some of the hackers' blogs, software repositories, IRC channels, etc. can be found on the internet. They are learning communities and they are innovation communities, no different from entrepreneurs, venture capitalists, researchers and even terrorist organizations. This is why they are successful and why we fear them. But, mounting a successful attack requires tremendous resources and patience. Hacker communities, as with all innovation communities, need to share information to be effective; they build on each other's work and discourse, sometimes directly but more often indirectly [26]. By analyzing the topics in hackers' discussions, we will be are likely to be the focus of upcoming attacks. Early insight can lead to early, and hence more effective, quality assurance and mitigation strategies.able to get early indication as to which vulnerabilities

As a result, we believe and will present early evidence that we can mine hacker discussion forums, blogs, and Internet Relay Chat (IRC) channels (e.g. freenode.net, AnonOps IRC, Metasploit IRC, Google Project Zero, blackhat.com, GMANE.org, seclists.org) to identify emerging concepts. In this way the software security community can be more proactive in detecting and eliminating vulnerabilities, rather than simply reacting to vulnerabilities as they occur. For example, the Heartbleed bug was discovered simultaneously by (defensive) security researchers at Google and at Codenomicon, avoiding potentially huge losses if hackers had found this bug first (in April, 2014 more than 2/3 of the world's web servers were vulnerable to Heartbleed).

In addition, our exploratory study has already mined publicly available vulnerability, exploit, and attack databases such as CVEs (cve.mitre.org), CVE Details (cvedetails.com), and the Open Web Application Security Project's (OWASP) WASC Web Hacking Incidents Database (WHID) (https://www.owasp.org/index.php/OWASP_WASC_Web_Hacking_Incidents_Database_Project) to create an initial ontology of prominent security concepts.

There are also important differences between these two types of data sources. By collecting data from various sources, we can assemble information about different aspects of the same vulnerabilities and exploits. This means that such differences can be combined and compared for better understanding of the conditions responsible for the time delays between vulnerabilities and exploits—a distinct advantage of utilizing big data.

3. Proactive Cybersecurity System (PCS) Vision

Our big data approach results in a vision of the Proactive Cybersecurity System (PCS) as shown in Figure 1. Grounded on a wide variety of unstructured big data sources, the PCS has two goals:

Goal I: identify potential attacks before they take place and cause harm, and based on this identification,

Goal II: develop preventive counter-measures.

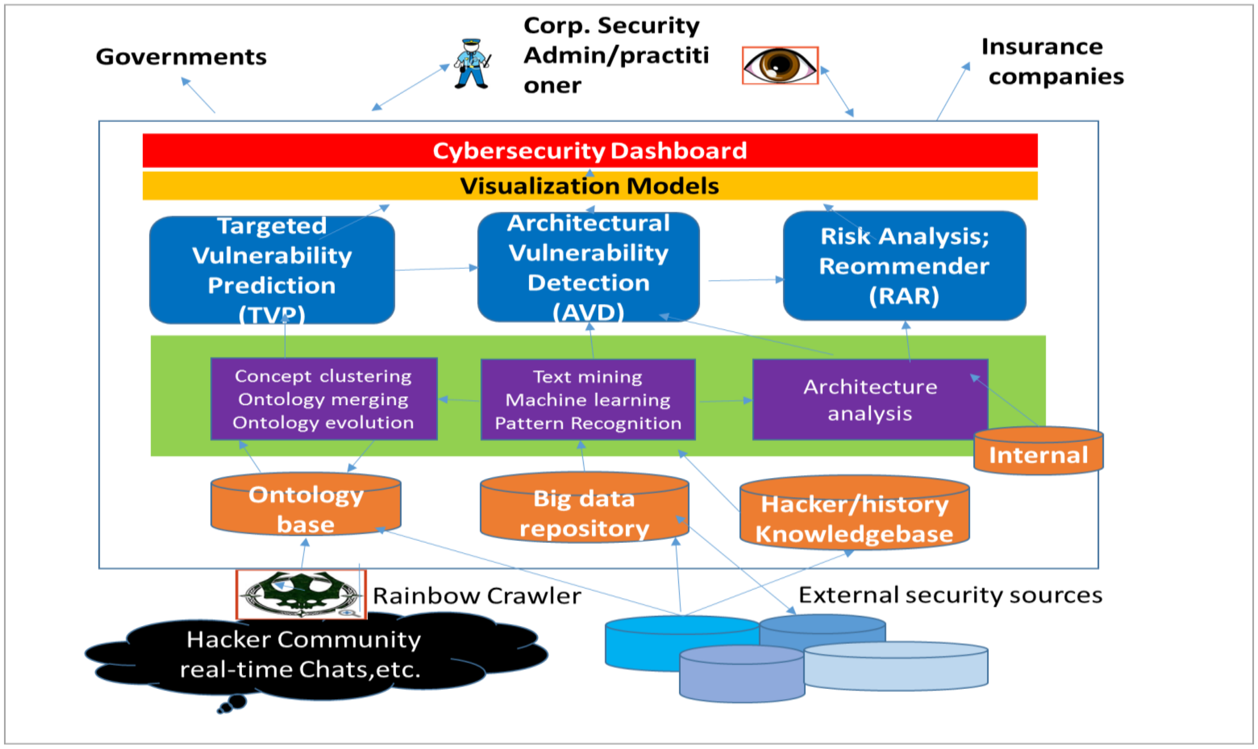

Figure 2. Proactive Security System

To achieve Goal I, a Targeted Vulnerability Prediction (TVP) subsystem detects, from hackers' ad hoc communities and publicly available security sources, the emerging concepts that are the early warning signs of likely vulnerability targets. To achieve Goal II, (a) an Architectural Vulnerability Detection (AVD) subsystem and (b) a Risk Analysis and Recommender (RAR) subsystem were designed. AVD adds a further capability of predicting the impact of the attack vectors identified in the TVP subsystem on a system architecture of a company. RAR analyzes the risks associated with identified vulnerabilities, estimate the costs of mitigation actions, and recommend refactoring and assurance strategies. The TVP, AVD, and RAR subsystems constitute the PCS, a modular service platform that combines data sources, data collection and processing tools, metrics and models for use of security personnel, researchers, governments, and insurance companies.

4. Research Framework and Challenges

Figure 3 shows our research framework for developing a PCS. We will discuss the research steps, analytical foundations, tools employed as well as the preliminary results for achieving Goal I and Goal II.

As shown in Figure 3, for TVP our research steps include (1) identifying big data sources; (2) collecting and managing big data; (3) identifying emerging concepts; (4) tracking concept evolution; and (5) prioritizing vulnerabilities. In parallel, we develop internal proactive measures for an organization by the AVD and RAR subsystems. Next we describe our preliminary directions and illuminate the challenges to be addressed in each step.

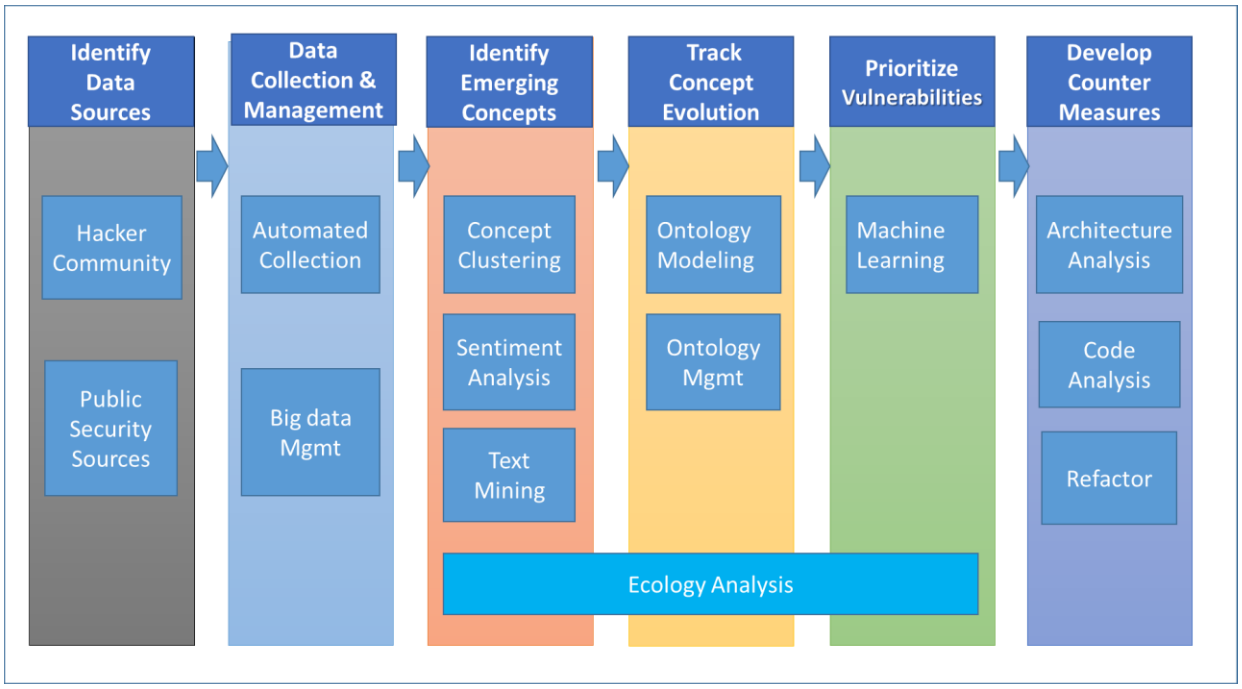

Figure 3. PCS Research Framework

4.1 Identify Big Data Sources

Inherent in the big data approach, identifying and assessing the data sources is a critical activity as the subsequent analysis and proactive measures rely on the quality, comprehensiveness, and reliability of the data. There are many potential data sources available and different data sources have different characteristics and provide different information. For instance, one important difference is that most of the public security databases do not provide information about who contributed an entry to the database. Hacker sources do typically identify who is making a contribution. In many cases, however, the names provided are fanciful and an individual may not use the same designation in different chats, lists and database contributions.

Furthermore, publicly accessible online databases are maintained by various organizations. MITRE's CVE database collects vulnerability, exploit and attack information. An offshoot of this is CVE Details. This website identifies vulnerability and corresponding exploit types for which advanced searching can be done. This data can be tabulated showing frequencies of instances of vulnerability or exploit types on a yearly basis. This tabularized data can also show the frequency of exploits across all types also on a yearly basis. We have already explored some of this data, and patterns have emerged. In most cases, there are spikes in certain years in the number of recognized vulnerabilities. In some cases, the changes from one year to another can be as great as 1000 occurrences.

Similar variations and spikes in frequency are seen in data on exploits, both PoC and hostile. This information can be mined from CVEs, CVE Details, the exploit databases, and the WHID. As with the vulnerability frequency data, the frequency of occurrence of PoC exploits and attacks changes over time. Determining the root causes for such patterns, particularly the spikes, is one of our research goals. We may find that some attacks come prior to PoC exploits and close in time to the discovery of a vulnerability, perhaps even before its discovery. In such cases initial attacks would not be preventable. Even if we don't find the requisite events or conditions that occur enough before all attacks, the events and conditions we do find will enable us to make predictions about a spike in attacks that we can mitigate. In addition to keeping track of instances of vulnerabilities, exploits and attacks, we also have to keep track of what category in the ontology they are instances of. We may find, for example, that while instances are increasing in a high-level category, they are only increasing in certain sub-categories, and not others. It is these specific increasing subcategories that provide a basis for mitigation strategies.

There are several challenges here. Exploit databases typically have much more extensive coverage of exploits than the CVE Details website. Also, the WHID's collection of attack instances is much smaller than the true number since organizations are often reluctant to acknowledge that they have been attacked. Because of these and other discrepancies among the data sources, our analyses will not treat any source as definitive. We will instead triangulate over several data sources, and in the TVP module we will generate a confidence score for the predictions, depending on the extent to which trends discovered in multiple data sources are compatible.

4.2 Collecting and Managing Data

Collecting and managing this big, unstructured data presents significant challenges. Quantifying instances of vulnerabilities and exploits is currently done through numerous manual searches, laboriously selecting and counting entries. One of our goals is to automate this process as much as possible, although we realize that a human will always be "in the loop", as indicated in Figure 2. We will utilize existing web spider technology to collect data from hacker forums. Also, we have gained substantial experience in network evolution visualization and successfully developed web-scraping and crowdsourcing tools, which will be core modules for data collection and management. Large volumes and different varieties of data will have to be collected from the main data sources, ingested, stored and prepared for analysis. A big data repository is thus planned for storing the raw data to allow "schema on read" [5][6] for different types of analysis. There are tremendous technical challenges in terms of preparing data for analysis. The data cleaning and integration is not a trivial task [6].

4.3 Identifying Emerging Concepts

Accurately identifying emerging concepts is critical to the success of PCS. To address the inherent complexity of the data collected, we are employing text mining, concept clustering and sentiment analysis techniques to identify: 1) emerging concepts against the background of more prominent and lasting ones; and 2) emerging hacker communities associated with the emerging concepts. Because of the huge amount of data involved, manual curation will not be possible in general, and so PCS needs to aid and guide a human analyst who will make the final interpretation and decision to develop countermeasures.

For 1), we are primarily applying text mining (extracting and clustering noun phrases [3]), concept clustering and mapping, and ontological analysis to identify and track concepts. The text mining results will provide continuous input for the concept clustering, mapping and sentiment analysis phase and together they will provide results for inclusion into an evolving ontology. Ontology building is done manually at the moment, but we are investigating ways to automate as much of this as possible.

For 2), the tasks are: a) elaborate the structure and evolution of hacker communities by analyzing their network structures; and b) determine which of the emerging concepts are not only likely vulnerabilities but which are likely targets of attacks and hence worthy of attention by a human analyst.

For concept clustering, mapping and sentiment analysis, we are employing the Leximancer tool (leximancer.com). Leximancer analyzes the frequencies and co-occurrence relationships between words in a text corpus and produces *concept maps* that show and name the significant concepts in the corpus. Leximancer also shows the relationships among the most significant concepts used in a text corpus, including those that express sentiment [16][24]. It enables rapid analysis of tens of thousands or more text entries in records like those collected in Gmane or CVE List, but also allows modulation of the results through researcher intervention and interpretation. As an example, a portion of the concept map from a completed automated analysis of the entire CVE List circa August 2015 is shown in Figure 4.

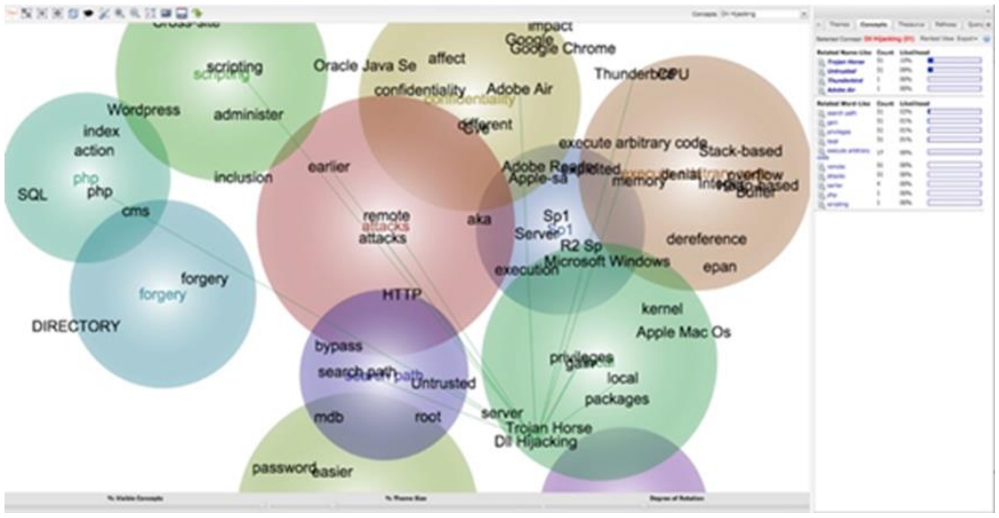

Figure 4: CVE Concept Map

4.4 Tracking Concept Evolution

We iteratively perform three interrelated processes to mine concepts and track their changes. The concepts to be mined and tracked cover conditions leading to the identification of vulnerabilities and exploits (both non-hostile and hostile) along with a characterization of the vulnerabilities and exploits themselves and their classification. The characterization differentiates, and the classification relates, the individuals, groups, communities and organizations, the systems and applications, and the processes, methods and techniques involved. The three processes are:

1) mining security data sources using noun-phrase parsing, automated terminology construction, statistical analysis and clustering to determine the most salient concepts [16] in the corpora being analyzed and track their changes through time;
2) mapping the relationships among these concepts and also tracking their changes through time, employing Leximancer. This will generate a series of maps representing the changing networks of the most prominent, relevant, and important concepts mined, including concepts representing both positive and negative sentiments;
3) building a security ontology [17][19][9], that we call the Emergent Vulnerabilities and Exploits Ontology (EVEO), based on the results of 1) and 2) that will help guide the construction and tracking of emerging concepts.

One challenge is to compare a series of maps, and to view the changing state of the ontology over time. However, we have already identified a promising candidate technology, TopicFlow, to aid us in visualizing topics and topic evolution [25].

4.5 Prioritizing Vulnerabilities

Not all vulnerabilities are of the same value. We have identified two approaches to help prioritize identified vulnerabilities. One is using machine learning to determine the characteristics of a “high priority” vulnerability. We will perform retrospective case studies as training sets for the machine learning. The other is to apply ecology theory [13] to help identify emerging concepts, refine the categorization, and prioritize the vulnerabilities. Two processes coexist in each concept's community: symbiosis and competition. As more and more participants join a community, their symbiosis allows them to coexist in the community, through collaboration and cooperation. At the same time, as more and more participants are in the community, they may face increasing competition with each other. Accordingly, the density of a community is used as an indicator for both symbiosis and competition among individual participants.

We will identify, as far as possible, the participants in the hacker discussion forums, blogs, and IRC channels who contributed to or commented on the topics and concepts detected. The participants for each concept will be considered as a *community* for that concept. If two participants contributed content to the same concept in one of the venues or they commented on each other's contributions, then there is a relation between them. Based on these participants and relationships, we are able to build a network for each concept for any period of time. To assess the impact of ecology on the evolution of concepts, at the community level, we will apply the density-dependence model [13] to explain the vital rates of concept communities (e.g., entry, growth, and exit rates). Using a technique of text mining—sentiment analysis—we will also be able to associate rates of entry with different sentiments, thus enhancing our understanding of concept trajectory and momentum.

4.6 Developing Countermeasures

The AVD and RAR subsystems will enable internal proactive measures for an organization, taking advantage of the Germination Period. For a specific organization producing software the Germination Period will be a short window of opportunity after their software product has been released, but before the hacker community discovers its vulnerabilities. Currently, there exist few proactive methods. As aforementioned, it has been shown [20] that a majority of security bugs—nearly two thirds—are "foundational." Taken together with the Germination Period, these observations suggest that one cannot simply try to find all of the security bugs in a system, but rather must take a strategic, risk-driven approach to security assurance. For this reason, we are constructing a cybersecurity countermeasures approach, extending the existing tools and methods:

1) Driven by input (e.g. candidate emerging vulnerabilities) from the TVP subsystem, we will analyze the architecture of individual systems, using the Architecture Analysis for Security (AAFS) method [22] to understand the risks posed by these vulnerabilities; The AAFS method grew out of existing architecture analysis techniques, such as the architecture tradeoff analysis method (ATAM) [8], but focuses solely on security. The rationale behind architectural analysis is that discovering design problems during coding or maintenance is too late, because addressing these problems later in the life cycle is costly, risky, and disruptive to a project. At the point in a project's lifecycle when a software architecture has been established, but before much code has been written, the architecture can be analyzed for risks [2].

2) A toolcalled Titan [28] will be used to identify architectural structures that are potentially implicated in the targeted vulnerabilities, to locate the design flaws within these structures, and to identify the specific files within these structures that have the highest probability of experiencing a security bug. The Titan tool chain takes, as input, a project's source code, its revision history (from a configuration management tool such as SVN), and its issues (from an issue-tracking system such as Jira) and, based on this input, clusters the architecture into a set of overlapping DRSpaces. These DRSpaces are then analyzed for architectural flaws—anti-patterns that we call hotspots. These hotspots have been shown to be highly correlated with bugs, changes, and churn [17]. And we have discovered that these results hold for security as well [10]. That is, when a file is implicated in architectural flaws, it is significantly more likely to be involved in a security bug. The more flaws a file is implicated in, the greater the probability that the file will experience security problems.

3) Using the knowledge from 1) and 2), the RAR subsystem will propose refactoring solutions to the architectures, based on removing the design flaws [15][17]. AAFS and Titan techniques serve to identify the risks in the system. To actually remove these risks, the system under scrutiny needs to be fixed and often this requires refactoring, to remove the identified hotspots. While it is true that many bugs are caused by pure coding errors, our Titan-based results suggest that architectural flaws play a large role in increasing the frequency of security bugs. Thus, no simple coding solution will fix this problem. The only way to fix it is to refactor the architecture, to remove the flaws. Fortunately, we have the necessary information to do just that. The Titan tool identifies not only flawed parts of the architecture, but also the reasons for the flaws and the precise set of files implicated.

4. Exploratory Study

We have conducted a retrospective exploratory study to gauge the feasibility of the proactive cybersecurity approach and the TVP design. We analyzed both the Gmane "Full-Disclosure" email list (http://dir.gmane.org/gmane.comp.security.fulldisclosure) and the CVE database for comparison. We wanted to see the differences between the contents of the hacker mailing list and the CVE database for purposes of characterizing already recognized vulnerabilities and identifying and characterizing emerging ones. The Gmane list, which covers 2010 to 2015, is well over 250 MB. It contains a lot of noise including binary code, source code and boilerplate information (such as advertisements for security products and services) that can be repeated thousands of times, but contribute little, if any, useful information. Cleansing such files is part of our text mining process.

Our initial approach was to extract and mine the contents of the Subject and Date fields. There were over 22,000 Subject and Date fields. We found similarities and differences in the coverage of DLL hijacking when we analyzed the two data sources. Additional information about product targets found in hacker discussions was not found in CVE database contents. The additional information that hacker discussions provide increases the potential for preventing or reducing attacks on these targets crucial for an early warning model relevant to attacks on these targets. Conversely, the relationships among DLL Hijacking on the one hand and Trojan Horse, local users, gaining privileges, untrusted search path vulnerabilities, and executing arbitrary code on the other are, in fact, the primary considerations in the CVE entries. Knowing these relationships is an important part of understanding how DLL Hijacking attacks are performed and what kinds of targets they are likely to aim for.

In short, the two data sources (Gmane and CVE) supplement one another. Both sources are needed for an early warning model that prevents attacks or mitigates their numbers. The retrospective case study re-enforces the importance of 1) identifying and integrating information from various data sources and 2) determining what information can be indicative of emerging vulnerabilities, new forms of exploit, or the (types of) targets of future attacks.

For example, in analyzing the text of "Full Disclosure" for 2009 we can see that DLL is an important concept, but "hijack" that has not emerged as a concept. There are other, however, potentially relevant concepts that are clustered close to DLL, such as "bypass" and "exploitation", as shown in Figure 5.

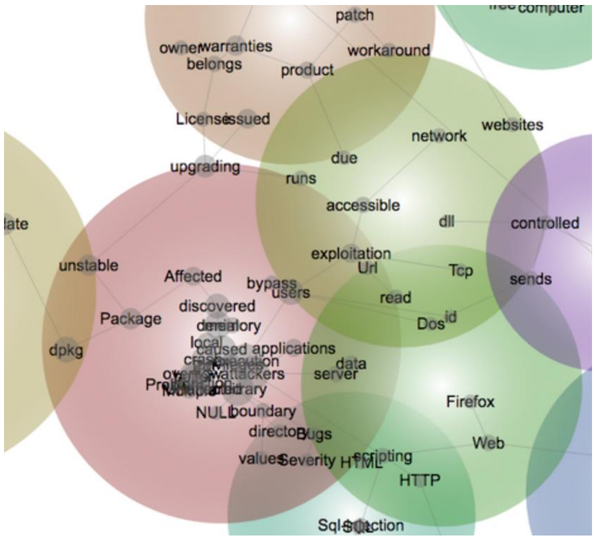

Figure 5: Concept Clusters from 2009 "Full Disclosure"

Using our ontology, however, we can attempt to "seed" the concept clustering process. An analysis of the same "Full Disclosure" list for 2009, but with seeding, shows DLL and "hijack" in a single cluster, as shown in Figure 6.

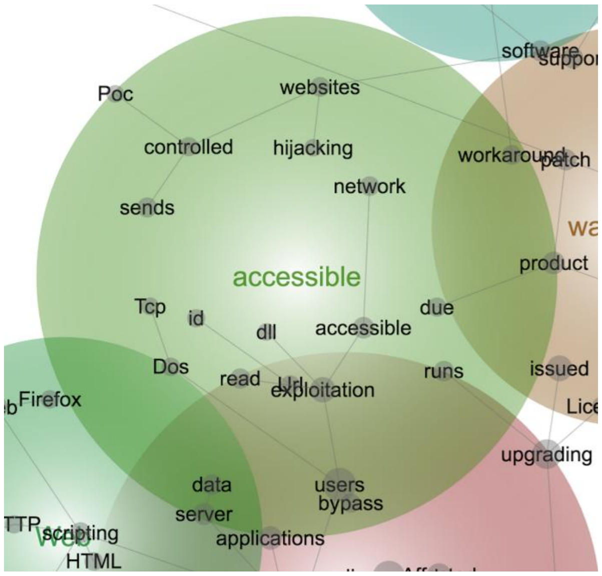

Figure 6: 2009 "Full Disclosure" with Seeding

Finally, in 2010, we can see that DLL and hijacking appear clustered together, as shown in Figure 7, and distinct from other attack types such as SQL injection.

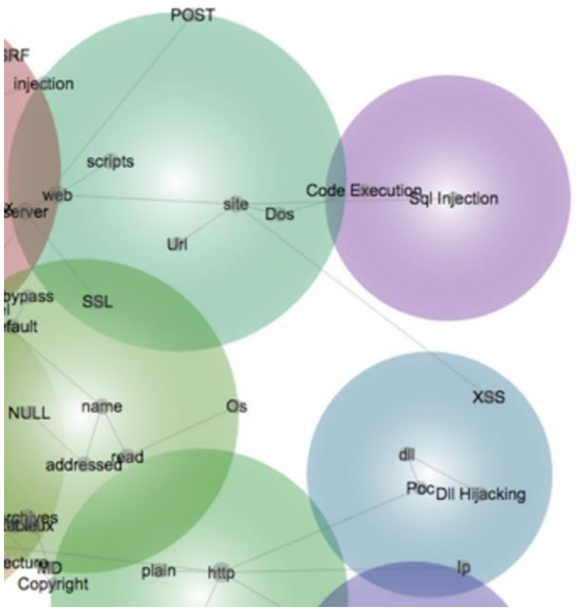

Figure 7: Concept Clusters from 2010 "Full Disclosure"

This case study demonstrates the two major components of the TVP module of PCS: text mining and concept clustering. Together they aid an analyst in identifying potential categories (types), subcategories, their characteristics and relations, all elements of an ontology. Building a useful PCS ontology requires keeping track not only of all the concepts that have been discovered, as well as their associated characteristics and relationships at every level of the concept hierarchy, but also a way of keeping track of when people became aware of the concepts and when their corresponding instances occur. Having a way of keeping track of these correspondences is crucial, and this is precisely what the PCS ontology is designed to do. There are other reasons why building an ontology is critical. The hierarchical and relational structure of the categories provide blueprints of how new categories are derived from existing categories. This could be the basis for detecting emerging concepts.

For example, using text mining results from the Full-Disclosure list, we found that there are occurrences of terms standing for concepts like SQL-injection, XSS-injection and SQL and XSS-injection. Since we also found HTML-injection, might we find HTML and XSS-injection? In fact, we did find this. However, if we did not, this would put us on the lookout for it—a proactive measure. Also, for sibling categories like remote code execution and local code execution, if a code execution exploit or a code execution attack has been identified, we would look for a remote-code execution exploit or a local-code execution exploit and a remote-code execution attack or a local-code execution attack. Assuming that an exploit of a vulnerability is to make it part of a viable method for attacking a system, if we find an exploit but not a corresponding attack, this might be an early warning sign that the corresponding attack is imminent.

6. Conclusions and Future Work

In this paper, we have explored the idea of a proactive approach to cybersecurity and shown promising progress towards this goal. We have identified an opportunity in the Germination Period, which is the time-period during which proactive measures may be most advantageously taken, and we have shown that a proactive approach to cybersecurity, utilizing big data, holds enormous potential. We also contributed the design of the Proactive Cybersecurity System, which serves as a research framework, and illuminates a number of research and practical challenges that need to be addressed. Big data is the predictive analytics foundation for the PCS. The PCS rests on a big data infrastructure for extracting information from hackers' communities and security data sources, transforming (cleansing) and loading the data, clustering and visualizing it, and curating it for future use. Our first exploratory retrospective study showed significant potential as a training set for machine learning. We are currently developing more retrospective studies and heuristics for machine learning and hope to develop the full PCS as envisioned.

A proactive approach to cybersecurity will be a game-changer. *If successful*, we expect our eventual research results will guide quality assurance and risk mitigation activities, allowing the security assurance community to be proactive rather than reactive. Although security assurance personnel must have been doing some of this already, they currently do so in an ad hoc fashion, based on their personal experience. Thus they are operating without proper decision support and with limited, typically organization-internal data. These existing efforts will be significantly enhanced by the PCS.

We must stress that the challenges that we have already encountered are not trivial. The data to be collected is vast and poorly structured, and the analysis is complex. We are truly looking for needles among haystacks. But in view of the enormous benefits that may be achieved by the proactive approach, we are compelled to share our preliminary results, hoping to engage broader participation and collaboration for building a proactive cybersecurity community and realizing the PCS vision.

7. Acknowledgement

The last co-author would like to acknowledge the support from the National Science Foundation's BIGDATA program (Grant # IIS-1546404).

8. References

[1] Afek, J., Sharabani, A. 2007. "Dangling Pointer: Smashing the Pointer for Fun and Profit." http://www.orkspace.net/secdocs/Conferences/BlackHat/USA/2007/Dangling%20Pointer-paper.pdf.

[2] Bass, L., Clements, P., Kazman, R. 2012. Software Architecture in Practice. Addison-Wesley.

[3] Bhat, T.N., Collard, J., Subrahmanian, E., Sriram, R.D., Elliot, J.T., Kattner, U.R., Campbell, C.E., Monarch, I. 2015. "Generating Domain Ontologies Using Root- and Rule-Based Terms," NIST Information Technology Laboratory.

[4] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M. 1996. Pattern-Oriented Software Architecture Volume 1: A System of Patterns. Wiley.

[5] Chen, H-M, Kazman R., and Haziyev S. 2016. "Agile Big Data Analytics Development: An Architecture-centric Approach," IEEE Proceedings of Hawaiian International Conference on System Science (HICSS-49), Grand Hyatt, Kauai.

[6] Chen, H-M, Kazman R., and Haziyev S. 2016. "Agile Big Data Analytics for Web-based Systems: An Architecture-centric Approach," *IEEE Transactions on Big Data*, in press, April 2016.

[7] Clark, S., Frei, S., Blaze, M., and Smith, J. 2010. "Familiarity Breeds Contempt," in Proceedings of the Annual Computer Security Applications Conference (ACSAC) 2010.

[8] Clements, P., Kazman, R., Klein, M. 2001. Evaluating Software Architectures: Methods and Case Studies. Addison-Wesley.

[9] Ekelhart, A., Fenz, S., Klemen, M., Weippl, E. 2007. "Security Ontologies: Improving Quantitative Risk Analysis," Proceedings of the 40th Annual Hawaii International Conference on System Sciences, IEEE Press.

[10] Feng, Q., Kazman, R., Cai, Y., Mo, R., Xiao, L., "An Architecture-centric Approach to Security Analysis", *Proceedings of the 13th Working IEEE/IFIP Conference on Software Architecture (WICSA 2016)*, (Venice, Italy), April 2016.

[11] Ferguson, J. 2007. "Understanding the Heap by Breaking It", https://www.blackhat.com/presentations/bh-usa-07/Ferguson/Presentation/bh-usa-07-ferguson.pdf

[12] FICO, "Using Predictive Analytics to Advance Cybersecurity", http://www.fico.com/en/blogs/fraud-security/using-predictive-analytics-advance-cybersecurity, retrieved Jan. 2016.

[13] Hannan, M.T., and Freeman, J. 1989. Organizational Ecology. Harvard University Press.

[14] IEEE Center for Secure Design. 2016. "Avoiding the Top 10 Software Security Design flaw," https://www.computer.org/cms/CYBSI/docs/Top-10-Flaws.pdf, retrieved March 1, 2016.

[15] Kazman, R., Cai, Y., Mo, R., Feng, Q., Xiao, L., Haziyev, S., Fedak, V., Shapochka, A. 2015. "A Case Study in Locating the Architectural Roots of Technical Debt," in Proceedings of the International Conference on Software Engineering (ICSE), Florence, Italy.

[16] Kazman, R., Goldenson, D., Monarch, I., Nichols, W., Valetto, G. 2016. "Evaluating the Effects of Architectural Documentation: A Case Study of a Large Scale Open Source Project", IEEE Transactions on Software Engineering.

[17] Mo, R., Cai, Y., Kazman, R., Xiao, l. 2015. "Hotspot Patterns: The Formal Definition and Automatic Detection of Architecture Smells," Proceedings of the The 12th Working IEEE/IFIP Conference on Software Architecture.

[18] Mundie, D.A., McIntire, D.M. 2013. "The Mal: A Malware Analysis Lexicon," Software Engineering Institute Technical Note CMU/SEI-2013-TN-010.

[19] Obrst, L., Chase, P., Markeloff, R. 2012. "Developing an Ontology of the Cyber Security Domain," Proceedings of the STIDS: MITRE Corp., 49-56.

[20] Ozment, A., Schechter, S.E. 2006. "Milk or Wine: Does Software Security Improve with Age?," Proceedings of the Usenix Security Symposium.

[21] PWC. 2015. "Managing cyber risks in an interconnected world", http://www.dol.gov/ebsa/pdf/erisaadvisorycouncil2015security3.pdf.

[22] Ryoo, J., Kazman, R., Anand, P. 2015. "Architectural Analysis of Security Vulnerabilities", IEEE Security and Privacy, September/October 2015.

[23] Seacord, R.C. 2005. Secure Coding in C and C++, Addison-Wesley.

[24] Smith, A., Humphreys, M. 2006. "Evaluation of Unsupervised Semantic Mapping of Natural Language with Leximancer Concept Mapping," Behavior Research Methods (38:2), 262-279.

[25] Smith, A., Malik, S., Shneiderman, B. "Visual Analysis of Topical Evolution in Unstructured Text: Design and Evaluation of TopicFlow" in Applications of Social Media and Social Network Analysis, Springer, 2015.

[26] Solomon, M. "Predictive Analytics: Using the Past to Create a More Secure Future" http://www.securityweek.com/predictive-analytics-using-past-create-more-secure-future, September, 2014

[27] Swanson, E.B., Ramiller, N.C. 1997. "The Organizing Vision in Information Systems Innovation," Organization Science (8:5), 458-474.

[28] Xiao, L., Cai, Y., Kazman, R. 2014. "Design Rule Spaces: A New Form of Architecture Insight," Proceedings of the Proceedings of the 36th International Conference on Software Engineering, 967-977.