The
Internet Protocol Journal

August 2018 Volume 21, Number 2

A Quarterly Technical Publication for Internet and Intranet Professionals

In This Issue

You can download IPJ back issues and find subscription information at:
www.protocoljournal.org

ISSN 1944-1134

From The Editor

In June 1998 we published the first issue of *The Internet Protocol Journal* (IPJ). Since then, we have produced 75 issues and a total of 2,848 pages. Today, IPJ has about 22,000 subscribers all around the world. Although two-thirds of our readers prefer the paper edition, a growing number of subscribers are downloading the PDF version instead. As we remarked in 2008, this shift in reading habits is related to the emergence of low-cost, high-resolution displays and printers, as well as improvements in Internet access technologies, particularly with respect to mobile devices.

In this 20th anniversary year, we decided to ask our most frequent contributor, Geoff Huston, to reflect on Internet developments since 2008. His article, "Another 10 Years," outlines the many ways in which the Internet has changed in this period, as well as a few areas where developments are still lacking.

The Internet continues to be used for numerous unsavory activities including fraud, identity theft, malicious software intrusion, denial-of-service incursions, and much more. These cyber attacks are getting increasingly sophisticated, as David Strom explains in his article entitled "Fileless Malware."

As I did 10 years ago, let me take this opportunity to thank all those people who make IPJ possible. Our authors deserve a round of applause for carefully explaining both established and emerging technologies. They are assisted by an equally insightful set of reviewers and advisors who provide feedback and suggestions on every aspect of our publications process. The process itself relies heavily on two individuals: Bonnie Hupton, our copy editor, and Diane Andrada, our designer. Of equal importance are our numerous individual donors and corporate sponsors, without whom we would be unable to publish and distribute the journal. Last, but not least, our readers give us encouragement, suggestions, and feedback that enables us to provide the most relevant material.

If you are wondering why this is issue has a cover date of "August" rather than "June," I can only apologize and blame it on a busy summer and a broken arm. I appreciate your continued patience and support.

—Ole J. Jacobsen, Editor and Publisher
`ole@protocoljournal.org`

Another 10 Years

by Geoff Huston, APNIC

The evolutionary path of any technology can often take strange and unanticipated turns and twists. At some points simplicity and minimalism can be replaced by complexity and ornamentation, while at other times a dramatic cut-through exposes the core concepts of the technology and removes layers of superfluous additions. The evolution of the Internet appears to be no exception, and it contains these same forms of unanticipated turns and twists. In thinking about the technology of the Internet over the last 10 years, it appears that it's been a very mixed story about what has changed and what has stayed the same.

A lot of the Internet today looks much the same as the Internet of a decade ago[0,1]. Much of the infrastructure of the Internet has stubbornly resisted various efforts to engender change. We are still in the middle of the process to transition the Internet to IPv6, as was the case a decade ago. We are still trying to improve the resilience of the Internet to various attack vectors, as also was true a decade ago. We are still grappling with various efforts to provide defined *Quality of Service* (QoS)[2] in the network, also true a decade ago. It seems that the rapid pace of technical change in the 1990s and early 2000s has simply run out of momentum, and that the dominant activity on the Internet over the past decade was consolidation rather than continued technical evolution. Perhaps this increased resistance to change is because as the size of the network increases, its inertial mass also increases. We used to quote *Metcalf's Law*[3] to each other, reciting the mantra that the value of a network increases in proportion to the square of the number of users.

A related observation appears to be that the inherent resistance of a network to change, or its inertial mass, is also directly related to the square of the number of users. Perhaps as a general observation, all large, loosely coupled, distributed systems are strongly resistant to efforts to orchestrate a coordinated change. At best, these systems respond to various forms of market pressures, but because the overall system of the Internet is so large and so diverse, these market pressures manifest themselves in different ways in different parts of this network. Individual actors operate under no centrally orchestrated set of instructions or constraints. Where change occurs, it is because some sufficiently large body of individual actors sees opportunity in undertaking the change or perceives unacceptable risk in not changing. The result for the Internet appears to be that some changes are very challenging, while others look like natural and inevitable progressive steps.

But the other side of the story is one that is about as diametrically opposed as it's possible to paint. Over the last decade, we've seen another profound revolution in the Internet as it embraced a combination of wireless-based infrastructure and a rich set of services at a speed that has been unprecedented. We've seen a revolution in content and content provision that has changed the Internet, and as collateral damage the Internet appears to be decimating the traditional newspaper and broadcast television sectors. Social media has all but replaced the social role of the telephone and the practice of letter writing. We've seen the rise of the resurgence of a novel twist to the old central mainframe service in the guise of the *cloud*[4,5] and the repurposing of Internet devices to support views of a common cloud-hosted content that in many ways mimic the function of display terminals of a bygone past. All of these developments are fundamental changes to the Internet and all of them have occurred in the last decade!

That's a significant breadth of material to cover, so I'll keep the story to the larger themes, and to structure this story, rather than offer a set of unordered observations about the various changes and developments over the past decade, I'll use a standard model of a protocol stack as the guiding template. I'll start with the underlying transmission media and then look at IP, the transport layer, and applications and services, and then close by looking at the business of the Internet to highlight developments of the last decade.

Below the IP Layer

What's changed in network media?

Optical systems have undergone sustained change in the past decade. A little over a decade ago production optical systems used simple on-off keying to encode the signal into the optical channel. The speed increases in this generation of optical systems relied on improvements in the silicon control systems and the laser driver chips. The introduction of *Wavelength-Division Multiplexing* (WDM) in the late 1990s allowed the carriers to greatly increase the carrying capacity of their optical cable infrastructure. The last decade has seen the evolution of optical systems into areas of polarisation and phase modulation to effectively lift the number of bits of signal per baud. These days 100-Gbps optical channels are commonly supportable, and we are looking at further refinements in signal detection to lift that beyond 200 Gbps. We anticipate 400-Gbps systems in the near future, using various combinations of a faster basic baud rate and higher levels of phase amplitude modulation, and dare to think that 1 Tbps is now a distinct near-term optical service.

Radio systems have seen a similar evolution in overall capacity. Basic improvements in signal processing, analogous to the changes in optical systems, have allowed the use of phase modulation to lift the data rate of the radio bearer.

The use of *Multiple-Input* and *Multiple-Output* (MIMO) technology, coupled with the use of higher carrier frequencies, has allowed the mobile data service to support carriage services of up to 100 Mbps in today's *Fourth-Generation* (4G) networks. The push to even higher frequencies promises speeds of up to 1 Gbps for mobile systems in the near future with the deployment of 5G technology.

While optical speeds are increasing, Ethernet packet framing still persists in transmission systems long after the original rationale for the packet format died along with that bright-yellow coaxial cable! Oddly enough, the Ethernet-defined minimum and maximum packet sizes of 64 and 1500 octets still persist. The inevitable result of faster transmission speeds with constant packet sizes results in an upper bound of the number of packets per second increasing more than 100-fold over the past decade, in line with the increase of deployed transmission speeds from 2.5 to 400 Gbps. As a consequence, silicon-based switches are demanding higher packet-processing rates. But one really important scaling factor has not changed for the past decade, namely the clock speed of processors and the cycle time of memory, which have not moved at all. The response so far has been in increasing reliance of parallelism in high-speed digital switching applications, and these days multi-core processors and highly parallel memory systems are used to achieve performance that would be impossible in a single threaded processing model.

In 2018, it appears that we are close to achieving 1-Tbps optical systems and up to 20 Gbps in radio systems. Just how far and how quickly these transmission models can be pushed into supporting ever-higher channel speeds is an open question.

The IP Layer

The most notable aspect of the network that appears to stubbornly resist all forms of pressure over the last decade, including some harsh realities of acute scarcity, is the observation that we are still running what is essentially an IPv4 Internet.

Over the past decade, we have exhausted our pools of remaining IPv4 addresses, and in most parts of the world the IPv4 Internet is running on some form of empty. We had never suspected that the Internet would confront the exhaustion of one of its most fundamental pillars—the basic function of uniquely addressing connected devices—and apparently shrug it off and continue on blithely. But, unexpectedly, that's exactly what has happened.

Today we estimate that some 3.4 billion people regularly use the Internet, and some 20 billion devices are connected to it. We have achieved this feat by using some 3 billion unique IPv4 addresses. Nobody thought that we could achieve this astonishing feat, yet it has happened with almost no fanfare.

Back in the 1990s we had thought that the prospect of address exhaustion would propel the Internet to use IPv6, which was the successor IP protocol that comes with a four-fold increase in the bit width of IP addresses. By increasing the IP address pool to some esoterically large number of unique addresses (340 *undecillion* addresses, or 3.4×10^{38}), we would never have to confront network address exhaustion again. But this transition was not going to be easy. There is no backward compatibility in this protocol transition, so everything has to change. Every device, every router, and even every application needs to change to support IPv6. Rather than perform comprehensive protocol surgery on the Internet and change every part of the infrastructure to support IPv6, we changed the basic architecture of the Internet instead. Oddly enough, it looks like this option was the cheaper one!

Through the almost ubiquitous deployment of *Network Address Translators* (NATs)[6, 7] at the edges of the network, we've transformed the network from a *peer-to-peer* network into a *client/server* network. In today's client/server Internet clients can talk to servers, and servers can talk back to these connected clients, but that's it. Clients cannot talk directly to other clients, and servers need to wait for the client to initiate a conversation in order to talk to a client. Clients "borrow" an endpoint address when they are talking to a server and release this address for use by other clients when they are idle. After all, endpoint addresses are only useful to clients in order to talk to servers. The result is that we've managed to cram some 20 billion devices into an Internet that has deployed only 3 billion public address slots. We've achieved this result though embracing what could be described as *time-sharing* of IP addresses.

All well and good, but what about IPv6? Do we still need it? If so, then are we going to complete this protracted transition? Ten years later the answer to these questions remains unclear. On the positive side, there is a lot more IPv6 usage around now than there was 10 years ago. *Internet Service Providers* (ISPs) are deploying much more IPv6 today than they did in 2008. When IPv6 is deployed within a service provider's network, we see an immediate uptake from these IPv6-equipped devices. In 2018, it appears that one-fifth of Internet users (that itself is now estimated to number around one-half of the human population on the planet) are capable of using the Internet over IPv6, and most of this capability has developed in the past 10 years. However, on the negative side the question must be asked: What's happening with IPv6 for the other four-fifths of the Internet? Some ISPs have made the case that they would prefer to spend their finite operating budgets on other areas that improve their customers' experience, such as increasing network capacity, removing data caps, or acquiring more on-net content. Such ISPs continue to see deployment of IPv6 as a deferrable measure.

It seems that today we are still seeing a mixed picture for IPv6. Some service providers simply see no way around their particular predicament of IPv4 address scarcity, and these providers see IPv6 as a necessary decision to further expand their network. Other providers are willing to defer the question to some undefined point in the future.

Routing

While we are looking at what's largely unchanged over the past decade we need to mention the routing system. Despite dire predictions of the imminent scaling death of the *Border Gateway Protocol* (BGP)[8] 10 years ago, BGP has steadfastly continued to route the entire Internet. Yes, BGP is as insecure as ever, and yes, a continual stream of fat-finger foul-ups and less common but more concerning malicious route hijacks continue to plague our routing system, but the routing technologies used in 2008 are the same as those we use in today's Internet.

The size of the IPv4 routing table has tripled in the past 10 years, growing from 250,000 entries in 2008 to slightly more than 750,000 entries today. The IPv6 routing story is more dramatic, growing from 1,100 entries to 52,000 entries. Yet BGP just quietly continues to work efficiently and effectively. Who would've thought that a protocol that was originally designed to cope with a few thousand routes announced by a few hundred networks could still function effectively across a routing space approaching a million routing entries and a hundred thousand networks!

In the same vein, we have not made any major change to the operation of our interior routing protocols. Larger networks still use either *Open Shortest Path First* (OSPF)[9] or *Intermediate System-to-Intermediate System* (IS-IS) depending on their circumstances, while smaller networks may opt for some distance vector protocol like *Routing Information Protocol Version 2* (RIPv2)[10] or even *Enhanced Interior Gateway Routing Protocol* (EIGRP)[11]. The work in the *Internet Engineering Task Force* (IETF) on more recent routing protocols such as the *Locator Identifier Separation Protocol* (LISP)[12] and the *Babel Routing Protocol*[13] seem to lack any real traction with the Internet at large. While they both have interesting properties in routing management, neither has a sufficient level of perceived benefit to overcome the considerable inertia of conventional network design and operation. Again, this example looks like another instance where inertial mass is exerting its influence to resist change in the network.

Network Operations

Speaking of network operation, we are seeing some stirrings of change, but it appears to be a rather conservative area, and adoption of new network management tools and practices takes time.

The Internet converged on using the *Simple Network Management Protocol* (SNMP) a quarter of a century ago, and despite its security weaknesses, its inefficiency, its incredibly irritating use of *Abstract Syntax Notation One* (ASN.1), and its use in sustaining some forms of *Distributed Denial-of-Service* (DDoS) attacks, it still enjoys widespread use. But SNMP is only a network monitoring protocol, not a network configuration protocol, as anyone who has attempted to use SNMP write operations can attest.

The more recent *Network Configuration Protocol* (NETCONF) and the *Yet Another Next Generation* (YANG) data modelling language are attempting to pull this area of configuration management into something a little more usable than *Command-Line Interface* (CLI) scripts driving interfaces on switches. At the same time, we are seeing orchestration tools such as *Ansible, Chef, Network Automation and Programmability Abstraction Layer with Multivendor* (NAPALM) and SALT enter the network operations space, permitting the orchestration of management tasks over thousands of individual components. These network operations management tools are welcome steps forward to improve the state of automated network management, but it's still far short of a desirable endpoint.

In the same time period as we appear to have advanced the state of automated control systems to achieve the driverless autonomous car, the task of fully automated network management appears to have fallen way short of the desired endpoint. Surely it must be feasible to feed an adaptive autonomous control system with the network infrastructure and available resources, and allow the control system to monitor the network and modify the operating parameters of network components to continuously meet the service-level objectives of the network? Where's the driverless car for driving networks? Maybe the next 10 years might get us there.

The Mobile Internet

Before we move up a layer in the Internet Protocol model and look at the evolution of the end-to-end transport layer, we probably need to talk about the evolution of the devices that connect to the Internet.

For many years, the Internet was the domain of the desktop personal computer, with laptop devices serving the needs to those with a desire for a more portable device. At the time the mobile phone was still just a phone, and its early forays into the data world were unimpressive.

Apple's iPhone, released in 2007, was a revolutionary device. Boasting a vibrant-colour touch-sensitive screen, just four keys, a fully functional operating system with Wi-Fi and cellular radio interfaces, and a capable processor and memory, its entry into the consumer market space was perhaps the major event of the decade. Apple's early lead was rapidly emulated by Windows and Nokia with their own offerings.

Google's position was more as an active disruptor, using an open licensing framework for the Android platform and its associated application ecosystem to empower a collection of handset assemblers. Samsung, LG, HTC, Huawei, Sony, and Google, to name a few, all use Android. These days almost 80% of the mobile platforms use Android, and some 17% use Apple's iOS.

For the human Internet, the mobile market is now the Internet-defining market in terms of revenue. There is little in terms of margin or opportunity in the wired network these days, and even the declining margins of these mobile data environments represent a vague glimmer of hope for the one dominant access provider industry.

Essentially, the public Internet is now a platform of apps on mobile devices.

End-to-End Transport Layer

It's time to move up a level in the protocol stack and look at end-to-end transport protocols and changes that have occurred in the past decade.

End-to-end transport was the revolutionary aspect of the Internet, and the *Transmission Control Protocol* (TCP)[14] was at the heart of this change. Many other transport protocols require the lower levels of the network protocol stack to present a reliable stream interface to the transport protocol. It was up to the network to create this reliability, performing data integrity checks and data flow control, and repairing data loss within the network as it occurred. TCP dispensed with all of that, and simply assumed an unreliable datagram transport service from the network and pushed the responsibility for data integrity and flow control to the transport protocol.

In the world of TCP, not much appears to have changed in the past decade. We've seen some further small refinements in the details of the TCP controlled rate increase and rapid rate decrease, but nothing that shifts the basic behaviours of this protocol. TCP tends to use packet loss as the signal of congestion and oscillates its flow rate between some lower rate and this loss-triggering rate.

Or at least that was the case until quite recently. The situation is poised to change, and change in a very fundamental way, with the debut of Google's offerings of *Bottleneck Bounded Rate* (BBR) and *Quick UDP Internet Connections* (QUIC).

The BBR control algorithm is a variant of the TCP flow-control protocol that operates in a very different mode from other TCP protocols. BBR attempts to maintain a flow rate that sits exactly at the delay bandwidth product of the end-to-end path between sender and receiver. In so doing, BBR tries to avoid the accumulation of data buffering in the network (when the sending rate exceeds the path capacity), and also tries to avoid leaving idle time in the network (where the sending rate is less than the path capacity).

The side effect is that BBR tries to avoid the collapse of network buffering when congestion-based loss occurs. BBR achieves significant efficiencies from both wired and wireless network transmission systems.

The second recent offering from Google also represents a significant shift in the way we use transport protocols. The QUIC protocol looks like a *User Datagram Protocol* (UDP) protocol from the perspective of the network. But in this case looks are deceiving. The inner payload of these UDP packets contain a more conventional TCP flow-control structure and a TCP stream payload. However, QUIC encrypts its UDP payload so the entire inner TCP control is completely hidden from the network. The ossification of the Internet transport is due in no small part to the intrusive role of network middleware that is used to discard packets that it does not recognise. Approaches such as QUIC allow applications to break out of this regime and restore end-to-end flow management as an end-to-end function without any form of network middleware inspection or manipulation. I'd call this development as perhaps the most significant evolutionary step in transport protocols over the entire decade.

The Application Layer

Let's keep on moving up the protocol stack and look at the Internet from the perspective of the applications and services that operate across the network.

Privacy and Encryption

As we noted in looking at developments in end-to-end transport protocols, encryption of the QUIC payload is not just to keep network middleware from meddling with the TCP control state, although it does successfully achieve that objective. The encryption applies to the entire payload, and it points to another major development in the past decade. We are now wary of the extent to which various forms of network-based mechanisms are used to eavesdrop on users and services. The documents released by Edward Snowden in 2013 portrayed a very active US Government surveillance program that used widespread traffic-interception sources to construct profiles of user behaviour and inference profiles of individual users. In many ways this effort to assemble such profiles is not much different from what advertising-funded services such as Google and Facebook have been (more or less) openly doing for years, but perhaps the essential difference is that of knowledge and implied consent. In the advertisers' case this information is intended to increase the profile accuracy and hence increase the value of the user to the potential advertiser. The motivations of government agencies are more open to various forms of interpretation, and not all such interpretations are benign.

One technical response to the implications of this leaked material has been an overt push to embrace end-to-end encryption in all parts of the network. The corollary has been an effort to allow robust encryption to be generally accessible to all, and not just a luxury feature available only to those who can afford to pay a premium.

The *Let's Encrypt*[15] initiative has been incredibly successful in publishing X.509 domain name certificates that are free, and the result is that all network service operators, irrespective of their size or relative wealth, can afford to use encrypted sessions, in the form of *Transport Layer Security* (TLS)[16], for their web servers.

The push to hide user traffic from the network and network-based eavesdroppers extends far beyond QUIC and TLS session protocols. The *Domain Name System* (DNS) is also a rich source of information about what users are doing; it also is used in many places to enforce content restrictions. There have been recent moves to try to clean up the overly chatty nature of the DNS protocol, using query name minimisation to prevent unnecessary data leaks, and developing both DNS over TLS and DNS over *Secure HTTP* (HTTPS) to secure the network path between a stub resolver and its recursive server. This effort is very much a work in progress at present, and it will take some time to see if the results of this work will be widely adopted in the DNS environment.[20, 21]

We are now operating our applications in an environment of heightened paranoia. Applications do not necessarily trust the platform on which they are running, and we are seeing efforts from the applications to hide their activity from the underlying platform. Applications do not trust the network, and are increasingly using end-to-end encryption to hide their activity from network eavesdroppers. The use of identity credentials within the encrypted session establishment also acts to limit the vulnerability of application clients to be misdirected to masquerading servers.

The Rise and Rise of Content

Moving further up the protocol stack to the environment of content and applications, we have also seen some revolutionary changes over the past decade.

For a small period of time the content and carriage activities of the Internet existed in largely separate business domains, tied by mutual interdependence. The task of carriage was to carry users to content, which implied that carriage was essential to content. But at the same time a client/server Internet bereft of servers is useless, so content is essential to carriage. In a world of re-emerging corporate behemoths, such mutual interdependence is unsettling, both to the actors directly involved and to the larger public interest.

The content industry is largely the more lucrative of these two industries and enjoys far less in the way of regulatory constraint. There is no concept of any universal service obligation, or even any effective form of price control in the services content providers offer. Many content service providers use internal cross-funding that allows them to offer free services to the public, as in free e-mail, free content hosting, free storage, and similar services, and fund these services through a second, more occluded, transaction that essentially sells the user's consumer profile to the highest-bidding advertiser.

All this activity happens outside of any significant regulatory constraint, a situation that has given the content-services industry both considerable wealth and considerable commercial latitude.

It should be no surprise that this industry is now using its capability and capital to eliminate its former dependence on the carriage sector. We are now seeing the rapid rise of the *Content Delivery Network* (CDN) model, where instead of an Internet carrying the user to a diverse set of content stores, the content stores are opening local content outlets right next to the user. As all forms of digital services move into CDN hostels, and as the CDN opens outlets that are positioned immediately adjacent to pools of economically valuable consumers, then where does that leave the traditional carriage role in the Internet? The outlook for the public carriage providers is not looking all that rosy given this increasing marginalisation of carriage in the larger content economy.

Within these CDNs we've also seen the rise of a new service model enter the Internet in the form of cloud services. Our computers are no longer self-contained systems with processing and compute resources; they look more and more like a window that sees the data stored on a common server. Cloud services are very similar, where the local device is effectively a local cache of a larger backing store. In a world where users may have multiple devices, this model makes persuasive sense, because the view to the common backing store is constant irrespective of the device used to access the data. These cloud services also make data sharing and collaborative work far easier to support. Rather than creating a set of copies of the original document and then attempting to stitch back all the individual edits into a single common whole, the cloud model shares a document by simply altering the access permissions of the document. There is only ever one copy of the document, and all edits and comments on the document are available to all.

The Evolution of Cyber Attacks

At the same time as we have seen announcements of ever-increasing network capacity within the Internet, we've seen a parallel set of announcements that note new records in the aggregate capacity of *Denial-of-Service* (DoS) attacks. The current peak volume is an attack of some 1.7 Tbps of malicious traffic.

Attacks are now commonplace. Many of them are brutally simple, relying on a tragically large pool of potential zombie devices that are readily subverted and co-opted to assist in attacks. The attacks are often simple, such as UDP reflection attacks where a single UDP query generates a large response. The source address of the query is forged to be the address of the intended attack victim, and not much more needs to be done. A small query stream can result in a massive attack. UDP protocols such as SNMP, the *Network Time Protocol* (NTP)[17], the DNS, and *memcached* have been used in the past and doubtless will be used again.

Why can't we fix this problem? We've been trying for decades, and we just can't seem to get ahead of the attacks. Advice to network operators to prevent the leakage of packets with forged source addresses was published nearly two decades ago, in 2000.[18] Yet massive UDP-based attacks with forged source addresses still persist today. Aged computer systems with known vulnerabilities continue to be connected to the Internet and are readily transformed into attack bots.

The picture of attacks is also becoming more ominous. Although we previously attributed these hostile attacks to "hackers," we quickly realised that a significant component of them had criminal motivations. The progression from criminal actors to state-based actors is also entirely predictable, and we are seeing an escalation of this cyber warfare arena with the investment in various forms of vulnerability exploitation that are considered desirable national capabilities.

It appears that a major problem here is that collectively we are unwilling to make any substantial investment in effective defence or deterrence. The systems that we use on the Internet are overly trusting to the point of irrational credulity. For example, the public key certification system used to secure web-based transactions is repeatedly demonstrated to be entirety untrustworthy, yet that's all we trust. Personal data is continually breached and leaked, yet all we seem to want to do is increase the number and complexity of regulations rather than actually use better tools that would effectively protect users.

The larger picture of hostile attack is not getting any better. Indeed, it's getting very much worse. If any enterprise has a business need to maintain a service that is always available for use, then any form of in-house provisioning is just not enough to withstand attack. These days only a handful of platforms can offer resilient services, and even then it's unclear whether they could withstand the most extreme of attacks.

A constant background level of scanning and probing goes on in the network, and any form of visible vulnerability is ruthlessly exploited. One could describe today's Internet as a toxic wasteland, punctuated with the occasional heavily defended citadel. Those who can afford to locate their services within these citadels enjoy some level of respite from this constant profile of hostile attack, while all others are forced to try to conceal themselves from the worst of this toxic environment, while at the same time aware that they will be completely overwhelmed by any large-scale attack.

It is a sobering thought that about one-half of the world's population are now part of this digital environment. A more sobering thought is that many of today's control systems, such as power generation and distribution, water distribution, and road-traffic-control systems are exposed to the Internet.

Perhaps even more of a worry is the increasing use of the Internet in automated systems that include various life-support functions. The consequences of massive failure of these systems in the face of a sustained and damaging attack cannot be easily imagined.

The Internet of Billions of Tragically Stupid Things

What makes this scenario even more depressing is the portent of the so-called *Internet of Things* (IoT). In those circles where Internet prognostications abound and policy makers flock to hear grand visions of the future, we often hear about the boundless future represented by this Internet of Things.[19] This phrase encompasses some decades of the computing industry's transition from computers as esoteric pieces of engineering affordable only by nations to mainframes, desktops, laptops, handheld devices, and now wrist computers. Where next? In the vision of the IoT we are going to expand the Internet beyond people and press on using billions of these chattering devices in every aspect of our world.

What do we know about the "things" that are already connected to the Internet?

Some of them are not very good. In fact, some of them are just plain stupid. And this stupidity is toxic, in that their sometime-inadequate models of operation and security affect others in potentially malicious ways. If such devices were constantly inspected and managed, we might see evidence of aberrant behaviour and correct it. But these devices are unmanaged and all but invisible. Examples include the controller for a web camera, the so-called "smart" thing in a smart television, or the controls for anything from a washing machine to a goods locomotive. Nobody is looking after these devices.

When we think of an IoT we think of a world of weather stations, webcams, "smart" cars, personal fitness monitors, and similar things. But what we tend to forget is that all of these devices are built on layers of other people's software that is assembled into a product at the cheapest possible price point. It may be disconcerting to realise that the web camera you just installed has a security model that can be summarised with the phrase: "no security at all," and it's actually offering a view of your house to the entire Internet. It may be slightly more disconcerting to realise that your electronic wallet is on a device that is using a massive compilation of open source software of largely unknown origin, with a security model that is not completely understood, but appears to be susceptible to be coerced into being a "yes, take all you want" device.

It would be nice to think that we've stopped making mistakes in code, and from now on our software in our things will be perfect. But that's hopelessly idealistic. It's just not going to happen. Software will not be perfect. It will continue to have vulnerabilities.

It would be nice to think that this Internet of Things is shaping up as a market where quality matters, and consumers will select a more expensive product even though its functional behaviour is identical to a cheaper product that has not been robustly tested for basic security flaws. But that too is hopelessly naive.

The Internet of Things will continue to be a marketplace where the compromises between price and quality will continue to push us on to the side of cheap rather than secure. What's going to stop us from further polluting our environment with a huge and diverse collection of programmed unmanaged devices with inbuilt vulnerabilities that will be all too readily exploited? What can we do to make this world of these stupid cheap toxic things less stupid and less toxic? So far we have not found workable answers to this question.

The Next 10 Years

The silicon industry is not going to shut down anytime soon. It will continue to produce chips with more gates, finer tracks, and more stacked layers for some years to come. Our computers will become more capable in terms of the range and complexity of the tasks that they will be able to undertake.

At the same time, we can expect more from our network. Higher capacity certainly, but also greater levels of customisation of the network to our individual needs.

However, I find it extremely challenging to be optimistic about security and trust in the Internet. We have made little progress in this area over the last 10 years, and there is little reason to think that the picture will change in the next 10 years. If we can't fix it, then, sad as it sounds, perhaps we simply need to come to terms with an Internet jammed full of tragically stupid things

However, beyond these broad-brush scenarios, it's hard to predict where the Internet will head. Technology does not follow a predetermined path. It's driven by the vagaries of an enthusiastic consumer marketplace that is readily distracted by colourful bright shiny new objects, and easily bored by what we quickly regard as commonplace.

What can we expect from the Internet in the next 10 years that can outdo a pocket-sized computer that can converse with me in a natural language? That can offer more than immersive 3D video with outstanding quality? That can bring the entire corpus of humanity's written work into a searchable database that can answer any of our questions in mere fractions of a second?

Personally, I have no clue what to expect from the Internet. But no matter what manages to capture our collective attention, I am pretty confident that it will be colourful, bright, shiny, and entirely unexpected!

References and Further Reading

[0] Vint Cerf, "A Decade of Internet Evolution," *The Internet Protocol Journal,* Volume 11, No. 2, June 2008.

[1] Geoff Huston, "A Decade in the Life of the Internet," *The Internet Protocol Journal*, Volume 11, No. 2, June 2008.

[2] Geoff Huston, "QoS — Fact or Fiction?" *The Internet Protocol Journal*, Volume 3, No. 1, March 2000.

[3] Metcalf's Law:
`https://en.wikipedia.org/wiki/Metcalfe%27s_law`

[4] T. Sridhar, "Cloud Computing—A Primer: Part One," *The Internet Protocol Journal,* Volume 12, No. 3, September 2009.

[5] T. Sridhar, "Cloud Computing—A Primer: Part Two," *The Internet Protocol Journal,* Volume 12, No. 4, December 2009.

[6] Geoff Huston, "Anatomy: Inside Network Address Translators," *The Internet Protocol Journal,* Volume 7, No. 3, September 2004.

[7] Geoff Huston, "In Defence of NATs," *The Internet Protocol Journal,* Volume 20, No. 3, November 2017.

[8] Geoff Huston, "The BGP Routing Table," *The Internet Protocol Journal,* Volume 4, No. 1, March 2001.

[9] Dennis Ferguson, Acee Lindem, and John Moy, "OSPF for IPv6," RFC 5340, July 2008.

[10] Gary Scott Malkin, "RIP Version 2," RFC 2453, November 1998.

[11] Donald Slice, Peter Paluch, Donnie Savage, Russ White, Steven Moore, and James Ng, "Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)," RFC 7868, May 2016.

[12] David Meyer, "The Locator Identifier Separation Protocol (LISP)," *The Internet Protocol Journal,* Volume 11, No. 1, March 2008.

[13] Juliusz Chroboczek, "The Babel Routing Protocol," RFC 6126, April 2011.

[14] Geoff Huston, "The Future for TCP," *The Internet Protocol Journal,* Volume 3, No. 3, September 2000.

[15] Daniel McCarney, "Automatic Certificate Management," *The Internet Protocol Journal,* Volume 20, No. 2, June 2017.

[16] William Stallings, "SSL: Foundation for Web Security," *The Internet Protocol Journal*, Volume 1, No. 1, June 1998.

[17] Geoff Huston, "Network Time Protocol," *The Internet Protocol Journal*, Volume 15, No. 4, December 2012.

[18] Paul Ferguson and Daniel Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing," RFC 2827, May 2000.

[19] Bob Hinden, "The Internet of Insecure Things," *The Internet Protocol Journal,* Volume 20, No. 1, March 2017.

[20] Geoff Huston and Joao Luis Silva Dama, "DNS Privacy," *The Internet Protocol Journal,* Volume 20, No. 1, March 2017.

[21] Zi Hu, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessels, and Paul Hoffman, "Specification for DNS over Transport Layer Security (TLS)," RFC 7858, May 2016.

GEOFF HUSTON, B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for building the Internet within the Australian academic and research sector in the early 1990s. He is author of numerous Internet-related books, and was a member of the Internet Architecture Board from 1999 until 2005. He served on the Board of Trustees of the Internet Society from 1992 until 2001. At various times Geoff has worked as an Internet researcher, an ISP systems architect, and a network operator. E-mail: **`gih@apnic.net`**

Fileless Malware

by David Strom

Malware authors have gotten more clever and sneaky over time to make their code more difficult to detect and prevent. One of the more worrying recent developments goes under the name "fileless." There is reason to worry because these kinds of attacks can do more damage and the malware can persist on your computers and networks for weeks or months until they are finally neutralized. Let's talk about what this malware is and how to understand it better so we can try to stop it from entering our networks to begin with.

Usually, the goal of most malware is to leave something behind on one of your endpoints: one or more files that contain an executable program that can damage your computer, corral your PC as part of a botnet, or make copies of sensitive data and move them to an external repository. Over the years, various detection products have gotten better at finding these *residues*, as they are called, and blocking them.

But the malware game is one of "cat-and-mouse," and as defenders get better at stopping the malware, the malware authors get better at evading these blockades. Back in the early days of the Internet, most blocking routines looked for certain signatures, either as the name of one of the running programs on your computer or specific patterns of behavior across your network. These options worked until the malware authors got better at hiding their signature moves.

This point is where the fileless versions come into play. They aim to leave as little residue as possible, so the detection products can't easily find them. Or better yet, to do something misleading, or under the guise of something that an uninfected operating system might do.

Actually, the fileless designation is somewhat of a misnomer since there is still something left behind. It may not be a complete executable file or *Dynamic Linked Library* (DLL), but enough of some code is used to actually be able to run some series of processes that can do the "dirty work" of the malware. Starting in 2016, researchers began to see more of these fileless efforts from attackers, and they have continued to become more popular because the malware can be a powerful infection that is neither easily found nor prevented.[1]

Fileless Attack Types

Fileless malware uses three different attack types: *Return-Oriented Programming, Scripting-Based Attacks,* and *Polymorphic Attacks.* Each is somewhat different.

The first type of attack is called *Return-Oriented Programming*, which is the most popular and could be considered the "classic" version. The malware can execute standard DLLs and other sequences of code that can compromise an otherwise uninfected system. This code could also be part of your desktop web browser, or common *Operating System* (OS) tools such as desktop applications. Since the code is already present in these operating system functions, there is no particular "file" actually being run that is unique to the malware itself. Instead, the malware author piggybacks on these routines to get the job done.

To make this kind of attack work, you have to be familiar with the program code that you intend to hijack for evil purposes, and be reasonably assured that the target endpoint is running the particular version of code for that operating system. Small variations in OS versions, such as from Windows 7 to 7.1 or MacOS 10.12.5 to 10.12.6, could foil the attack because the code base has changed. Or if Microsoft or Apple (in particular, given their popularity and installed base) has issued a patch to fix the potential exploit.

Scripting-Based Attacks are the second fileless category. Another avoidance technique is to execute malware using built-in Windows scripting engines such as Microsoft Office, Windows *PowerShell*, or Microsoft's HTML Application Host. These attacks typically take advantage of process hooking and don't leave any file-based residues on the endpoint. If your detection systems can't see the script execution or understand the command-line arguments, you can't readily figure out that it is malware.

For example, a typical malware script allocates memory, resolves Windows application program interfaces, and downloads some executable directly to the memory of the target PC. After it gets in memory, it starts up a malicious service and begins to use the target to explore the local network to find other targets, often by starting other malicious PowerShell scripts that use privilege escalation and remote execution. That is a lot of stuff going on to avoid detection and to do the dirty work of the malware. But most of these activities can operate "under the radar," and perhaps be discovered only months later with detailed forensic analysis that can capture the sequence of events that played out for the particular attack.

Scripting attacks are gaining favor, mainly because there is so much built-in software on a typical modern PC that can do most of what a piece of malware needs to do: to access a shared network drive, copy portions of files, set up some sort of monitoring tool, and so forth. Why reinvent the criminal's wheel when it is sitting on the average desktop or laptop?

A third method is called *Polymorphic Attacks*. These attacks adapt to a variety of conditions, operating systems, and circumstances and try to evade security scans and protection products to infect your endpoints. They are called polymorphic because they shift their signatures, attack methods, and targets so that you can't easily identify and catch them. Attackers typically use polymorphism as just one of many code obfuscation methods to hide from defenders, such as determining if they are running inside a *Virtual Machine* (a favorite ploy researchers use), or encrypting their code to mask their executables.

Over the past several years, security vendors have begun to take this notion of polymorphism that attackers use and turn it into a defensive maneuver. The idea is to make a target Web server or other piece of network infrastructure appear to change frequently so it can't be easily identified or infected. Sometimes this method is called a *moving-target defense*, which could be a synonym or refer to some aspect of the defense that changes the nature of your applications or code locations. These vendors include Morphisec, Shape Security, and Polyverse, all startup companies. One startup, CyActive, was successful enough to be purchased by PayPal.

Polymorphic defenses can limit the amount of time a potential attacker can invade a network, since their target system appears to move around the network or change properties.

Researchers are seeing combinations of all three types of attacks to make them even more sophisticated and difficult to track down. Sometimes, malware authors program multiple attack types to ensure that something will evade your defenses and penetrate your network. As I said earlier, it is a game of cat and mouse.

Fileless Samples

Let's look at a few recent examples of fileless malware to illustrate the differences and how fileless malware has evolved over time.

Back in 2014, the retailer Target experienced a now-infamous breach. It turns out malware was placed on its network through a very simple strategy: someone's network access credentials were discovered and copied, in this case belonging to an employee of Target's heating vendor. What is noteworthy about this attack is its simplicity, and the fact that Target's network was a flat topology with no *virtual LANs* (vLANs) or other segments. This example is a reminder that bad network practice can make any kind of malware—fileless or otherwise—dangerous. Brian Krebs studied what went wrong and reported on this attack in his blog.[2]

As most of us know by now, back in 2016, the *Democratic National Committee* (DNC) was hacked. The attack used a fileless malware product that took advantage of both PowerShell and *Windows Management Instrumentation* (WMI) in order to "get a foot into the door" of the political party's systems.

WMI is commonly used for day-to-day management tasks such as deploying automation scripts, running a process at a given time, or getting information about the installed applications or hardware.

The DNC malware also used PowerShell as a staging tool to execute other scripts to compromise a system. It used WMI to install backdoors that allow persistence by enabling the adversary to launch malicious code automatically, after a specified period of system uptime or according to a specific schedule. Again, the malware used all of these tactics to avoid detection.[3]

August Stealer was discovered at the end of 2016 and was attributed to the TA530 criminal group. Targeted at customer service and call center staffs, it used infected Word macros and PowerShell scripts that were delivered via phished e-mails. The e-mails were designed to look like queries from users over support issues and used various subject lines such as the following:

- Erroneous charges from [recipient's domain]
- [recipient's domain] - Help: Items vanish from the cart before checkout
- [recipient's domain] Support: Products disappear from the cart during checkout
- Need help with order on [recipient's domain]
- Duplicate charges on [recipient's domain]

August contains stealing functionality targeting credentials, cryptocurrency wallets, and sensitive documents from the infected computers.[4]

Discovered earlier in 2017, *Duqu2* is a good example of the first fileless malware products. The malware was found in more than 140 enterprise networks of banks, government offices, and telecom companies across 40 different countries.

It takes the form of a malicious PowerShell script and a series of the following Windows Registry values that at the time were unique to the malware and used to identify the infected systems:

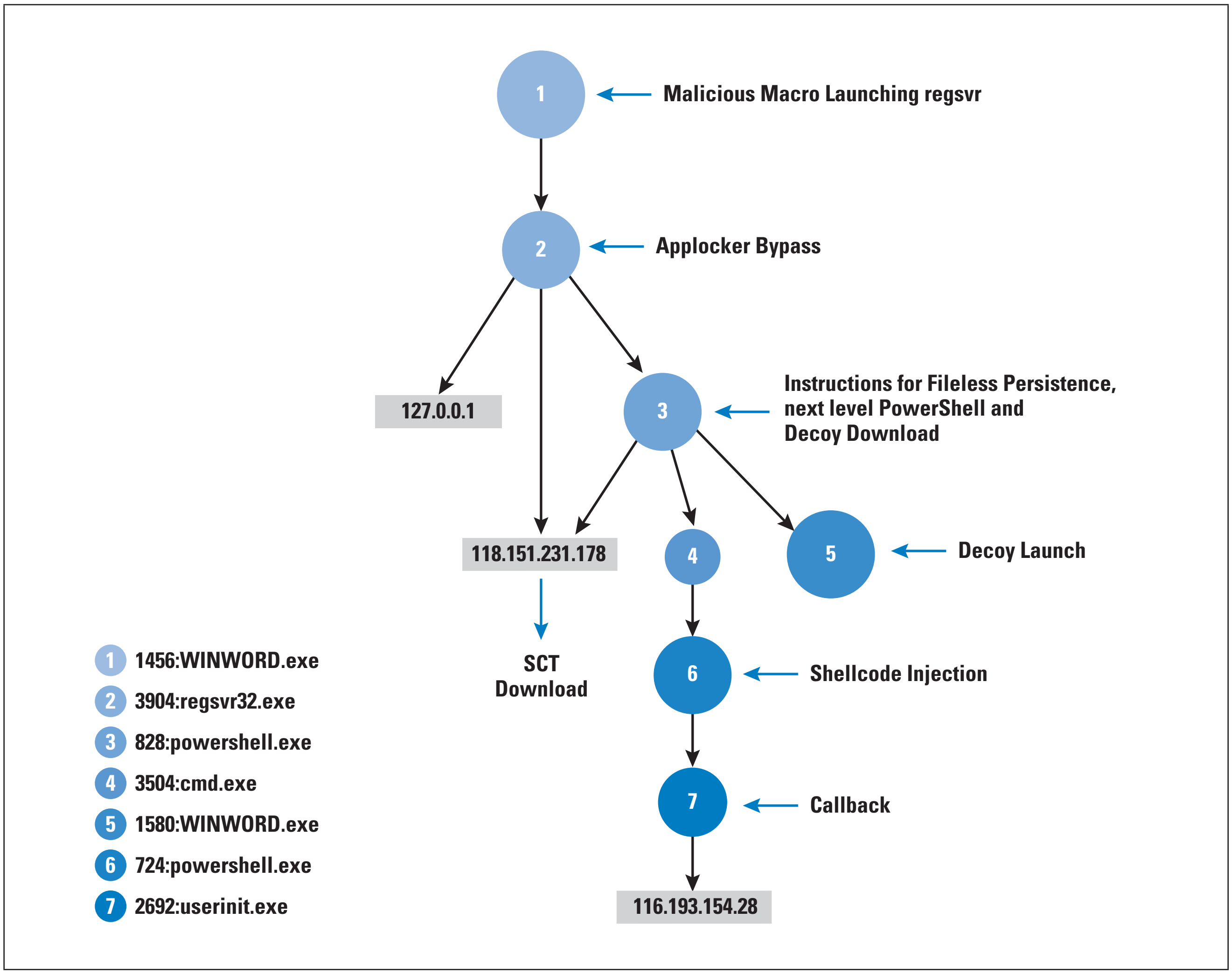
- **`HKLM\SYSTEM\ControlSet001\services\`** – path is modified after using the SC utility
- **`HKLM\SYSTEM\ControlSet001\services\PortProxy\v4tov4\tcp`** – path is modified after using the NETSH utility

After finding its way onto the target hard drives, it then starts up via a malicious Windows installer or MSI file, which then deletes itself and renames various files to hide its operations. After the malware is installed on a PC, it just runs in the memory of the PC.

"That's why memory forensics is critical to the analysis of malware and its functions. In fact, detection of this attack would be possible in memory, network, and Windows Registry only," says one group of researchers from Kaspersky Labs that studied its operations.[5] Obviously, running in memory means the Duqu2 malware won't last after the PC is rebooted—one drawback of many fileless products.

Poison Ivy, also discovered earlier this year, is an example of fileless malware that was used on a specific target, in this case Mongolian government officials. It takes the form of a malicious Microsoft Word macro. If the target has enabled macros—which is a typical setting for most users—it runs and creates a remote-access connection to log keystrokes and capture screens and videos from the PC. All these actions are done from memory-resident programs taking advantage of certain PowerShell command sequences. Figure 1 shows its various modules (courtesy of FireEye).

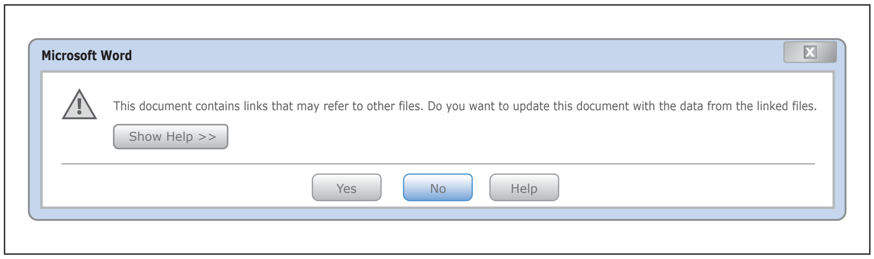
Figure 1: Poison Ivy Modules



Poison Ivy also tried to evade detection by Microsoft's *AppLocker* protection system by inserting a reference to itself in AppLocker's whitelisted applications using a series of Windows programs and scripts. It also created a series of decoy documents to make its operations seem benign to the infected user. As you can see, this software is very complex, with several different stages and methods to find its way into a user's PC. It has also been used in other circumstances besides the Mongolian case.[6]

Other targeted fileless campaigns, such as the *OilRig* malware[7], have been attributed to Iranian state-sponsored actors. This campaign targeted 250 e-mail accounts of various Israelis, including ironically cybersecurity researchers at Ben Gurion University. While Microsoft released a patch back in April 2017 that prevents this malware from spreading, many enterprises haven't yet applied it. Ironically (again), the malware authors used the details from a published proof-of-concept to design their tool accordingly.

This particular malware used an infected Word document that was sent as an attachment and used to steal information from targeted PCs. It used a specialized fileless version of the *Helminth Trojan* malware. Earlier versions of OilRig used infected macros, but this attack used an embedded Web link using an **`.HTA`** executable file. This type of file is automatically run by the Windows program **`MSHTA.EXE`** (for Microsoft HTML applications). Normally, when this program runs an **`.HTA`** file, it displays the following warning message:



Microsoft Word

This document contains links that may refer to other files. Do you want to update this document with the data from the linked files.

Figure 2: This warning about file permissions from Windows is only briefly seen by users when they click on an infected file.

However, this malware anticipates this situation, and automatically sends an "Enter" command so that the warning window is quickly dispatched and the malware does its business. Other targeted malware campaigns include one targeting American restaurant computers using the *Fin7* malware[8]. In the past, this malware targeted banks and government financial filing documents. And like other fileless attacks, it hides inside a Microsoft Word document that is attached to phishing e-mails. One new twist with the Fin7 restaurant attacks is that it executes various attacks completely in memory, without using any PowerShell commands.

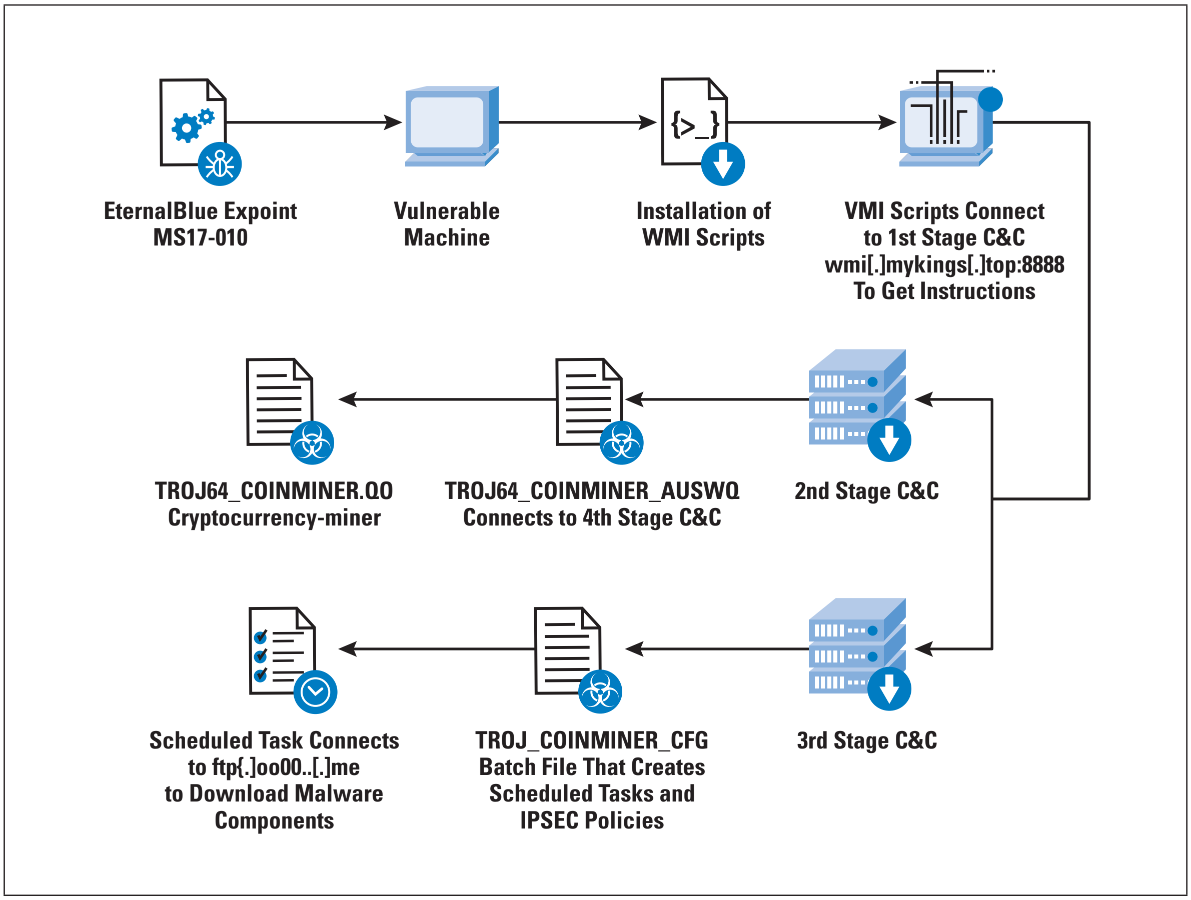
Another obfuscation technique goes by the name of *DoubleAgent*. It takes advantage of an undocumented feature in Microsoft Application Verifier. This verifier is code that has been around since at least Windows XP and is a Windows feature that lets developers do runtime verifications of their applications for finding and fixing security problems.

Unfortunately, it has an undocumented feature that security researchers from Cybellum discovered. The feature gives attackers a way to replace the legitimate verifier with a rogue one so they can gain complete control of the application. Cybellum said, “DoubleAgent gives the attacker the ability to control the AV and perform all the operations above without being detected, while keeping the illusion that the AV is working normally.”[9]

Security vendors have recently issued patches to correct this flaw, but again this example demonstrates that malware writers are getting better at finding these sorts of hidden mechanisms to avoid discovery and being blocked.

In July 2017, a new fileless malware was discovered called *CoinMiner*[10], which was found mainly in Japan and Indonesia. The purpose of this malware was to create a hidden bitcoin mining application, to generate cryptocurrency for the attacker. It uses WMI to persist beyond reboots and execute a series of scripts. CoinMiner invades a PC through the *EternalBlue* exploit[11], which is the same method that was used by the *WannaCry* worm. Figure 3 is a diagram of its logic flow (courtesy of Trend Micro):

Figure 3: CoinMiner Logic Flow



Common Prevention Steps

Given the scope of these exploits, here are a few steps to take to prevent infections across your network:

- Apply patches quickly and across all systems. Microsoft issues regular patches for Windows, and the other operating system vendors do the same for their systems. Don't delay an update, because you can see some criminals take advantage of unpatched systems with their malware. The EternalBlue exploit is a good example: the patch to prevent this attack was available for more than a month before this exploit was launched.
- Segment your network carefully and make sure you understand access rights, especially of third parties.
- Restrict administrator rights to the minimum number of systems. Many of the WMI-based exploits count on profligate use of admin rights that aren't needed.
- Disable Windows programs that aren't needed, such as WMI, PowerShell, and support for ancient protocols such as *Server Message Block* (SMB) v1.
- Whitelist applications to further restrict what can run on most endpoints.

Conclusion

As you can see, the bad guys have gotten better at plying their trade, and through the use of fileless techniques, they are making their malware harder to detect and protect. Hopefully, by learning about some of these past examples, you can tune your own defenses accordingly and do a better job of keeping them infection-free.

References

[1] Ericka Chickowski, "Fileless Malware Takes 2016 By Storm," DarkReading, December 2016.
`http://www.darkreading.com/vulnerabilities---threats/fileless-malware-takes-2016-by-storm/d/d-id/1327796`

[2] "Target Hackers Broke in Via HVAC Company," Krebs on Security blog, February 2014.
`https://krebsonsecurity.com/2014/02/target-hackers-broke-in-via-hvac-company/`

[3] "DNC Hack Exhibits One of 3 Attack Trends To Watch for in 2017," Crowdstrike blog, January 2017.
`https://www.crowdstrike.com/blog/dnc-hack-exhibits-one-of-3-attack-trends-to-watch-for-in-2017/`

[4] "August in November: New Information Stealer Hits the Scene," Proofpoint, December 2016.
`https://www.proofpoint.com/us/threat-insight/post/august-in-december-new-information-stealer-hits-the-scene`

[5] "Fileless attacks against enterprise networks," Securelist, February 2017.
`https://securelist.com/fileless-attacks-against-enterprise-networks/77403/`

[6] "Poison Ivy: Assessing Damage and Extracting Intelligence," FireEye Special Report, 2014.
`https://www.fireeye.com/content/dam/fireeye-www/global/en/current-threats/pdfs/rpt-poison-ivy.pdf`

[7] Michael Gorelik, "Iranian Fileless Attack Infiltrates Israeli Organizations," Morphisec Cyber Security Blog, April 2017.
`http://blog.morphisec.com/iranian-fileless-cyberattack-on-israel-word-vulnerability`

[8] Michael Gorelik, "FIN7 Takes Another Bite at the Restaurant Industry," Morphisec Cyber Security Blog, June 2017.
`http://blog.morphisec.com/fin7-attacks-restaurant-industry`

[9] Michael Engstler, "DoubleAgent: Zero-Day Code Injection and Persistence Technique," Cybellum blog, March 2017.
`https://cybellum.com/doubleagentzero-day-code-injection-and-persistence-technique/`

[10] Buddy Tancio, "Cryptocurrency Miner Uses WMI and EternalBlue To Spread Filelessly," TrendLabs Security Intelligence Blog, August 2017.
`http://blog.trendmicro.com/trendlabs-security-intelligence/cryptocurrency-miner-uses-wmi-eternalblue-spread-filelessly/`

[11] "EternalBlue," Wikipedia article,
`https://en.wikipedia.org/wiki/EternalBlue`

DAVID STROM has written for *The Internet Protocol Journal* before on e-mail topics; he runs the *Inside.com* Security e-mail newsletter. He was the founding editor-in-chief of *Network Computing* (USA) magazine and is the co-author of the 1998 book, *The Internet Message: Closing the Book with Electronic Mail,* with Marshall T. Rose. E-mail: `david@strom.com`

Postel Award Presented to Steven G. Huter

The Internet Society, a global non-profit dedicated to ensuring the open development, evolution, and use of the Internet, recently presented the prestigious *Jonathan B. Postel Service Award* to Steven G. Huter, Director for the *Network Startup Resource Center* (NSRC) and a Research Associate at the University of Oregon. For decades he has worked with people around the world to strengthen the infrastructure, partnerships, and expertise upon which the Internet has been developed in more than 120 countries, particularly in support of research and education.

© Stonehouse Photographic/Internet Society

"Steve Huter is the quintessential candidate for the Postel Award. For a quarter of a century, Steve has enabled hundreds of institutions to build and operate new components of the Internet. His dedication to this task mirrors Postel's own and continues to this day. Literally millions have benefited from Steve's work," explains Vint Cerf, founding president of the Internet Society.

Mr. Huter was selected by an international award committee comprised of former Postel Award winners. The committee placed particular emphasis on candidates who have supported and enabled others in addition to their own contributions. The award is being presented to Mr. Huter in recognition of "his leadership and personal contributions at the Network Startup Resource Center that enabled countless others to develop the Internet in more than 120 countries." The NSRC was formally begun in 1992 by Randy Bush and John Klensin with funding from a U.S. National Science Foundation grant to provide technical assistance to people setting up networks in developing areas to support scientific collaboration.

"Steve epitomizes the values and spirit of the Postel Award. For more than twenty-five years he has energetically brought the fruits of the Internet to developing countries using his unique combination of a multicultural background, technical knowledge, unfailing energy and commitment," adds Steve Crocker, CEO and co-founder of Shinkuro, Inc.

Mr. Huter joined the NSRC in 1993, where he has led the development and implementation of programs that provide technical training, equipment, and expertise across Africa, Asia-Pacific, Latin America-Caribbean, and the Middle East.

"It is a tremendous honor to be acknowledged for helping to advance Jon's vision and philosophy of developing the Internet into a global resource," said Mr. Huter on receiving the award.

“The most important thing I learned from Jon Postel and the founders of the NSRC is to cultivate a culture of network operators who help each other via technical exchange and resource sharing; this is an effective way to empower more network engineers and enable continuous progress for a community of peers in all regions of the world. Thank you to the NSRC team and all who have contributed over the years towards achieving this objective and enriching the Internet.”

The Postel Award was established by the Internet Society to honor individuals or organizations that, like Jon Postel, have made outstanding contributions to the data communications community. The award is focused on sustained and substantial technical contributions, service to the community, and leadership. Kathy Brown, President and CEO of the Internet Society presented the award including a US$20,000 honorarium and a crystal engraved globe, during the 102nd meeting of the *Internet Engineering Task Force* (IETF) held in Montreal, Canada, July 14–20, 2018.

Founded by Internet pioneers, the *Internet Society* (ISOC) is a non-profit organization dedicated to ensuring the open development, evolution and use of the Internet. Working through a global community of chapters and members, the Internet Society collaborates with a broad range of groups to promote the technologies that keep the Internet safe and secure, and advocates for policies that enable universal access. The Internet Society is also the organizational home of the IETF.

The NSRC, which is based at the University of Oregon, was established in 1992 to provide technical assistance to organizations setting up computer networks in new areas to connect scientists engaged in collaborative research and education. For the past few decades, the NSRC has helped develop Internet infrastructure and network operations communities in Africa, Asia-Pacific, Latin America-Caribbean, and the Middle East. The NSRC is partially funded by the *International Research Network Connections* (IRNC) program of the U.S. National Science Foundation and Google, with additional contributions from dozens of public and private organizations.

Check your Subcription Details!

If you have a print subscription to this journal, you will find an expiration date printed on the back cover. For the last couple of years, we have “auto-renewed” your subscription. Now we ask that you log in and perform this simple task yourself. The subscription portal is here: `https://www.ipjsubscription.org/` This process will ensure that we have your current contact information, as well as delivery preference (print edition or download). For any questions, contact us by e-mail at: `ipj@protocoljournal.org`

Thank You!

Publication of IPJ is made possible by organizations and individuals around the world dedicated to the design, growth, evolution, and operation of the global Internet and private networks built on the Internet Protocol. The following individuals have provided support to IPJ. You can join them by visiting **`http://tinyurl.com/IPJ-donate`**

Fabrizio Accatino
Scott Aitken
Jacobus Akkerhuis
Antonio Cuñat Alario
Matteo D'Ambrosio
Jens Andersson
Danish Ansari
Tim Armstrong
Richard Artes
David Atkins
Jaime Badua
Hidde Beumer
John Bigrow
Axel Boeger
Gerry Boudreaux
L de Braal
Kevin Breit
Thomas Bridge
Ilia Bromberg
Christophe Brun
Gareth Bryan
Caner Budakoglu
Stefan Buckmann
Scott Burleigh
Jon Harald Bøvre
Olivier Cahagne
Antoine Camerlo
Tracy Camp
Fabio Caneparo
Roberto Canonico
David Cardwell
John Cavanaugh
Lj Cemeras
Dave Chapman
Stefanos Charchalakis
Greg Chisholm
Marcin Cieslak
Brad Clark
Narelle Clark
Steve Corbató
Brian Courtney
Dave Crocker
Kevin Croes
John Curran
André Danthine
Morgan Davis
Jeff Day
Freek Dijkstra
Geert Van Dijk
David Dillow
Richard Dodsworth
Ernesto Doelling
Eugene Doroniuk
Karlheinz Dölger
Joshua Dreier
Lutz Drink
Andrew Dul
Holger Durer
Peter Robert Egli
George Ehlers
Peter Eisses
Torbjörn Eklöv
ERNW GmbH
ESdatCo
Steve Esquivel
Mikhail Evstiounin
Paul Ferguson
Kent Fichtner
Gary Ford
Jean-Pierre Forcioli
Christopher Forsyth
Craig Fox
Fausto Franceschini
Tomislav Futivic
Edward Gallagher
Andrew Gallo
Chris Gamboni
Xosé Bravo Garcia
Kevin Gee
John Gilbert
Serge Van Ginderachter
Greg Goddard
Octavio Alfageme Gorostiaga
Barry Greene
Martijn Groenleer
Geert Jan de Groot
Gulf Coast Shots
Sheryll de Guzman
James Hamilton
Stephen Hanna
John Handin
Martin Hannigan
John Hardin
David Harper
Edward Hauser
David Hauweele
Marilyn Hay
Headcrafts SRLS
Hidde van der Heide
Johan Helsingius
Robert Hinden
Alain Van Hoof
Edward Hotard
Bill Huber
Hagen Hultzsch
Kevin Iddles
Mika Ilvesmaki
Karsten Iwen
David Jaffe
Ashford Jaggernauth
Jozef Janitor
John Jarvis
Dennis Jennings
Edward Jennings
Aart Jochem
Richard Johnson
Jim Johnston
Jonatan Jonasson
Daniel Jones
Gary Jones
Amar Joshi
Merike Kaeo
Andrew Kaiser
Christos Karayiannis
David Kekar
Jithin Kesavan
Jubal Kessler
Shan Ali Khan
Nabeel Khatri
Anthony Klopp
Henry Kluge
Michael Kluk
Andrew Koch
Carsten Koempe
Alexader Kogan
Antonin Kral
Mathias Körber
John Kristoff
Terje Krogdahl
Bobby Krupczak
Murray Kucherawy
Warren Kumari
Darrell Lack
Yan Landriault
Markus Langenmair
Fred Langham
Richard Lamb
Tracy LaQuey Parker
Simon Leinen
Robert Lewis
Martin Lillepuu

Sergio Loreti
Guillermo a Loyola
Hannes Lubich
Dan Lynch
Miroslav Madić
Alexis Madriz
Carl Malamud
Michael Malik
Yogesh Mangar
Bill Manning
Harold March
Vincent Marchand
David Martin
Jim Martin
Timothy Martin
Gabriel Marroquin
Carles Mateu
Juan Jose Marin Martinez
Ioan Maxim
David Mazel
Miles McCredie
Brian McCullough
Joe McEachern
Jay McMaster
Mark Mc Nicholas
Carsten Melberg
Kevin Menezes
Bart Jan Menkveld
William Mills
Desiree Miloshevic
Thomas Mino
Wijnand Modderman
Mohammad Moghaddas
Charles Monson
Andrea Montefusco
Fernando Montenegro
Joel Moore
Soenke Mumm
Tariq Mustafa
Stuart Nadin
Mazdak Rajabi Nasab
Krishna Natarajan
Darryl Newman
Paul Nikolich
Marijana Novakovic
David Oates
Ovidiu Obersterescu
Mike O'Connor
Mike O'Dell
Carlos Astor Araujo Palmeira
Alexis Panagopoulos
Gaurav Panwar
Manuel Uruena Pascual
Ricardo Patara
Dipesh Patel
Alex Parkinson
Craig Partridge
Dan Paynter
Leif-Eric Pedersen
Juan Pena
Chris Perkins
David Phelan
Derrell Piper
Rob Pirnie
Jorge Ivan Pincay Ponce
Victoria Poncini
Blahoslav Popela
Tim Pozar
David Raistrick
Priyan R Rajeevan
Paul Rathbone
Bill Reid
Rodrigo Ribeiro
Justin Richards
Mark Risinger
Ron Rockrohr
Carlos Rodrigues
Lex Van Roon
William Ross
Boudhayan Roychowdhury
Carlos Rubio
Timo Ruiter
RustedMusic
Babak Saberi
George Sadowsky
Scott Sandefur
Sachin Sapkal
Arturas Satkovskis
Phil Scarr
Elizabeth Scheid
Jeroen Van Ingen Schenau
Carsten Scherb
Dan Schrenk
Richard Schultz
Roger Schwartz
SeenThere
Scott Seifel
Yury Shefer
Yaron Sheffer
Doron Shikmoni
Tj Shumway
Jeffrey Sicuranza
Thorsten Sideboard
Andrew Simmons
Pradeep Singh
Henry Sinnreich
Geoff Sisson
Helge Skrivervik
Darren Sleeth
Bob Smith
Mark Smith
Job Snijders
Ronald Solano
Asit Som
Ignacio Soto Campos
Peter Spekreijse
Thayumanavan Sridhar
Ralf Stempfer
Matthew Stenberg
Adrian Stevens
Clinton Stevens
John Streck
Viktor Sudakov
Edward-W. Suor
Vincent Surillo
T2Group
Roman Tarasov
David Theese
Douglas Thompson
Rey Tucker
Sandro Tumini
Phil Tweedie
Steve Ulrich
Unitek Engineering AG
John Urbanek
Martin Urwaleck
Betsy Vanderpool
Surendran Vangadasalam
Buddy Venne
Alejandro Vennera
Luca Ventura
Tom Vest
Dario Vitali
Randy Watts
Andrew Webster
Tim Weil
Jd Wegner
Rick Wesson
Peter Whimp
Jurrien Wijlhuizen
Pindar Wong
Romeo Zwart
Bernd Zeimetz
廖 明沂.

Follow us on Twitter and Facebook @protocoljournal https://www.facebook.com/newipj

Call for Papers

The *Internet Protocol Journal* (IPJ) is a quarterly technical publication containing tutorial articles ("What is...?") as well as implementation/operation articles ("How to..."). The journal provides articles about all aspects of Internet technology. IPJ is not intended to promote any specific products or services, but rather is intended to serve as an informational and educational resource for engineering professionals involved in the design, development, and operation of public and private internets and intranets. In addition to feature-length articles, IPJ contains technical updates, book reviews, announcements, opinion columns, and letters to the Editor. Topics include but are not limited to:

- Access and infrastructure technologies such as: Wi-Fi, Gigabit Ethernet, SONET, xDSL, cable, fiber optics, satellite, and mobile wireless.
- Transport and interconnection functions such as: switching, routing, tunneling, protocol transition, multicast, and performance.
- Network management, administration, and security issues, including: authentication, privacy, encryption, monitoring, firewalls, troubleshooting, and mapping.
- Value-added systems and services such as: Virtual Private Networks, resource location, caching, client/server systems, distributed systems, cloud computing, and quality of service.
- Application and end-user issues such as: E-mail, Web authoring, server technologies and systems, electronic commerce, and application management.
- Legal, policy, regulatory and governance topics such as: copyright, content control, content liability, settlement charges, resource allocation, and trademark disputes in the context of internetworking.

IPJ will pay a stipend of US$1000 for published, feature-length articles. For further information regarding article submissions, please contact Ole J. Jacobsen, Editor and Publisher. Ole can be reached at `ole@protocoljournal.org` or `olejacobsen@me.com`

This publication is distributed on an "as-is" basis, without warranty of any kind either express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. This publication could contain technical inaccuracies or typographical errors. Later issues may modify or update information provided in this issue. Neither the publisher nor any contributor shall have any liability to any person for any loss or damage caused directly or indirectly by the information contained herein.

Supporters and Sponsors

Supporters	*Diamond Sponsors*
Internet Society cisco	Google ICANN
Ruby Sponsors	*Sapphire Sponsors*
Your logo here!	Your logo here!

Emerald Sponsors

Afilias

APNIC

APRICOT Asia Pacific Regional Internet Conference on Operational Technologies

cisco

CNNIC 中国互联网络信息中心 CHINA INTERNET NETWORK INFORMATION CENTER

COMCAST

DE-CIX

EQUINIX

JPRS JAPAN REGISTRY SERVICES

JUNIPer NETWORKS

lacnic

Linked in

linx

NSRC Network Startup Resource Center

NTT Communications

RIPE NCC RIPE NETWORK COORDINATION CENTRE

TEAM CYMRU

VERISIGN

WIDE PROJECT

Corporate Subscriptions

AFRINIC The Internet Numbers Registry for Africa

amsix amsterdam internet exchange

DNS-OARC

IWL

ISC Internet Systems Consortium

Limelight NETWORKS

netnod

qa cafe

SIDN

For more information about sponsorship, please contact `sponsor@protocoljournal.org`

The Internet Protocol Journal
NMS
535 Brennan Street
San Jose, CA 95131

ADDRESS SERVICE REQUESTED

The Internet Protocol Journal is published quarterly and supported by the Internet Society and other organizations and individuals around the world dedicated to the design, growth, evolution, and operation of the global Internet and private networks built on the Internet Protocol.

Email: ipj@protocoljournal.org
Web: www.protocoljournal.org