

IQT

QUARTERLY

VOL. 6 NO. 4

SPRING 2015



cybersecurity: # reboot

Identify. Adapt. Deliver.

TABLE OF CONTENTS

On Our Radar By Nat Puffer	02
A Look Inside: Cybersecurity Reboot	05
Building Trust in Insecure Code By Jeff Williams	06
No Silent Failure: The Pursuit of Cybersecurity A Q&A with Dan Geer	10
The Return of Dragons: How the Internet of Things Is Creating New, Unexplored Territories By John Matherly	14
The Insecurity of Things By Stephen A. Ridley	17
Memory Forensics for Malware Detection and Analysis By Vico Marziale	21
Tech Corner A technology overview from IQT portfolio company ReversingLabs	24
From the Portfolio	29

ON OUR
RADAR

On Our Radar

By Nat Puffer

The fight to secure information is being lost.

This was at the center of a debate within In-Q-Tel's Infrastructure and Security Practice last summer. Every day we would hear about another epic breach of consumer data or critical company secrets. In an industry that uses analogies as a foundation, all of them have crumbled. The "castle perimeter" has crumbled. The "hunters" are not efficient enough to seek out the persistent and embedded foothold of determined adversaries. A 2013 article cited Mandiant as discovering a foothold in a network that went undiscovered for six years and three months.¹ The same article cited the average time from initial breach to discovery as 229 days. If this isn't an indication of critical systemic failure, I'm not sure what is.

On the other side of the argument, "losing" means something is lost, and implies that there's some idea what winning would look like. But what if we haven't really lost anything? Perhaps breaches and data theft, like power outages or rush hour traffic, are just part of living in the modern world. While the numbers vary for effect, Target was reported to have lost 40 million payment cards in 2013; Home Depot lost 56 million payment cards in 2014; and Anthem lost 69 million patient records in 2015.² Yet every time one of these breaches is announced, we go about our days like nothing has happened. We still use our credit cards at Target. We still use our health insurance. We've become accustomed to the idea that companies can't keep our information secure against a sophisticated attacker, and in some cases, even an unsophisticated one.

If we get substantially more out of modern conveniences than they cost us, how is that a loss? Stated differently, isn't instantly streaming movies to a tablet worth having to change a credit card number from time to time?

At the core of the tradeoff is that people on a day-to-day basis value convenience over security. We shouldn't be surprised that this extends to people who develop software; that security lags behind innovation. In 2011,

Steve Yegge wrote a blog post at Google capturing this concept, and I cite this post frequently to demonstrate the nature of the information security problem. Yegge asserted that there are two competing forces as systems are developed. The first is how functional and useful the system is, dubbed accessibility. The second is security. Without any external influence, accessibility will always win at the cost of security. As stated in Yegge's post, "[D]ialing Accessibility to zero means you have no product at all, whereas dialing Security to zero can still get you a reasonably successful product such as the PlayStation Network."³

It's a strange coincidence that it was also a Sony breach that brought lapses in security to the attention of the general public. Was the most recent breach of Sony Entertainment a game changer? If you are an information security professional, your likely answer is a jaded "no." We have been here before. There was nothing extraordinary about the breach technically. The concept of nation-state attackers penetrating corporate systems has appeared in commercial penetration testing reports since 2004. The only change, if any, is the rate at which we have to respond to new threats with limited resources. If agility is defined as the ability



Without any external influence, accessibility will always win at the cost of security.

to minimize the time required to accomplish a different task, we're finding agility is critical.

However, if you haven't been living in information security or worried about operating system internals and the ways they can be exploited (i.e., the majority of people), your perspective may be that the Sony breach signals a major change. There is a wider awareness that system breaches aren't just about criminal gangs looking to steal credit cards anymore. There is a new world where countries use the Internet as a platform for attack or retribution. This was highlighted by the White House Press Corps asking the President directly about a computer breach of a private company, and being assured that appropriate action would be taken.⁴ Similar questions have been echoed in board rooms and C-suite conversations across the country: How would we fare? What's at risk? What are we doing differently today given these revelations?

The next steps after those questions are even more interesting. If you believe that all the investment in current mitigation strategies has been working, something has changed and you need to adapt. If you think that all your investments have failed, it's time to change course. In either case, something needs to change. The constant stacking of security tools hasn't worked. The doctrine of Defense in Depth has to answer for its cost of complexity when measured against a lack of success. While we may not want to abandon the idea altogether, we need to adapt with a new way of operating. This will likely include variable response, situational awareness, and security orchestration.

Belief in Variable Response

Endless stacking of security safeguards is a flawed strategy and every asset cannot be protected equally. For a long time there was a belief that you could secure everything in a reasonably complex network if you put more resources and tools against the problem. This resulted in a limited staff chasing every possible

indicator of compromise with equal veracity, which is exhausting. That exhaustion bore little fruit, which bred complacency. That complacency, over time, led to the very thing you wanted to prevent.

Going forward, companies will be faced with developing doctrine based on hard questions. What assets are priorities? Which systems would potentially be sacrificed for others? Is it okay to isolate and restore a user's desktop automatically based on a machine's determination of a threat? The user would lose his or her work, but you would limit chasing ghosts through the system.

At the root of this are questions surrounding "How?" How do I know what to prioritize? How can I create the maximum effect with the minimal resources? Developing a sense of situational awareness to focus operations will be critical in answering those questions.

Situational Awareness

Over the past year and a half, IQT has been tracking the rise of the private threat intelligence market. State of the Internet and Annual Breach reports have existed for a long time, but this space was something new. Vendors were looking to increase the rate and specificity of intelligence into something actionable, forming two camps.

The first camp produced finished products. These were full reports with in-depth analysis and attribution. Actors, campaigns, tools, techniques, and procedures are all tracked and discussed, with fun names created for many of them. There was a focus on attribution, which we weren't convinced would be useful for those without certain authorities. The fact that activity was supposedly part of a People's Liberation Army (PLA) initiative may be interesting, but the overall economics of a private company are more of a driving factor in dealing with China than an IT department's report for the CIO. However, as the Sony breach has raised awareness inside commercial enterprises, so has the

value of attribution. Understanding the larger context of “Who” is suddenly as important as “What” or “Why” even if prosecution isn’t an option.

The second camp produced machine-readable Indicators of Compromise (IOC). This information was intended to enrich the detection systems within a network through a couple of mechanisms. First, IOCs could be fed into a central incident management solution. The goal was to correlate indicators like IP addresses with existing log data to determine if network traffic that would normally appear benign is something to worry about. The second use was to add IOCs to detection capabilities to search out new potential compromises.

Both are useful and have a place. We expect to see significant activity as vendors from one camp add capabilities from the other, collection biases in feeds are worked through, and infrastructure for delivery, consumption, and sharing is rolled out. The promise is that focus and prioritization will eventually result from the ability to put global context around local alerts. The reality is that it may be too early to tell if the intelligence is sustainable and provides efficiency, or becomes an overwhelming deluge of data. Ultimately, the indicators need to support and drive action in the organization.

Action Through Security Orchestration

Awareness without the ability to act is useless. Action that consumes all your resources is limiting. The ability to make decisions on the intelligence we have and act with minimal resources in a timely manner is the goal, but has eluded most complex IT organizations. Networks of sufficient complexity are typically built by several generations of employees using products from a number of vendors spanning years. Legacy systems, acquisitions, new initiatives, and day-to-day break-fix cycles consume any available time personnel might

have otherwise put towards automation on a system-wide scale. Vendors have added to the problem with protection of their market share by limiting centralized management solutions to work only with products in their portfolio. Bring your own device (BYOD) policies and the Internet of Things (IoT) pile on even more complexity and amplify the problem.

Several third-party vendors are chasing this problem with fresh eyes. The core concept is more in line with a knowledge-based system or workflow management solution than something fully autonomous. Identify actions that are frequent and time consuming, streamline the decision making and sign-off process, and efficiently execute across the infrastructure. While limited, this ability to act, potentially in machine time, on identified threats that are based on a broad view and defined prioritization might begin to tilt the odds back in our favor. However, the ability to capture business process and maintain automation as vendors update their solutions and companies change their infrastructure has been the downfall of these solutions before. These solutions will succeed based on their ability to capture and keep pace with changes in doctrine and business processes.

Rebooting Cybersecurity

Perhaps we need to restart, building upon what we have with variable response, situational awareness, and security orchestration in mind. Doctrine and awareness enable action that can drive up the cost for the attacker by constantly removing their footholds while keeping the defender’s resources relatively flat. Tracking, identifying, and attributing attacks changes the political landscape. If winning in this context means establishing a livable equilibrium, changing the economics and politics of the attacker may be a good first step. **Q**

Nat Puffer is a Senior Member of the Technical Staff within In-Q-Tel’s Infrastructure and Security Practice. He has led investments in network security, data storage, and cloud computing. Prior to IQT, Puffer was with Knowledge Consulting Group, where he was responsible for growing the Cyber Attack and Penetration Division. He previously held consulting roles with Neohapsis and Symantec. Puffer earned a bachelor’s degree in Integrated Science and Technology and a master’s degree in Computer Science from James Madison University.

REFERENCES

- ¹ Lennon, M. (2014, April 10). *Just One-Third of Organizations Discover Breaches on Their Own*. Retrieved from SecurityWeek: <http://www.securityweek.com/just-one-third-organizations-discover-breaches-their-own-mandiant>
- ² Palermo, E. (2015, February 6). *10 Worst Data Breaches of All Time*. Retrieved from Tom’s Guide: <http://www.tomsguide.com/us/biggest-data-breaches,news-19083.html>
- ³ Yegge, S. (2011). *Steve’s Google Platforms Rant*. Retrieved from <https://plus.google.com/+RipRowan/posts/eVeouesvaVX>
- ⁴ *Remarks by the President in Year-End Press Conference*. (2014, December 19). Retrieved from Whitehouse.gov: <http://www.whitehouse.gov/the-press-office/2014/12/19/remarks-president-year-end-press-conference>

A Look Inside: Cybersecurity Reboot



Cybersecurity is at an inflection point; many of the systems designed over the past two decades are in various states of disarray, and the industry is scrambling to find solutions to myriad problems. Networks are vulnerable, as evidenced by a variety of attacks on prominent companies and government organizations. Traditional security challenges are further exacerbated by the emergence of the Internet of Things (IoT) and the explosion of connected devices.

Jeff Williams of Aspect Security opens this issue with a discussion on trusting software with critical vulnerabilities. Williams suggests using instrumentation-based application security tools, which are more scalable and powerful than legacy approaches.

An interview with IQT's Dan Geer explores the state of cybersecurity through multiple angles, including Internet protocols, the role of identity, and challenges with personally identifiable information.

Next, John Matherly of Shodan examines the IoT's security implications. Shodan, a search engine for Internet-connected devices, detects things like power plants, control systems, smart TVs, and cameras that are often ripe for exploitation. Efforts like Shodan have raised IoT security awareness, but we are still in the early stages of testing, maintaining, and securing industrial and consumer connected devices.

Stephen A. Ridley of Xipiter continues the discussion on IoT security, positing that the threat landscape is shifting from an emphasis on servers and endpoints to everyday devices. Ridley's research at Xipiter has surfaced a divide between software and hardware security, which gives rise to vulnerabilities in IoT devices as well as broader computer networks that rely on embedded systems.

Vico Marziale of BlackBag Technologies explores methods for malware detection and analysis. Modern malware is becoming increasingly complex, and using the newest tools, such as memory forensics, is an essential piece of a cybersecurity reboot.

Finally, we close the issue with a technology overview from IQT portfolio company ReversingLabs. The company's hashing algorithm uses a new method of intelligently hashing a file's features rather than its bits, making it orders of magnitude more effective than traditional tools for detecting malware.

Cybersecurity remains a critical focus area for both industry and government, as evidenced by recent activities ranging from legislation to breaches. We hope that this issue of the *IQT Quarterly* heightens cybersecurity awareness and encourages readers to think critically about how to approach and protect against the growing set of cyber challenges. **Q**

Building Trust in Insecure Code

By Jeff Williams



For at least two decades, we've been sprinting to automate all aspects of government, defense, business, and our personal lives, and this trend isn't likely to change. So here's the problem: We now have a mountain of insecure code, we aren't very good at building new code securely, and the attackers are starting to take advantage.

The cybersecurity situation is untenable. There is no way that society will give up on automation. We always seem to choose short-term progress over uncertain future danger. So we have no choice but to trust software with everything important in our lives, even though we know that software has critical vulnerabilities.

The Insane Complexity of Modern Software

If we're going to do something about cybersecurity, the first step is to take a hard look at the real facts about our software.

After details of the recent JPMorgan Chase & Co. breach were revealed, a former employee told *The New York Times* that it was as if the attackers "stole the schematics to the Capitol — [JPMorgan] can't just switch out every single door and window pane overnight."

Unfortunately, the reality is considerably worse than that. It took JPMorgan decades to create its software infrastructure, and there simply is no practical way to replace it. The Capitol analogy is far too small. This is like discovering that an entire city has been built with

asbestos insulation, lead pipes, and unshielded power lines. And they all need to be swapped out without disrupting service.

A typical midsized financial organization or government agency has a software portfolio of more than 1,000 applications. The largest of these portfolios can exceed 10,000 applications. Each of these applications, on average, has hundreds of thousands of lines of custom code, and the largest can have more than 10 million lines.

Each application also includes anywhere from dozens to hundreds of software libraries, frameworks, and components. The total size of these components is typically ten times the size of the actual source code. This third-party code runs with the full privilege of the applications that use it, but without any good way to figure out who wrote it or whether it harbors undiscovered vulnerabilities or backdoors.

By way of comparison, consider the U.S. Federal Tax Code, which has also grown dramatically over the years. Currently, the tax code totals 4.4 million lines of "code,"

which is roughly 73,954 printed pages. This is the size of a handful of applications. Finding all the loopholes in the tax code would be a difficult job for a team of lawyers, but finding all the vulnerabilities in a modern enterprise is hundreds of times harder.

So after two decades of high-speed coding, a typical large financial organization has accrued an astonishing pile of code — almost 1 billion lines of code. And these portfolios are growing rapidly — typically by 20 percent each year.

We Need Unified Application Security Tools That Scale

Today, there are two completely separate domains of application security tools. In development, tools like static analysis security testing (SAST), dynamic application security testing (DAST), and component lifecycle management (CLM) are intended to detect vulnerabilities so they can be prevented early. On the other hand, production tools, such as intrusion detection systems (IDS), next generation firewalls, data loss protection (DLP) tools, and web application firewalls (WAF) attempt to detect and prevent attacks and breaches.

Neither of these legacy approaches to automating application security provides much protection. Simply put, these technologies do not have enough context to provide accurate security analysis or protection. They are also quite difficult to use, requiring experts to onboard applications, run scans, and interpret results.

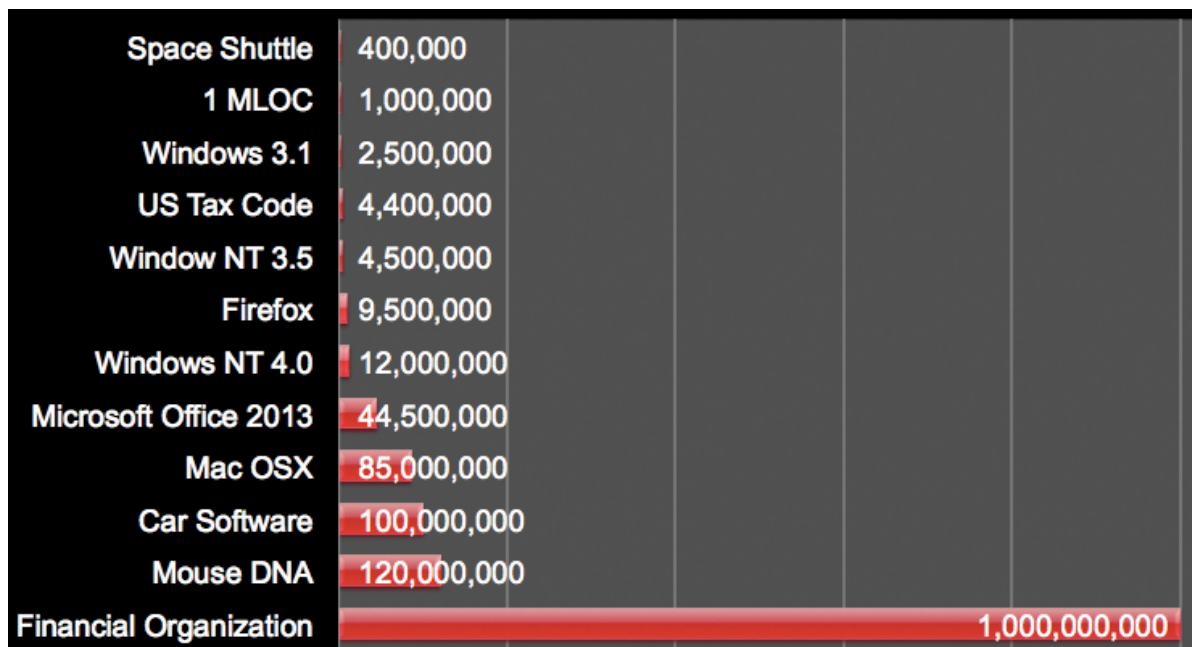
The false alarms from these tools create significant wasted work, broken applications, and animosity towards security. Even worse, involving experts creates a process bottleneck that discourages projects from cooperating with security.

Ideally, development and operations would work together to achieve application security. Fortunately, the analysis required to detect and prevent vulnerabilities, attacks, and breaches is fairly similar. A software vulnerability is like an open window. An attack is like someone climbing in that window. And a breach is like someone exiting the window with stolen valuables.

Recent advances are unifying the ability to detect and prevent vulnerabilities, attacks, and breaches into a single technology that can be deployed inside applications, as opposed to perimeter protections that lack the appropriate context to protect applications. These products leverage advances in software instrumentation to weave security directly into applications.

The Power of Security Instrumentation

Security instrumentation is a technology that allows both sensors and defenses to be deployed inside a running application without any modifications to source code or development processes. The sensors allow both vulnerabilities and attacks to be detected quickly and accurately, with the full context available inside the running application. The defenses allow the application to prevent vulnerabilities from being accessed and to



Lines of code in common applications and systems.

Source: <http://www.informationisbeautiful.net/visualizations/million-lines-of-code/>

Detect and prevent vulnerabilities



Detect and prevent attacks



Detect and prevent breaches



block real attacks. In essence, instrumentation allows you to enhance applications by enabling them to detect their own flaws and defend themselves against attacks.

Using instrumentation-based application security tools (interactive application security testing, or IAST) is quite different than static analysis, dynamic analysis, web application firewalls, data loss protection, or any other legacy application security technology. With instrumentation, IT teams enable their application servers with a security agent in development, test, staging, and production. During startup, the security agent weaves vulnerability analysis, attack detection, and security defenses directly into the application itself — enabling a sort of self-protection. From that point forward, the applications are continuously protected and report any interesting security information to a central console.

The sensors directly monitor the running application, providing the agent with unparalleled intelligence about security in the running application. Static and dynamic tools have very limited information, but instrumentation yields access to everything on the server, including the code, runtime data, control flow, libraries, frameworks, configuration files, and back-end connections. All this contextual information allows extremely accurate vulnerability and attack analysis. Although the analysis happens continuously and in real time, there are no size limitations on the applications to be analyzed and the analysis includes all third-party libraries, frameworks, and components.

A security agent can be used in development, test, quality, and production environments at the same time. All the agents report to a central console that allows analytics and management across the entire application portfolio. The console allows all stakeholders in the security of an application to gain a clear view of vulnerabilities, attacks, and defenses across the company. These analytics enable informed decision making and strategic investment in security.

This approach is a significant change from the current model, which is essentially to perform periodic scans and penetration tests on one application at a time and deliver a PDF report.

Continuous Application Security

Applications are already being instrumented for performance reasons. Everyone understands the benefits of continuous performance monitoring from within the application. The only way the application security community can scale and keep pace with rapidly advancing threats is by adopting this new continuous paradigm.

When security instrumentation agents become part of an organization's standard server builds, businesses can take an entirely different approach to application security. The agent can identify and prevent vulnerabilities during development as well as identify and stop attacks and breaches in operations.

Let's examine how a typical organization can use an instrumentation-based, continuous application security approach to secure a portfolio of applications:

AUTOMATIC INVENTORY: Each time a new development projects starts, an agent on the development environment server starts profiling the application and reports security telemetry to a centralized application security console. The new application gets added to the organization's application portfolio, along with basic information about the codebase, software architecture, libraries, and frameworks.

SECURE CODING: As development continues, coders get instant feedback about the vulnerabilities in the code they are developing and testing. When they make a mistake that introduces a flaw, they get microtraining in exactly what they did wrong, why it's important, and how to fix it. With instantaneous security coaching, developers are empowered to commit clean code to their source code repositories.

TEST AND STAGING: In test and staging environments, the agent continuously analyzes the security of applications to identify vulnerabilities without requiring any security expertise or access to source code. This stage provides a double check to ensure that no insecure code was checked in and that no complex interactions created a new vulnerability. The agent verifies all the libraries, frameworks, and other third-party code in use to make sure that there are no known vulnerabilities. The agent also verifies that libraries don't contain zero-day flaws from being improperly invoked by developers.

INTELLIGENCE AND ANALYTICS: The agent maintains a real-time application security dashboard for every application that is always up-to-date. There is no need to schedule scans because the agent is part of the application. This allows compliance efforts and other security activities to be much more efficient. Executives can keep abreast of application security across divisions to ensure that the company is properly protected.

ATTACK DETECTION: In operations, the agent shifts to a more defensive posture. The agent now seeks out attacks and breaches. The agent sees attacks in context, from within the running application, making attack and breach detection very accurate. All attacks generate detailed logs, alerts, and notifications via existing channels.

ACTIVE DEFENSE: The agent also defends the application by deploying narrowly tailored defenses for a specific attack on a specific application, or broad protection for a range of attacks across the entire portfolio. These defenses are enabled early in development so that they are known to be safe for production.

REAL-TIME RESPONSE: When a new vulnerability is discovered in a third-party library or a new class of vulnerability is discovered, the agent is automatically updated with new rules and reports problems immediately. This eliminates the need to rescan the

organization's entire software portfolio when new security problems like Heartbleed are uncovered.

By weaving vulnerability and attack detection and prevention into the application itself, instrumentation empowers applications to defend themselves. The protection moves with the application from development to production, from internal networks to the Cloud, and across all types of applications.

Instrumentation is the Future of Trusting Insecure Code

Security becomes increasingly difficult as our applications get more connected, more complex, and more critical. We desperately need new scalable approaches to protect businesses and government from cyber attacks.

Security instrumentation is simultaneously simpler and more powerful than the array of disconnected products on the market today. The time has come to stop scanning, stop firewalling, and stop attempts to do security work at the perimeter. Instead, we can add security analysis and protection directly to applications, protecting them accurately and efficiently wherever they run.

Gartner has recognized the power of the instrumentation approach. They call both IAST and runtime application self-protection (RASP), "must-have security technologies." Gartner application security expert Joseph Feiman says, "Applications can be better protected when they possess self-protection capabilities built into their runtime environments, which have full insight into application logic, configuration, and data and event flows. RASP technology is emerging to offer these capabilities and fulfill these demands."

Security instrumentation is a revolutionary way to augment mountains of insecure code with strong security defenses, so that these applications deserve the trust that people, companies, and government agencies have already placed in them. **Q**

***Jeff Williams** brings more than 20 years of security leadership experience as co-founder and CTO of Aspect Security. He holds three patents in instrumentation-based application security and is founder and CTO at Contrast Security. Williams is also a founder and major contributor to OWASP (The Open Web Application Security Project), where he served as the Chair of the OWASP Board for eight years and created the OWASP Top 10, OWASP Enterprise Security API, OWASP Application Security Verification Standard, XSS Prevention Cheat Sheet, and many other widely adopted free and open projects.*

No Silent Failure: The Pursuit of Cybersecurity

A Q&A with Dan Geer

On the scale of the Internet, starting fresh just isn't an option. It has also proven more difficult to transition legacy systems at that scale. An obvious example is the impending exhaustion of the IPv4 address space not being a significant catalyst for system-wide adoption of IPv6. Is the Internet as it exists today fundamentally flawed and un-securable?

This is both a design question and a practicality question. On the design side, the most important ARPAnet technical decision made was and is the end-to-end principle. We would not have what today we call the Internet were it not for the design freedom that precluded putting policy in the network fabric itself. Policy in the network fabric, be it security policy or any other policy for that matter, rewards stasis and rewards bigness (which is a bit redundant, to be sure). When my team was designing what became the X Window System, our mantra was "all mechanism, no policy." When you opt, at the fundamental design level, for a focus on mechanism and an avoidance of policy, you magnify the realm of the possible. When you opt for policy over mechanism, you magnify the hegemony of those who set policy.

That makes the question of whether the Internet is or is not un-securable moot, moot in the sense of whether a fish without a bicycle is less of a fish. The protocol design that brings us the Internet as we colloquially know it does not address security, per se. On the other hand, its operation as a digital artefact very much depends on such hard-to-secure services as naming and routing. Both naming (DNS as the prime example) and routing (likewise BGP) have been areas of attack, and there have been metaphorical casualties including casualties from what can only be called friendly fire. Nevertheless, it is only with services and the ends (in the end-to-end sense) that any duty of security lies. The job of the network, per se, is to deliver bits, and to do so in a way that is maximally tolerant to random faults in the network fabric.



It is pretty clear that the duration of overlap between IPv4 and IPv6 will be long. As their security profiles mismatch, we will learn something about those mismatches, e.g., is IPv6 multihoming likely to produce inadvertent IPv4 bridging?

Even if there were a better design for the Internet, one with more secure foundational protocols, have we reached a point where transition and backwards compatibility with current systems prevent meaningful progress towards security?

Again, this depends on whether the job of The Network is to make people safe, or is it the job of those who offer services to make their counterparties safe? Backward compatibility, "legacy drag" if you prefer, is always the burden of the early adopters as evidenced, for example, by the number of third world countries that will never have wireline telephony and who, therefore, have no backward compatibility issues like compatibility with circuit-oriented design.

The fundamental requirement for a "secure Internet" is little different than would be found in some other area of human endeavor, e.g., postal mail. For good and realistic reasons, an envelope pays a higher rate of postage than does a postcard and for that price differential it is "sealed against inspection." Were trucking companies denied common carrier designations across the board, and were instead responsible for every package's contents, both the quantity of shipped goods and the cost effectiveness of shipping would bear no resemblance to today's economy. We can go there, of course, as all it requires is a President with a phone and a pen, but it is unlikely.

In a sense, the widespread adoption of virtual private networks is exactly the provision of security — inside a walled garden network with checkpoints at the borders thereof. Put differently, the security of self that is found inside a large, single-tenancy commercial campus is of the same spirit; you have little freedom there, can only enter and exit in surveillable ways, and enjoy a greater degree of safety than you do anywhere else. In short, if someone wants a secure network, then layer it on the Internet we have (SIPRNet, anyone?).

A central component of secure systems is often identity, especially as it relates to authentication and authorization. Broadly, the Internet seems to have those resources which rely on your true identity: banking, utilities, shopping; and those that value anonymity or avatars: forums, select social media, etc. Can we enhance the assurance of the former without stripping away the freedom of the latter? Is this a necessary step in securing the Internet?

Yes, but only insofar as we are willing to admit (to ourselves as well as in policy) that we each have different identities and that our freedom depends on being able to keep identity roles separate — parent, employee, hobbyist, colleague, etc. In an odd way, the constant labor within classified organizations to rightly ascertain "need to know" is being now played out in the arena of general society to a degree never before seen, both as to geographic scope and to real-time concurrency. Yes, of course, in a small enough town everyone knows everyone else's business, but the restraint of mutual interest and having to live together within the town's confines exerts a brake on runaway misuse of information ("Yeah, I heard that, but it's none of my business," or even, "Do unto others as you would have them do unto you").

A unitary identity, one issued by a central authority, is already the case in less free locales. The debate over whether to go that way in the U.S., such as via RealID extended to the Internet, is, however, likely soon to be irrelevant. As the sensor fabric of daily life grows, identity becomes not an assertion ("my name is Dan") but an observable ("sensors agree that that is Dan"). Arguing over whether or not to show identification and from what authority may be well on its way to "fighting the last war."

There has been a notable shift in the targeting of data stolen by cyber criminals. While credit cards are

still sought after, health records and social security numbers seem to be targeted more often and stolen in greater numbers. The rationale is that the viable lifetime of a credit card is relatively short compared to a social security number combined with other personally identifiable information (PII). Are we looking at a forced requirement to a new universal identifier as more identifying information is stolen?

One might argue precisely the opposite, that "one ring to bind them all" (or one identity to tie all others together) increases risk. The question would be whether a unitary identity induces or precludes cascade failure in the event of compromise. We certainly have the raw technologic capability to make biometric identification the core, but biometrics are only safe if the biometric never leaves the device on which it is used (because biometrics cannot be re-issued, they must never be exposed to a need to do so).

The case of medical data, which is to say the mandate for electronic health records, is particularly fraught. The malpractice liability system requires records be kept by the provider and, therefore, the full record for a given patient will never be unitary — each provider will retain original records as only in so doing will the provider be able to give evidence in any future litigation. This, in turn, generates a need for a common identifier, just as you suggest, to tie the component records together into "the whole picture" or, at least, to be able to do so in principle even if daily practice does not assemble that unitary record. It seems likely that discovery motions will have to be couched in a unitary identifier if they are to be meaningful and not, themselves, a cause of unwarranted disclosure of the records of other, unrelated parties.

In the case of medical data, however, the trump card is held by the insurer who will doubtless enforce unitary identification as a condition of coverage. Where the insurer of last resort is a unit of government, the urge to reuse an existing identifier will be irresistible (e.g., the Social Security number reconfigured as the Medicare claim number through the addition of one or two alphanumerics).

Part of the issue with breaches of PII is their ubiquity across critical service providers. In many cases, corporations are actively and liberally sharing information amongst themselves, notably through monetizing data provided to advertisers. Even if there were a more secure token for identification, does the

underlying data that is collected and shared pose an equal risk to an individual's security on the Internet? Are we so focused on the PII that we are missing the larger concept of identity derived from our collective observed actions on the Internet?

It can be said that during the 1990s the commercial sector caught up with the military sector in the application of cryptography — not its design but its application. In like sense, the commercial sector is fast catching up with the military sector in the capture and fusion of traffic analysis data. The core of almost all behavioral analysis systems is monetizing merged information collected in the same spirit as military traffic analysis. The number of companies promising that retailers can better understand their own customers is growing fast. JavaScript-based analytics, such as from Google, adorn nearly every merchant website (The HTTP Archive reports that on the wire JavaScript traffic is five times the byte count of HTML traffic).

It is perhaps worth noting that any customer of a website can be tracked independent of identification or not; it is only when the customer goes to pay that there is a name put to whatever click-tracking might have been done. In the early 2000s, many attempts were made to configure electronic commerce transactions such that the credit card issuing bank did not know what the website customer was buying, while the web merchant did not know what payment mechanism was being used by the website customer, but everyone remained assured of their part of the transaction (another manifestation of bringing "need to know" out of the military into the commercial). All those attempts failed — merchants would have none of it because doing the transactions that way made the process of accepting exchanges and returns much more difficult. That the buying customer insisted on the right to return merchandise duly bought caused the commercial sector to fail to take a very important security (and privacy) step, even then — when installed base was tiny compared to today.

The Internet of Things (IoT) is not only expanding the scope of what we consider Internet-connected devices. We seem to have an accelerating dependence on the connectivity and operation of things with embedded software that are limited in protections with an extended lifetime and are loosely managed. These systems are also being built on commodity hardware with open

source foundations for the firmware. While history has shown that novel security enhancements are often temporary impediments to the attacker, do you believe that the adoption of an alternative computing architecture, considering both performance and security, is necessary to prevent whole classes of attacks?

Let's do a worked example: let's count cores in the Qualcomm Snapdragon 801 processor. The central CPU is four cores, the Adreno 330 GPU another four, video out is one more, the Hexagon QDSP is three, the modem is at least two and most likely four, and Bluetooth is another one, as is the USB controller and the GPS. The Wi-Fi is at least one and most likely two, and none of this includes charging, power, or display. That makes somewhere between 18 and 21 cores. In the vocabulary of the IoT, I ask whether that is one thing or the better part of two dozen things? It is pretty certain that each of those cores can reach the others, so is the perimeter to be defended the physical artefact in the user's pocket or is it the execution space of each of those cores separately?

I looked at seven different estimates of the growth of the IoT as a market phenomenon — everything from smart electric meters to networked light bulbs to luxury automobiles, and the median is a compound annual growth rate of 35 percent. If perimeter control is to remain the paradigm of cybersecurity, then the number of perimeters to defend in the IoT is doubling every 17 months.

The only two strategies that have caught my eye are 1) work coming out of the University of Pennsylvania that makes a case for software release so rapid that opponents do not have time to train on one version before it is obsoleted, and 2) work by a startup in Boston that allows binaries to be scrambled in a way stronger than address space layout randomization (ASLR). Put differently, the targeted software does not live long enough to attack or the targeted software is mutated out of all recognition by the attacker. Both aim for a massive reduction in the vulnerable base not by enhanced programming skills, but by removing the economic advantage attackers enjoy within the ever more massive numbers of things that are network reachable.

As a side note, open source software will never displace closed source software where performance expressly matters. Quoting from a mailing list thread that makes the point:

My database cluster runs on commodity \$8k servers. Depending on the workload details, my software is

*5-100x faster on that hardware than anything in open source. This means that if I was to use open source, it is equivalent to me spending at least \$40k *per server* in additional costs to achieve performance parity. It is an arbitrage opportunity for commercial software vendors, and enterprises are doing the math.*

*It is becoming increasingly common for people to replace a million dollars of hardware and "free" software with \$100k worth of hardware and \$500k software. Open source is so inefficient for many use cases that using it is prohibitively expensive despite the lack of price tag. It is expensive to run a 120-node Cassandra cluster when you can replace it with a 20-node cluster *running on the same cheap hardware* without any loss of performance or throughput.*

We seem to be moving towards a world where the Internet saturates daily life and becomes interwoven into the things all around us. The smartphone may be at the forefront of this phenomenon. It is generally accepted that these phones carry an identifier, can be tracked, and are managed to some degree by the carrier. Are we facing a similar choice with the broader range of IoT devices; that they should each carry an identifier that can be centrally located and generally managed?

This is a horse that is already out of the barn. By mandate, newish cars have tire pressure sensors that each and severally have unique Bluetooth radios. Were I laying IEDs roadside, I'd certainly listen for your tire's Bluetooth signature as a detonation signal. Closer to home, if no one device actually identifies you as you, the confabulation of all your devices certainly will. Any time you put up a radio, the only remaining question for the traffic analyst is what is the quality of the antenna, and we are putting up lots of radios (e.g., Fitbit and friends). Our laws say that were I to take your photograph in the public street you would have no cause to complain. Is

there any reason to believe that that absence of a cause to complain is wavelength specific? If you are giving off photons in the visible spectrum, you've no recourse to my wallet (via tort). If you are giving off photons in the infrared spectrum, my porch light will turn on if my stoop is close enough to the sidewalk. If you are giving off photons in the microwave spectrum, is there any reason that I cannot observe them? Your heart is giving off a weak but detectable radio signal and there is already a commercial product to identify you via that signal which, like all things biologic, is unique.

Perhaps that is the point, with close enough inspection, everything is unique and thence a source of identification. The camera on your cellphone is not perfect and therefore marks pictures taken by it as much as the barrel on your long rifle marks bullets that pass through it. What the IoT is doing is make "close enough inspection" a default condition, not a remarkable one.

The hardest question with IoT devices might however be this: Should they have a remote management interface? If they do not, then there is no way to solve problems (like security problems) discovered after deployment. If they do, then that interface becomes the battleground to end all battlegrounds, as the most expert attacks are ones that pull up the ladder once they are on board. At the very least, devices that cannot be updated and which are immortal are an existential threat, an accumulating one. We will soon be doing a, shall we say, natural experiment as self-driving cars with over-the-air software upgrade capabilities proliferate. Automobiles are heavily regulated, so unhappy outcomes that can be mitigated with money transfer will be mitigated with money transfer. If I am correct that cybersecurity is irretrievably offense-dominant, then all sovereigns who can will invest in offense for battle and disconnected operability for defense. Were this not already obvious, the IoT makes it more so. **Q**

Daniel E. Geer, Jr., Sc.D., is Chief Information Security Officer at In-Q-Tel. An entrepreneur, author, scientist, consultant, teacher, and architect, Dr. Geer's career milestones include: *The X Window System and Kerberos* (1988), the first information security consulting firm on Wall Street (1992), convener of the first academic conference on electronic commerce (1995), the "Risk Management is Where the Money Is" speech that changed the focus of security (1998), the Presidency of USENIX Association (2000), the first call for the eclipse of authentication by accountability (2002), principal author of and spokesman for "Cyberinsecurity: The Cost of Monopoly" (2003), co-founder of SecurityMetrics.Org (2004), convener of MetriCon (2006-), author of "Economics & Strategies of Data Security" (2008) and of "Cybersecurity and National Policy" (2010), and creator of the Index of Cyber Security (2011) and the Cyber Security Decision Market (2012).

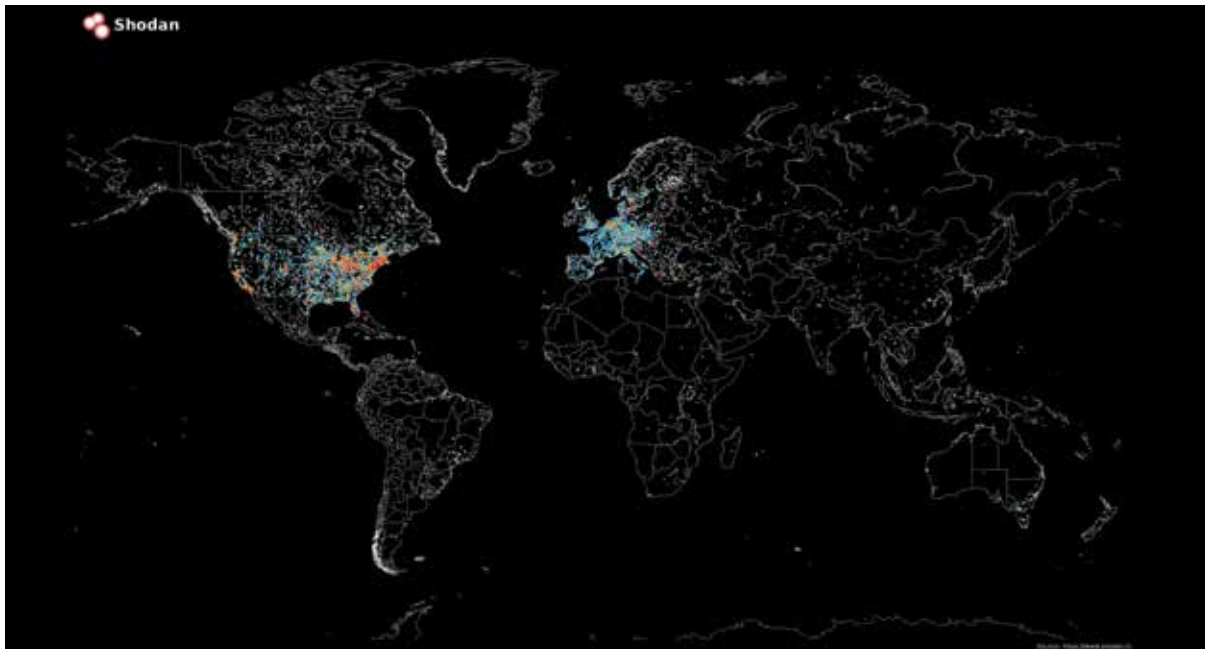


Figure 1 | Industrial Internet of Things devices found online by Shodan.

The Return of Dragons: How the Internet of Things is Creating New, Unexplored Territories

By John Matherly

The Internet of Things (IoT) is a recently popularized term for a trend that started more than a decade ago. Once the Internet took off, businesses quickly realized that networked devices could save them a lot of money.

The Internet would allow them to remotely monitor a factory, centralize data collection across offices, and make it easier to interact even on a local network. But companies had already spent hundreds of thousands of dollars, sometimes millions, on machinery they couldn't afford to replace. Hence the plan to create a serial-to-Ethernet adapter; a device wouldn't need to know anything about networks — it could operate as it always had and the adapter would bring it into the connected era. The migration happened slowly, but the benefits could be reaped immediately. At this point, scanning the Internet wasn't considered a feasible task and security by obscurity reigned supreme. These early connected devices — turbines, factories, substations, HVAC — form what is now dubbed the Industrial Internet of Things (IIoT).

Exposing IoT Insecurities

The response by vendors to the exposure of their devices on the Internet has been lukewarm. After a 2011 report showed that more than 10,000 control systems were accessible on the Internet, I was approached by a project manager of a large vendor to help track down the company's devices.¹ The goal was to identify customers who had misconfigured their products and gather market intelligence. I was excited that a company was finally interested in being proactive about the security of its customers, but my enthusiasm was short-lived. Early conversations with engineers surfaced a roadblock: No one believed that it would be possible to find any insecure devices of theirs on the Internet. I showed them the existing data that had been collected

for the report. I showed them how some of their devices had already been discovered, and it was simply a matter of knowing how to speak the language of their machines to find the rest of them. Unfortunately, I was unable to convince their technical staff that this would be a problem in the future and they should take it seriously. The engineers were experts in control systems, which misled them into thinking they were also experts on everything related to their product, including networks and security. A few months later, a major news story revealed that one of their products had a backdoor. They lost customers, had to notify affected companies, faced negative press, and had to beef up their security team as a result. They now have a computer emergency response team (CERT) for their products. This could have happened to any organization; many vendors, both for the industrial and consumer IoT, have poor security cultures. There is a fundamental lack of technical understanding of how the Internet works, what it means to be connected, and how things are turning into computers that rival our desktops in computing power. Since 2011, many vendors of industrial products have vastly improved their stances on security and they're moving in the right direction. Due to the nature of the industry, this is a slow process, but vendors are starting to become proactive.

The latest numbers indicate that there are at least 2 million devices on the Internet that make up the current IIoT (see Figure 1). They range from office buildings to traffic lights and even nuclear power plants. What unites all of them is the amount of information they divulge to anyone who knows how to communicate with the machine. Unlike the World Wide Web, the IoT is about machine-to-machine communication. These connected things weren't designed to share information with end users, they were made to communicate with other machines or engineers. This results in the devices freely sharing data about which version of software is running, how much power it generates, when it was last serviced, the make and model of the product, and sometimes even who installed the device. All of this is very useful when trying to debug a problem or ensuring your infrastructure has been properly maintained, but that same information is now also being made accessible to anyone over the Internet. For example, it is possible using Shodan to determine how much power is generated by Nordex wind turbines across the world.² How many turbines does Nordex's average customer have? Are they patching their systems? Answers to these questions are just a few mouse clicks away thanks to the IoT.

In the early 2000s, a series of worms and viruses popped up across the media seemingly every other week. "Code Red," "ILOVEYOU," and "SQL Slammer" were only a few of the iconic worms that made the news and caused havoc. The malware industry is still a booming business, but to the average consumer it has mostly fallen out of sight. The increase in computing power throughout the house combined with the sophisticated, modern techniques of malware create an environment that is ripe for exploitation. There have already been reports of refrigerators being compromised and used to send spam. How do you defend against this? When was the last time you patched your TV? Does your smart light bulb auto-update and apply security patches? Appliances and other household items are turning into full-fledged computers, yet they're still treated like the dumb devices they used to be. If your desktop gets infected, you hopefully have a backup and simply reinstall the system. But how do you reinstall your refrigerator? That mentality doesn't translate well to physical devices that lack a keyboard (or to vendors that prefer you'd rather upgrade to their latest iteration). For example, the majority of Android phone manufacturers consistently score poorly when it comes to providing operating system updates to their products.³

Raising Security Awareness

One of the unexpected benefits of the term "Internet of Things" is that it has become much easier to discuss security issues. It has also changed the way Shodan is framed. Shodan launched in 2009 as the world's first search engine for Internet-connected devices. For the first few years of Shodan, journalists who approached me with congratulations on the unique project declined to run stories because their readers wouldn't understand the technology. Today, I don't need to explain how the Internet works, why a refrigerator would be connected to the network, or how your coffee machine knows when you're home. It's all part of the vision that has been perpetuated by movies, TV shows, and the vendors selling the next generation of devices. The question isn't why something would be connected to a network, but rather, why isn't it secure by default? To that point, vendors are put in an awkward position where they need to move as quickly as possible to establish themselves as the leading platform. They also need to offer as many features as possible to replicate The Jetsons experience — all while trying to engineer devices in a world where consumers don't make purchasing decisions based on security features. Most people want to know how smart devices will




Smart devices are slowly changing the landscape of the Internet and in the process unleashing a new wave of insecure products.

make their lives easier, save money, and help protect their loved ones.

Security isn't the only thing that's been put on the chopping block. In April 2013 I received an email from a user named Mark in Chicago, who was browsing Shodan and uncovered a webcam feed that showed an elderly woman being assaulted live on camera. He had the foresight to record everything, and within a week the police had arrested the daughter of the elderly woman as the assailant. The daughter was paid by the state to take care of her mother and was required to set up a webcam so she could remotely monitor her mother's activities. Unbeknownst to her, the webcam was setup without authentication and Mark happened to catch the daughter attacking her mother. Stories like these are increasingly common and will continue to proliferate as various aspects of our lives become connected. We're already seeing more discussion around how privacy will be affected by smart cars, smart grids, and smart TVs. Much like security, though, I haven't observed any change in purchasing behavior based on privacy features in a product as measured by devices being online before and after a major news story about security or privacy breaks. Siemens recently announced a privacy policy for its smart TVs, and various app platforms are beginning to offer privacy protections and permission systems that are similar to smartphones.

The IoT Ahead

The world has become a magical place: With the push of a button you can turn off the sprinklers of your house halfway across the world. Your house can sense when you're home and turn on the heat, or make a fresh cup of coffee when you wake up in the morning. It is so obviously the way of the future that every major hardware manufacturer is trying to carve out its stake. The high level of competition, growth potential, and media coverage creates an environment that discourages security, maintainability, and testing. From an external look at the IoT, the industrial devices are still dominating the market. Home automation systems and smart TVs are slowly gaining traction as they become more affordable to the average consumer. Smart devices are slowly changing the landscape of the Internet and in the process unleashing a new wave of insecure products. And they provide instant, real-time, global access to usage information that was previously kept offline. Finally, we're still in the early stages of even detecting IoT devices. Since they often speak a custom, proprietary protocol, it is a painstaking and time-consuming process to learn the new machine languages. The amount of information that can be obtained will only increase as the IoT grows and the tools for interacting with it improve. 

John Matherly is an Internet cartographer, speaker, and founder of Shodan, the world's first search engine for Internet-connected devices. He previously worked as a freelance software engineer at a variety of companies, including bioinformatics work. Matherly has been featured on CNBC, CNN Money, Bloomberg, and in *The Washington Post* and *Forbes*.

REFERENCES

¹ <https://www.cl.cam.ac.uk/~fms27/papers/2011-Leverett-industrial.pdf>

² <https://www.shodan.io/search?query=jetty+2000+200+bytes>

³ <http://www.computerworld.com/article/2476223/android/android-upgrade-report-card--kitkat--six-months-later.html>

The Insecurity of Things

By Stephen A. Ridley



Every day we read about some newfangled Internet-connected device, such as fitness trackers, smoke alarms, televisions, cars, wall outlets — even Internet-connected water bottles.¹ Many of these devices are built with rushed-to-production software embedded in cheap micro-controllers.

Some believe that this Internet of Things (IoT) and embedded system explosion is driven by gadget hungry early adopters. For the consumer market, this may be true, but the real driver for this explosive growth is industry and the age old mantra, “better, faster, cheaper.” Ultimately, the proliferation of Internet-connected embedded devices is merely a side effect of the technological transition from a world built around ASICs (application-specific integrated circuits) to a world powered by cheap, general purpose SoCs (system(s)-on-a-chip).

The Explosive Growth of IoT Devices

The old ASIC design model is a costly process requiring multiple layers of engineering, with the final product being a very specific device (or chip) suitable only to the tasks for which it was designed. These chips require multiple iterations or complete redesigns should the device need to be upgraded or repurposed.

Today, this is less common due to the proliferation of multi-purpose, inexpensive SoCs. Unlike ASICs, SoCs contain everything that is needed to implement a complete general purpose computer. A single SoC chip (for less than \$10) contains a core microprocessor,

sound card, USB controller, Ethernet, Wi-Fi controller, Bluetooth, RAM, flash or disk storage, serial interfaces, infrared controllers, video controller, LCD controllers, and more.

Within the last five years, hobbyists and developers alike seem to have collectively realized that these SoCs are easily applicable to other areas. Hence the growth of low-cost full computer devices (like the Raspberry Pi) which merely consist of an SoC and the simple circuitry to connect the in-built functionality of these SoCs to common connectors for USB, power, and memory cards. Combined with the more recent uptick in active development and support of operating systems for ARM (the primary chip architecture for SoCs), the result is cheap SoC-based computers running full operating systems that are able to effectively eradicate many industry use cases for custom ASICs and custom circuitry.

The Shifting Threat Landscape

With the proliferation of these generic, multi-purpose devices, an average C programmer can make hardware do useful things with everything from robotics and drone navigation to everyday devices.² For example,

the hardware that implements the operational logic of an elevator control system no longer needs to be built by a team of electrical engineers and firmware authors, it can instead be implemented by a single software developer (with little to no knowledge of hardware design) by embedding software on an SoC. Conversely, the same system can also be implemented by an electrical engineer (with little knowledge of software development) writing his or her first simple program for an SoC.

For this reason, B2B companies like industrial control system (ICS) manufacturers are the predominant beneficiaries of the cheap development, deployment, and support afforded by SoCs. This also accounts for the glut of new Internet-connected appliances we see flooding the consumer market.

With all this comes a completely new frontier for security vulnerabilities. These vulnerabilities are further exacerbated by many of these fragile systems being actively networked or connected to the open Internet. The threat landscape is shifting from an emphasis on servers and endpoints (e.g., laptops and desktops) to everyday things like VoIP phones and cameras. Now there is new organizational risk of adopting these technologies.

As an example of this shift: Shodan, a search engine cataloging Internet-connected devices, found that several of the headquarter security camera networks and HVAC systems at Google were freely accessible on the Internet. Google takes information security very seriously, but the subcontractor that installed the cameras and HVAC systems did not. The recent Target Corporation compromise that resulted in one of the largest credit card thefts in history was also found to be attributable to the insecure systems installed by an HVAC sub-contractor.³

From stories of ATM and car hacks to webcam extortion schemes and exploited hotel locks, every day there is new evidence for this shifting threat model.⁴ And these don't just have to be localized, targeted attacks. In a recent publication by Brian Krebs it was found that "some of the biggest [denial-of-service (DoS)] attacks take advantage of Internet-based hardware — everything from gaming consoles to routers and modems."⁵ If the majority of Internet traffic generated during present-day DoS attacks originates from remotely compromised embedded systems and IoT devices, then it is no longer speculation that these systems are actively being exploited. These are the first tremors of a larger seismic shift in the threat landscape.



The Facedancer21 USB coupler for security research.

Can You Create Custom Hardware to Compromise Software?

Xipiter has conducted operating system security research with the Facedancer21, a device that tells a prognostic and cautionary tale of embedded security as a whole.⁶ The Facedancer21 is effectively a USB coupler that allows a researcher to interconnect two computers via USB. The Facedancer circuitry fools one of these computers (the target) into believing the other computer (the attacker) is actually a USB device, like a cellphone, thumb drive, keyboard, or Wi-Fi adapter. Hardware like this did not exist publicly before the Facedancer. From this capability alone, the Facedancer has allowed researchers to find vulnerabilities in fully patched current operating systems like Windows 8, Linux, and OSX.

Why was the Facedancer able to do this? These vulnerabilities stem from a fundamental design assumption inherent to modern operating systems. Modern operating systems inherently trust drivers. Drivers have mostly unfettered access to everything from within the heart of the operating system (the kernel). These USB drivers in turn inherently trust that the devices they interface with will only perform a set of pre-defined operations. Modern operating systems therefore inherently trust virtually every USB device to behave politely, and in doing so expose privileged memory and system resources to these devices.

This inherent trust is ultimately a weakness that makes it possible to subvert the security protections of every major operating system by exploiting weak drivers via USB. Xipiter (at the request of clients), has privately used the Facedancer to compromise everything from streaming set-top boxes to point-of-sale systems, all via externally accessible USB ports. Set-top boxes may not be a big deal, but the compromise of ATMs and point-of-sale systems via USB have obvious and monetary impact.

The Uncanny Valley of Software and Hardware

What has become immediately evident from Xipiter's consulting work and custom devices is a severe lack of embedded security expertise within information security. We believe that this ultimately stems from the very hardware/software divide from which these vulnerabilities are born: Hardware folks don't understand software and software folks don't understand hardware. We call this divide the uncanny valley.

And in this uncanny valley there is gold for attackers. The simplicity of the vulnerabilities found in embedded systems (and even the larger computer systems that rely on small embedded systems/chips internally) is predicated on this uncanny valley. The uncanny valley gives rise to two root causes of these vulnerabilities:

- 1) Lack of developer collaboration:** As with our previous elevator control system example, the developers of the software and hardware are not communicating or are poorly experienced. The software developers implementing the business logic have no understanding of the underlying hardware and its features or associated risks.
- 2) Reliance upon security through obscurity:**

Ultimately, many of these manufacturers create an appliance or device that has a clear value proposition to their customers. However, during development, security is never a consideration or the designers assume that security comes from the physical device being too obscure for anyone to target.

During the research for Xipiter's "Software Exploitation Via Hardware Exploitation" course, the co-developer discovered that virtually every current Apple computer is exploitable via the Thunderbolt port. Unlike similar attacks through FireWire ports, this attack did not require that the target computer have a special or specifically vulnerable driver installed.

Security researcher Joseph Fitzpatrick created a device he called the SLOTSREAMER that allows an attacker to simply plug his device into any Apple computer to unlock the lock screen or directly access anything currently in memory on the target system. How can this happen in 2015 on modern operating systems? We again see this as evidence of the uncanny valley. Designers of Thunderbolt at Apple incorporated direct memory access via this port through the PCI Express bus. Software designers at Apple were completely

oblivious to this low level of access and happily developed their secure software without considering this attack vector.

Impact of the Shifting Threat Landscape

Attackers today need to spend months or years of man hours finding vulnerabilities, weaponizing exploits, and building exploitation toolkits for operating systems cluttered by antivirus, software updates, intrusion detection, endpoint detection, etc. Why expend the resources, budget, and time when a determined adversary can find easy-to-exploit vulnerabilities in routers, access points, networked storage, mobile phones, smart home appliances, ICS systems, set-top boxes, or televisions? All of which are equally (if not more) operationally valuable to an attacker's mission.⁷

Furthermore, not only are the vulnerabilities very simple to exploit, but there is little to no conventional software protection on these devices. None of the devices Xipiter has investigated employ hardware protections (like self-destructing internals or tamper switches). These devices often have infrequent software updates or have update mechanisms that are trivially disabled. Should an attacker be detected on these systems with conventional methods (e.g., intrusion detection systems), very few of the industry leaders in forensics and incident response actively offer embedded systems analysis expertise.

Most of the public research to date on embedded device security is presented as corner cases or relevant to only specific devices (e.g., exploitation of an Xbox, or GoPro camera). This is unfortunate, because the majority of software security professionals miss the significance of publications like CPU cheat codes that effectively give internal access to hidden CPU functionality. In 2010, a relatively obscure hardware hacker that went by the handle "Czernobyl" discovered that all Advanced Micro Devices (AMD) processors ever produced had a built-in developer mode password that unlocked privileged debug functionality in AMD CPUs.⁸ This backdoor was presumably placed into processors by AMD developers in early stages of CPU design to facilitate their internal debugging. This code remained embedded through production. This is sadly very common in CPU fabrication and design; there are quite a few of these backdoors and vulnerabilities in other prominent desktop CPU products that have not made it to light of day.

In 2009 at the Chaos Communication Congress (26C3) in Berlin, researcher Felix Domke revealed that he had

discovered undocumented functionality in a peripheral device found commonly inside desktop PCs that granted that device unfettered direct memory access (via hardware) to system memory.⁹ This access would allow malicious firmware embedded in the device to completely subvert OS protection mechanisms and be used for persistence of malware outside of the purview of the operating system. This highlights possible attack vectors via undocumented functionality in hardware or malicious actors somewhere in a supply chain.¹⁰

The Way Forward

On the consumer side, there are a number of security issues. The primary issue is that there is no central body or organization performing advocacy for privacy and data handling. At CES 2015, FTC Chairwoman Edith Ramirez was the first to vocally call for more regulation of data brokers and services supporting the IoT.¹¹ Other than HIPAA and payment card industry (PCI) regulation, there are few standards bodies protecting consumers and their data with regard to IoT and embedded systems. Card payment systems may be somewhat covered by PCI policy, but what about biometric data in fitness trackers, mobile phone unique identifiers, or audio recordings of your voice? In early 2015 someone uncovered this sobering excerpt from a Samsung smart TV end user license agreement:

"Please be aware that if your spoken words include personal or other sensitive information, that information will be among the data captured and

transmitted to a third party through your use of [Samsung smart TV] Voice Recognition."

Other products that use similar voice-capture technology include Apple's Siri and Amazon's upcoming Echo. Increasingly these cloud-based voice recognition technologies are incorporated not just in mobile devices, but also navigation systems and automobiles. There are no regulatory bodies that impose restrictions or guidelines on how this data is used. Additionally, there is no existing service for consumers that functions like consumer reports or Equifax Credit Watch for the security of mobile applications and embedded systems. There exists no single resource for concise, actionable information about how adoption of a device or app will impact personal privacy.

On the industry side, the emerging threat landscape is clear. Organizations need to look more holistically at security policy. These policies should no longer include only endpoint protection, insider threat mitigation, access control, and the historical canon of infosec policy. Now the threats are coming from new places, and these policies need to be broadened to include a new range of technologies and devices. It is important to note that this is an emerging trend. We're still just before the curve on embedded security. There are sparse product and service offerings in this area now simply because of the uncanny valley. We also haven't yet experienced the watershed event that will cause the reactionary security industry to shift focus — but it appears imminent. **Q**

Stephen A. Ridley is a security researcher at Xipiter. He has more than 10 years of experience in software development, software security, and reverse engineering. Prior to Xipiter, Ridley served as the Chief Information Security Officer of a financial services firm and prior to that was a Senior Researcher at Matasano. He also was Senior Security Architect at McAfee, and a founding member of the Security and Mission Assurance (SMA) group at a major U.S. defense contractor where he did vulnerability research and reverse engineering in support of the U.S. Intelligence Community. He has spoken about reverse engineering and software security at Black Hat, ReCon, CanSecWest, EUSECWest, Syscan, and other prominent information security conferences. Ridley is a co-author of "The Android Hacker's Handbook," published by Wiley & Sons.

REFERENCES

- Phillips, Jon. *TechHive*. July 24, 2014. <http://www.techhive.com/article/2457644/why-have-consumers-spent-1-million-on-vessyl-an-absurd-calorie-counting-cup.html> (accessed February 2015).
- "Why Writing Firmware Is Kinda Like Software Exploitation." March 22, 2014. <http://dontstuffbeansupyournose.com/2014/03/22/why-writing-firmware-is-kind-like-software-exploitation/> (accessed February 2015).
- Krebs, Brian. *Target Hackers Broke in Via HVAC Company*. February 5, 2014. <http://krebsonsecurity.com/2014/02/target-hackers-broke-in-via-hvac-company/> (accessed February 2015).
- Greenberg, Andy. *Forbes*. July 23, 2012. <http://www.forbes.com/sites/andygreenberg/2012/07/23/hacker-will-expose-potential-security-flaw-in-more-than-four-million-hotel-room-keycard-locks/> (accessed February 2015).
- Krebs, Brian. "The Internet Of Dangerous Things." January 29, 2015. <http://krebsonsecurity.com/2015/01/the-internet-of-dangerous-things/> (accessed February 2015).
- INT3. *INT3.CC*. January 2013. <http://int3.cc/products/facedancer21> (accessed February 2015).
- "The Insecurity Of Things." December 2014. <http://www.xipiter.com/musings/using-the-shikra-to-attack-embedded-systems-getting-started> (accessed February 2015).
- Goodin, Dan. *The Register*. November 15, 2010. http://www.theregister.co.uk/2010/11/15/amd_secret_debugger/ (accessed February 2015).
- Domke, Felix. "Blackbox JTAG Reverse Engineering." *26th Chaos Communication Congress*. Berlin: CCC, 2009.
- Rosenfeld, Karri. "Attacks and Defenses for JTAG." *Polytechnic Institute of New York University*, 2010: 2-13.
- FTC. *Internet of Things: Privacy & Security in a Connected World*. FTC Staff Report, FTC.gov, 2015.



Memory Forensics for Malware Detection and Analysis

By Vico Marziale

To start with something of an understatement, dealing with modern malware is a complex undertaking.

On the detection side, malware authors can hide deep in the kernel of the operating system they've infiltrated, rendering most userland tools ineffective at detecting them. They use obscure methods, leveraging the newest additions to the newest versions of operating systems and methods from years past that still work, to load their code on a machine and run in a way that is hidden from all but the deepest of investigations.

Alternately, on the analysis side, the inherent code complexity of modern malware that must interact with ever more heterogeneous systems, like cloud services and mobile devices, poses a problem. Malware authors also go to tremendous lengths to complicate the analysis process — especially the nation-state actors that have recently made headlines with Stuxnet, Duqu, Flame, and Regin. They are careful to obscure the contents of malicious binary executable files themselves via encryption or compression. The code itself is intentionally obfuscated with convoluted pathways, unreachable code, and obscure API usage. Some malware is instrumented to detect when it is being run

in a debugger and stops an analyst from witnessing its inner workings, such as lying dormant for some period of time, or attempting to crash the machine. The same goes for malware being executed in virtual environments, such as VMware and malware analysis sandboxes like Cuckoo Sandbox. Some of the more interesting malware samples reside in memory only, leaving absolutely no traces of their existence on disk.

To fight through these obfuscations, there are several well-understood malware analysis techniques that vary in complexity and in the amount of information that can be discovered. Generally, a combination of these techniques is leveraged in any given analysis.

Static Analysis

The first type of malware analysis we'll address, static analysis, entails analyzing the binary file of the malware. One of the simplest methods to analyze a piece of malware is to run low-level tools including hex editors (e.g., xxd), text extraction tools to find readable strings in files (e.g., Unix strings utility), and string search tools (e.g., Unix grep utility) on the executable. This can reveal information including IP addresses, URL strings, ports, file names, etc., which exist as strings in the executable and provide insight into the operation of the specimen. While this technique is trivial to execute, it provides extremely limited information, and is often easily thwarted by encrypting the binary and other obfuscation techniques. More advanced static analysis entails full program reverse engineering by disassembling the malware executable. This, at its most basic, makes use of a static disassembler tool (e.g., IDA) to break an executable into a set of machine instructions. These instructions are then painstakingly analyzed to determine the exact behavior of the specimen. This type of analysis can provide a more complete picture of the function of a specimen, but often requires a tremendous time investment from analysts with significant skills, and there is still the encryption problem.

Behavioral Analysis

A second type of malware analysis, behavioral analysis, involves gathering information on the behavior of a piece of malware by executing it and then recording and analyzing system behavior while it runs. This sidesteps the encrypted binary problem, as the executable must decrypt itself in order to run on the machine. Sysinternals is an example of a tool that can record activity such as new processes starting, file accesses, and registry updates. Similarly, program debuggers (e.g., OllyDbg), and malware execution sandboxes (e.g., Cuckoo Sandbox) increase an analyst's ability to view what a particular malware sample is attempting to accomplish. Examining this data can shed significantly more light on the operation of a specimen, providing far more information than strings, but it requires more resources, time to sort out the big picture of what a specimen is doing, and greater technical expertise on the part of the analyst. This is not a perfect solution, as it is common for malware to attempt to hide itself from these types of monitoring applications. If a malware sample detects that it is being monitored while running, it can delay any malicious behavior, choose to exit, or exhibit only benign behavior for the rest of its execution.

Memory Forensics

Yet another method is memory forensics, a static-behavioral hybrid that offers an attractive tradeoff in time and expertise requirements versus the depth of information that can be gleaned from a malware sample. Further, memory forensics can be used to detect malware on infected systems and analysis of known malware samples. As in behavioral analysis, it relies on the fact that in order to execute, malware must be unencrypted and generally de-obfuscated in memory. But as in static analysis, it relies on analyzing a static file so that the malware cannot actively manipulate the analysis.

When using memory forensics for malware analysis, typically a clean machine (or virtual machine) is infected with a malware specimen. Then a byte-for-byte copy of the contents of physical RAM on the machine is taken and saved in a file (a memory image), using tools like KnTDD. This static image is then used as the basis for analysis using a number of tools and techniques similar to those listed above. When a machine on your network is suspected of an infection, the same type of memory image can be taken and detection/analysis can proceed using similar tools and techniques.

Just like on the binary, hex viewers and string extractors can be used against the memory image — only at this point, some or all of the malware should be unencrypted, and therefore should likely provide more information than possible on the encrypted binary itself. The Volatility Framework and other memory forensics tools can also find and export individual processes from memory images. These exported processes can be scanned for strings, but, more importantly, can be loaded into a disassembler for further analysis. Again, this representation of the process is highly likely to have many of the obfuscations present in the on-disk binary, like encryption, removed, as they must be in order for the process to have executed properly.

Additionally, tools like Volatility can parse a wealth of information from a memory image. Just as one might see when performing behavioral analysis, Volatility can mimic the output of some Sysinternals and other command line tools that provide information about what was occurring on the machine when the memory image was taken, like listing running processes, open files, loaded dynamic link libraries (DLLs), and open network connections. Further, it can dig deeper into the workings of the kernel, detecting and parsing out hooked system calls, memory-resident registry hives and the master file table (MFT), loaded drivers, and much more.

```
python damm.py -p processes --db after.db --diff before.db
```

```
processes
Status offset name pid ppid prio image_path_name create_time exit_time threads session_id handles isWow64 pslist psscan thrdproc pspcid
csrss session desktop 1256 1940 8 New 0x17d22e0 python.exe 2013-10-31 23:23:14 UTC+0000 2013-10-31 23:23:19 UTC+0000 0 -268370093 False False True False
False False False False False Changed 0x18b4d38 svchost.exe 1080 692 8 2013-10-31 17:21:26 UTC+0000 66->71 False False True False False
False False False False False Changed 0x1915198 winlogon.exe 648 376 13 2013-10-31 17:21:25 UTC+0000 24->26 False False True False False
False False False False False Changed 0x1900120 services.exe 692 648 9 2013-10-31 17:21:25 UTC+0000 16->18 False False True False False
False False False False False Changed 0x1800360 svchost.exe 1124 692 8 2013-10-31 17:21:26 UTC+0000 5->6 False False True False False
False False False False False Changed 0x1875490 explorer.exe 1636 1596 8 2013-10-31 17:21:27 UTC+0000 13->14 False False True False False
False False False
```

Figure 1 | DAMM's Processes plugin immediately tells the analyst that one new process has been created and that several other processes have changed.

It is important to emphasize that since the investigative basis is a memory snapshot, we sidestep two significant difficulties in malware analysis. First, since the malware itself is no longer currently running, it cannot act to hide itself in the face of the analyst's probing. Second, since execution of malware sample requires the binary to be unpacked, the version of the specimen in memory will often be largely de-obfuscated.

This type of analysis is not perfect either, however. Current analysis methods for memory images are limited in that they are designed to analyze a single memory image at a time. Recall that when attempting to analyze malware, a common technique is to spin up a clean virtual machine, infect it with said malware, and then acquire a memory image. This infected image is then the starting point for analysis. What should be apparent is that large portions of such a memory capture can effectively be ignored - namely any part that the malware did not affect. With a single image, the only way to determine the changes made by the malware is to have a deep understanding of what should be in the image, as in default Windows processes and other artifacts specific to your infrastructure, and then look at everything else. This can be a tedious, time consuming, error prone process, requiring the effort of highly skilled analysts. To ease this process, BlackBag Technologies has been working on a new method to analyze an unknown piece of malware: differential analysis of memory images.

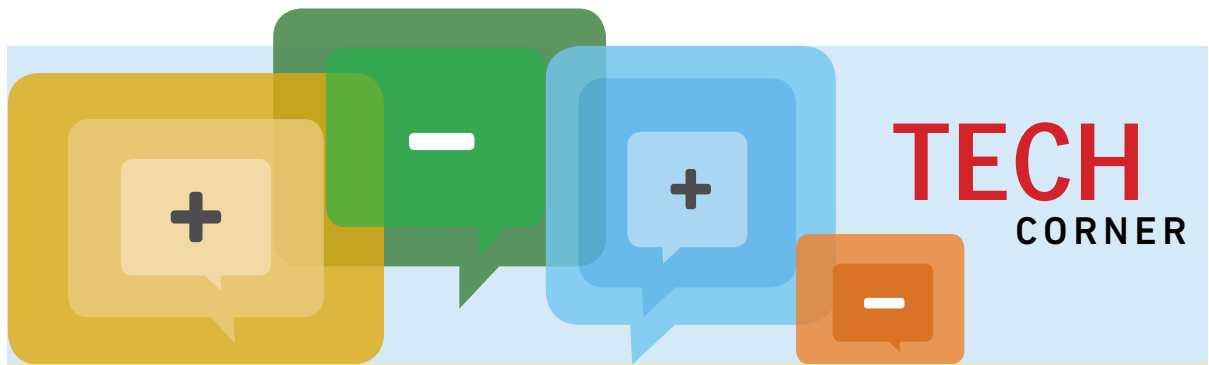
The result of this research is a free, open source, proof-of-concept tool, called Differential Analysis of Malware in Memory (DAMM), for determining the actions of a piece of malware by leveraging memory images taken

before and after the execution of the malware. While certainly not all of the changes will be the result of the executing malware, we can be sure that many of the changes the malware is responsible for are there. DAMM provides a plugin-based system that can be used to build modules with specific knowledge of any type of in-memory artifacts, like running processes, drivers, or loaded DLLs. Plugins have already been created to analyze changes in a number of artifacts, including processes, open network sockets and files, hooked functions, and changes in other in-memory artifacts including the MFT. These plugins have been designed specifically to display the differences in successive memory images in a manner immediately useful to investigators.

As seen in Figure 1, using DAMM's Processes plugin immediately tells the analyst that one new process (denoted by "New" in the first column) has been created and that several other processes have changed (actual changes denoted by "->"). This and other plugins give the analyst a wealth of information on which actions the malware took, such as hooking system calls, opening files for writing, opening network connections, creating or deleting files, or setting registry keys for persistence. Automatically generating this information significantly reduces the amount of time required to figure out which actions the malware performed without necessarily having to do deeper analysis.

Malware detection and analysis is a complicated undertaking. Leveraging the newest tools and techniques, like memory forensics, is absolutely essential to fight the rising tide of malware all around us. [Q](#)

Dr. Vico Marziale, Senior Research Developer at BlackBag Technologies, has a Ph.D. in Computer Science from the University of New Orleans. He currently focuses on R&D in the digital forensics world and has previous experience in penetration testing, digital forensic investigation, malware analysis, and incident response. Marziale is co-developer of the Scalpel file carver, Registry Decoder for Windows registry forensics, and Spotlight Inspector for analysis of OSX Spotlight metadata indexes. He has published, presented, and demonstrated at events including Digital Forensics Research Conference (DFRWS), Open Source Digital Forensics Conference (OSDFCon), Open Memory Forensics Workshop (OMFW), Black Hat, multiple Security BSides, DOD Cybercrime, and RSA. Marziale is also one of the organizers of Security BSides NOLA and a member of the American Academy of Forensic Sciences.



To supplement the *IQT Quarterly*'s focus on technology trends, *Tech Corner* provides a practitioner's point of view of a current challenge in the field and insight into an effective response.

Reviving the Lost Art of Cyber Defense: Malware Detection Using Fast and Scalable Functional File Correlation

In recent news, the failure of detection at the perimeter has made many Fortune 1000 CISOs skeptical of any value in defending their networks and endpoints. Perimeter has become for them mere police tape — not crime scene protection, but rather a psychological deterrent to would-be trespassers.

Such emotional reactions are indicative of a deep-seated despair facing today's responders and the mismatched state of available technologies, legislature, and internal capabilities tasked with defense. It has never been so important to consider ways to improve our defense posture by taking an honest look at detection successes, failures, and potential game-changing strategies for the future. As cybersecurity has been compared to game theory on many occasions, our response can never be refusing to play.

Over the last decade, cyber threat detection solutions have become increasingly ineffective, evidenced by the market's sentiment that "AV (antivirus) is dead."

In response, AV companies began to heavily invest in augmenting their technologies, adding sometimes even as many as 10-13 agents, URL filtering, and domain and IP address intelligence. A new product category was created, the Anti-Malware Suite, increasing the dismal 30 percent detection rate to 95 percent (according to independent testers). This still meant that more than 50,000 new and unique threats were missed every day.

Today, these companies are swamped with growth in samples while their legacy signature building methods and proactive protection (read: behavioral and network

indicator) methods fell short of expectations. The underlying product development strategy of prioritizing threats that target the largest number of targets does not work for the motivated attackers of today that have been behind the most sophisticated APT and financial fraud cases, and whose goals are decidedly not innocent or attention-getting.

Tremendous growth in Internet traffic, connected devices, and software has been married with even larger growth in malicious code or malware. Most of the growth amounts to adware, potentially unwanted programs, and virulently polymorphic banking Trojan strains such as Zeus. The situation has become significantly worse as platforms that enable less sophisticated hackers move to open source or are modified by different actors. This has resulted in a massive growth of threats captured and reported at a rate that exceeds Moore's Law, reaching more than 1 million unique malware samples per day.

This unprecedented growth in malware has challenged traditional detection methods, necessitating new classification and detection ideas. In a traditional lab, humans must inspect an exponentially increasing number of suspected malware files. Better automation and proactive technologies are essential to keep up with this malware escalation.

While the volumes have increased greatly, it is the flexibility of the Windows platform that still attracts the vast majority of threats. Android comes in second and offers more infection vectors than iOS and MacOS combined.

Historic malware detection methods utilize simple file hashing, pattern-based signatures, and behavior-based matching. The goal was not to attempt to look into forward-looking code similarity algorithms, but rather to model the past. A brute force, manual analytic process with more than 1,000 analysts (some large AV labs treat this process as a production line, dedicating 30 seconds to each signature) could never deliver signatures to cover today's malware arrival rate. Hence, new methods for better inspecting files at the endpoint and in the lab are required. YARA rules, a pseudo-rich signature building language, and whitelisting have gained many adherents, even if effective only on a small subset of endpoints.

Some of the more interesting new approaches to proactive malware discovery and classification include:

- 1) ImpHash and PEhash examine file similarity of certain high-level portable executable (PE) format characteristics. While yielding notable results for Mandiant and Google, this method is blind to .NET, packed and protected malware, and content hiding behind installers.
- 2) Machine learning, post-processed control flow, and symbolic analysis incur lengthy and costly infrastructure while attempting to address malware evasion and virtualization techniques through raw computing power (e.g., DARPA's Cyber Genome project and Zynamics' VxClass project).
- 3) Static or dynamic behavioral sequencing using Ngrams have been popularized by companies such as HBGary, Cylance, or projects like CodeDNA. While these methods can be compelling, they also suffer from a high rate of false positives (behaviors look alike) and false negatives (evasion, packing, protection) that are easily exploited by adversaries. One notorious problem with aggregate scores and sequence patterns is that everything becomes relative at some point, as users need to set and reset initial parameters.
- 4) Mathematical approaches leveraging Bayesian algorithms and neural networks, while successful on limited handpicked data sets, use mathematics to model what has been handpicked by the adversary as its evasion technique or strategy. The preponderance of protected and virtualized binaries escape easy mathematical interpretation.
- 5) Fast and scalable functional file correlation that is performed after de-obfuscation and removal of non-salient file format elements (e.g., ReversingLabs Hashing Algorithm, or RHA).

ReversingLabs Hashing Algorithm: Predictive Malware Detection

ReversingLabs has recently implemented a new approach to automatic code similarity matching. More than 2 million unique PE malware files have been processed and with the absolute reduction better than 92 percent of unique identifiers (and as high as 99 percent for certain malware families). This algorithm is an excellent automated tool for classification and description of known threats based on underlying code similarity while ignoring typical polymorphic, obfuscation, malformation, and packing characteristics.

RHA can also automatically classify unknown and whitelisted content. Code similarity indicators are generated in milliseconds and are indicative of the code similarity among unknown files. These similarities can then be analyzed for anomalies, dynamic behaviors, and symbolic execution. In this way, RHA becomes a magnet for new and unusual file clusters that defy standard whitelisting and blacklisting identification methods. They generate new alerts by correlating seemingly unrelated binary evidence.

RHA demonstrated impressive results in the spring of 2014, when Arbor Networks embarked on a large malware identification study. The results shown in

Family	Total Samples	RL Unique Identifier Hashes	Percentage Reduction
Athena	95	68	28.42%
Beta Bot	224	133	40.63%
Blackshades	117	79	32.48%
Citadel	3,533	1,658	53.07%
DarkComet	1,079	464	57.00%
DaRK DDoSeR	137	39	71.53%
Dirt Jumper	184	105	42.93%
Expiro	2,956	168	94.32%
Gamarue	2,342	153	93.47%
Ghostheart	141	87	38.30%
MachBot	627	5	99.20%
Nitol	129	71	44.96%
Pushdo	193	74	61.66%
Shylock	847	319	62.34%
Simda	1,396	303	78.30%
YoyoDDos	54	31	42.59%
TOTAL	14,054	3,757	73.27%

Figure 1 | Automatically generated malware classification signatures.

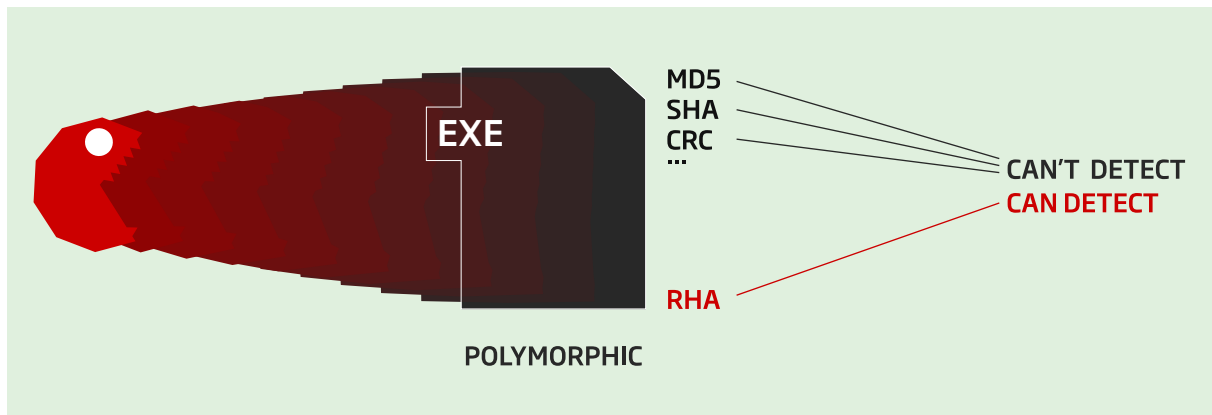


Figure 1 indicates the great potential for matching advanced malware, and particularly polymorphic file infectors, especially when considering the effects of traditional evasion, packing, and obfuscation methods. Overall signature coverage effectiveness was almost 75 percent in the first implementation of the algorithm.

RHA has a strong affinity for the underlying execution format and hence requires separate implementations for PE, .NET, ELF, .dex, PDF, Flash, and more. RHA applies to a broad set of executable formats including Windows, Linux, MacOS, mobile, and all the way to firmware and embedded formats. This provides a universal and flexible methodology that can easily move from one threat vector to another.

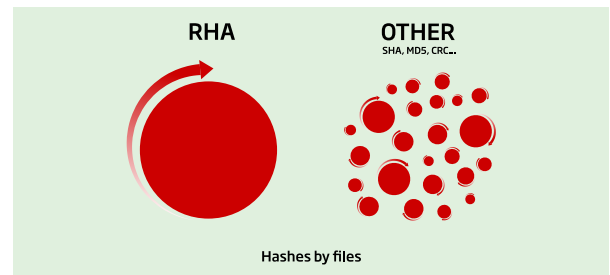
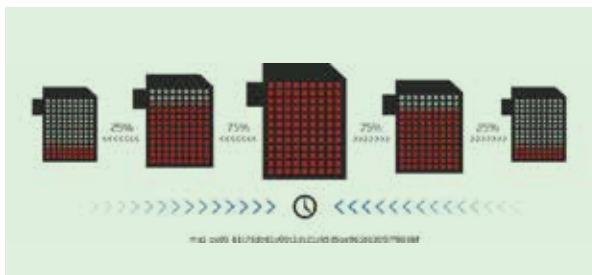
Traditional hashing algorithms (e.g., MD5, SHA-1) provide an important tool for security applications. Although commonly used for whitelisting and blacklisting, traditional hashes have significant drawbacks for detecting malware. First, a malicious file must be seen before a hash can be created so polymorphic attacks are not detectable. Second, hashes are fragile, enabling malware authors to make inconsequential changes to files to avoid detection.

RHA addresses these issues by intelligently hashing a file's features rather than its bits. Files have the same RHA hash when they are functionally similar. This makes RHA orders of magnitude better than traditional hashes for malware detection. One RHA hash can potentially identify thousands of functionally similar malware files even though each has a unique SHA-1 hash. Further, RHA will detect malware that doesn't yet exist because it is functionally similar to known malware.

How RHA Works

RHA enables correlation of files based on functional features. These attributes include format-specific header information, file layout, and functional file information (e.g., code and data relationships). RHA calculates functional similarity at four "Precision Levels" — 25, 50, 75, and 100 percent — each based on an increasing number of attributes. Precision Level represents the degree that a file is functionally similar to another file. A higher Precision Level will match fewer files, but the files will have more functional similarity.

RHA can be applied to any executable file format. First, format-specific features are abstracted into categories such as structure, layout, content, symbols, functionality, and relationships. Then, algorithms are



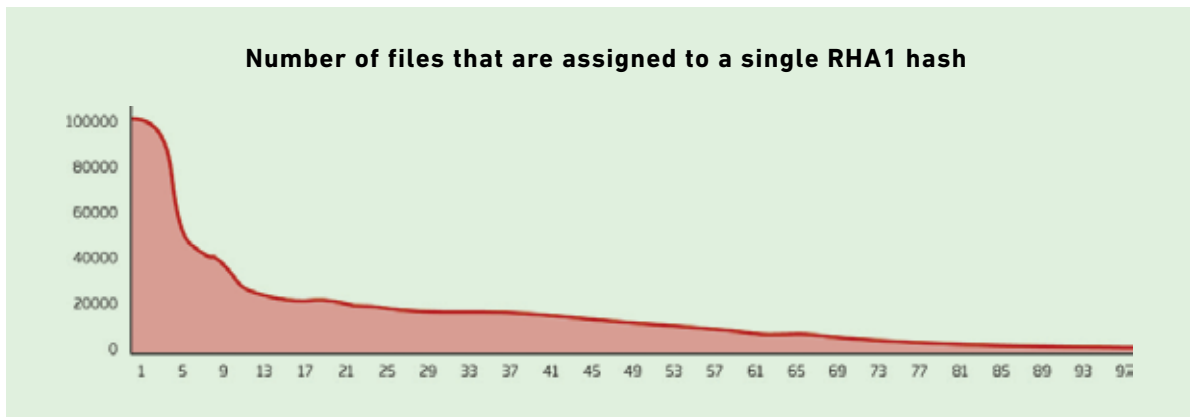


Figure 2 | Unique binaries that map to a single RHA1 hash at the lowest Precision Level.

implemented to evaluate the attributes of each category for similarity at each precision level. Algorithms will vary for each format but usually entail data sorting and simplification. The algorithms calculate a hash for each Precision Level so that functionally related files fall into the same hash group.

Each Precision Level's hash is deterministic and tied to functional configuration. This makes Precision Levels distinct with no overlaps in hash lookup. This hash determinism ensures the fastest possible hash lookup times.

Validation

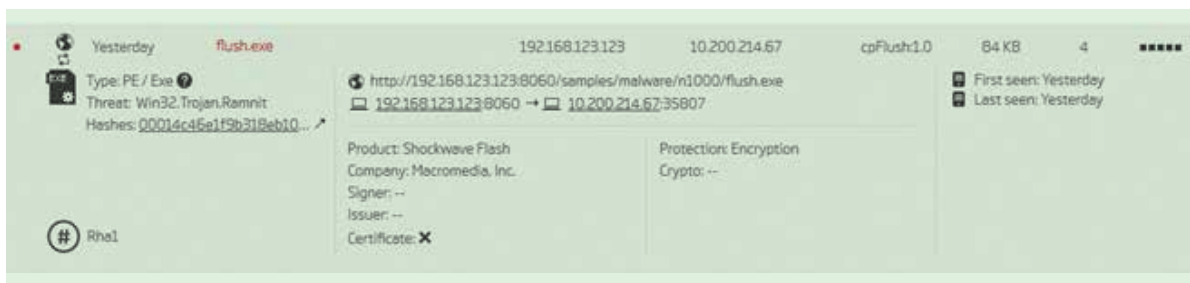
The effectiveness of RHA was tested using 7.75 million unique malware samples that were detected as part of the Zeus malware family by at least one antivirus vendor. The samples were processed with the algorithm at the lowest precision level resulting in 475,000 unique RHA1 hashes. This effectively reduced the working malware set size by 93 percent.

We expected a reduction in sample uniqueness for members of the same malware family, but didn't expect the magnitude of reduction. We analyzed the sample

data to better understand why the effectiveness was so high. Hence, we started with the hashes that yielded the most matches. Figure 2 shows the number of unique binaries that map to a single RHA1 hash at the lowest Precision Level.

The top matching RHA file sample showed that our best match wasn't on a particular malware family, but on a packing wrapper used to mask the true attack. This was not a common off-the-shelf packer, such as Ultimate Packer for Executables (UPX), but a custom packing solution developed exclusively to hide malware presence.

Since packing can obscure detections and their malware family groupings, we turned to antivirus solutions to see how they classified the top match. Figure 3 shows the normalized threat names for the 100,000 files of the most prevalent RHA hash. There wasn't a consensus on the threat name and only one antivirus vendor classified these samples as Zeus. Since it's clear that the packing layer interferes with proper detections, we've upgraded our TitaniumCore solution to support this custom packing solution we call cpFlush.



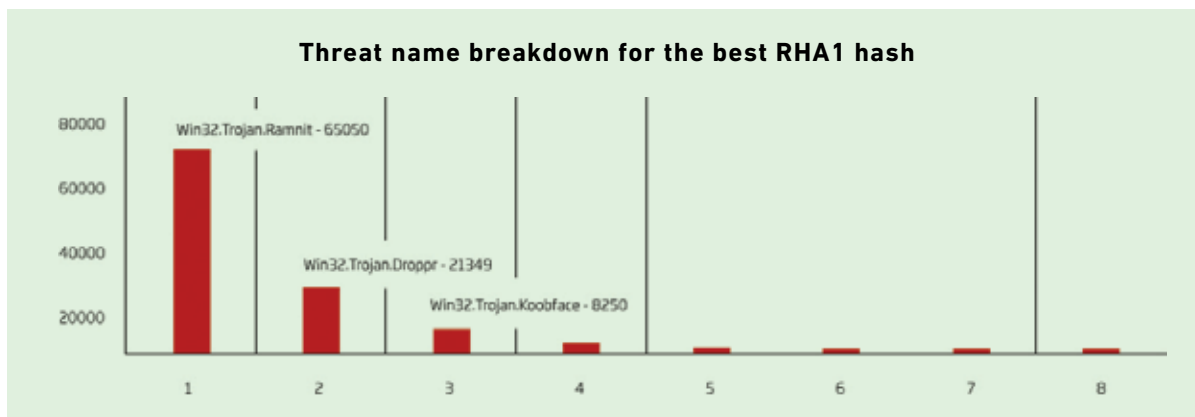


Figure 3 | Normalized threat names for the 100,000 files of the most prevalent RHA hash.

Unpacking the files showed that the top match was also using multiple packing layers. The number of corrupted and incorrectly packed files was low, so we could successfully unpack 95 percent of the samples. Comparing the RHA of files at each layer of packing showed that they remained within the same functional hash buckets. This indicates that the differences between these files were indeed minor.

RHA, even at the lowest precision level, showed no collisions with whitelisted files and therefore was safely applied to our automatic RHA cloud classification.

The custom packer was blacklisted using its format signature. RHA enables us to detect multiple malware families that use it.

RHA provides a new security tool for effectively detecting present and future malware. The power of this tool is multiplied when used with an extensive file reputation database like ReversingLabs' TitaniumCloud. This combination enables large-scale detection of new malware variants through function similarity to known malware. [Q](#)

ReversingLabs is an IQT portfolio company that delivers industry-leading threat detection and analysis solutions to address the latest generation of cyber attacks. To learn more, visit www.reversinglabs.com.



FROM THE PORTFOLIO



Big Switch Networks

Big Switch Networks is a bare metal SDN (software-defined networking) company whose SDN fabric solutions embrace industry standards, open APIs, open source, and vendor-neutral support for both physical and virtual networking infrastructure. The company was recently named one of CRN's Top 20 Virtualization Vendors. In February, Big Switch announced the availability of BSN Labs, an online portal that allows users to trial its SDN products. The company has been an IQT portfolio company since November 2013 and is located in Palo Alto, CA.



HyTrust

HyTrust is the cloud security automation (CSA) company, providing extra-strength security for cloud infrastructure. The company was recently named one of CRN's 20 Coolest Cloud Security Vendors. In January, HyTrust co-founder Eric Chiu was interviewed by NBC about cybersecurity in light of ISIS attacks, and by *IT Security Guru* for his views on cybersecurity legislation. HyTrust is based in Mountain View, CA and has been a part of the IQT portfolio since June 2013.



RedSeal Networks

RedSeal Networks is a leading developer of security assurance software for medium to large-sized organizations. RedSeal CEO Ray Rothrock was quoted by *The Wall Street Journal* following the White House Summit on Cybersecurity and Consumer Protection, and by NBC Chicago for his 2015 cyber predictions. RedSeal Networks is based in San Mateo, CA and joined the IQT portfolio in December 2010.



Tenable Network Security

Tenable Network Security makes IT risk monitoring solutions that protect organizations from threats, vulnerabilities, and compliance violations. Tenable CEO Ron Gula was recently cited in *Security Week* with reactions to the executive order on cybersecurity information sharing. Gula applauded the White House's attention to cybersecurity legislation and awareness in the face of rapidly evolving technology. Tenable became an IQT portfolio company in July 2012 and is located in Columbia, MD.

