RAPID7

RAPID7 RESEARCH

UNDER THE HOODIE

Lessons from a Season of Penetration Testing

By Rapid7 Global Consulting

July 24, 2018

2018

CONTENTS

EXECUTIVE SUMMARY

In 2017, Rapid7 launched the "Under the Hoodie" project to demystify the practice of penetration testing by surveying those who are in the field and conducting the investigations on what they most commonly see during client engagements. We have renewed this approach in 2018 to continue providing visibility into this often occult niche of information security. To this end, this paper presents the results of 268 engagements (251 of which involved live, production network tests), conducted from early September of 2017 through mid-June of 2018.

Rapid7 offers penetration testing services of all scopes and sizes, but in general, we find that our customers prefer external penetration tests, where the simulated attacker can only reach the target organization over the internet. Fifty-nine percent of all penetration tests performed in the survey period were externally based, where the targets tend to be internet-facing vectors such as web applications, email phishing, cloud-hosted assets, and/or VPN exposure. External penetration tests make sense for most organizations, given the preponderance of internet-based attackers. However, we always advocate for a penetration test that includes an internal component in order to understand the impact of a compromise and to quantify the gaps in an organization's defense-in-depth strategy.

The three broad categories of compromise Rapid7 penetration testers pursue are software vulnerabilities, network misconfigurations, and network credentials.

- Overall, Rapid7 penetration testers were able to exploit at least one in-production vulnerability in 84% of all engagements. That figure rises to 96% of all internally-based penetration tests.
- In a similar vein, penetration testers were able to abuse at least one network misconfiguration at a slightly lower rate of 80%, but among internal assessments, a misconfiguration was leveraged in the investigator's favor 96% of the time.
- Finally, at least one credential was captured in 53% of all engagements, and 86% of the time when looking purely at internal engagements.

Given these three basic areas of interest, penetration testers successfully gain complete administrative control of the targeted organization's network 67% of the time when the internal LAN or WLAN is in scope.

This success rate and these proportions of internal versus external weaknesses make intuitive sense to most people involved in the day-to-day work of penetration testing. However, we should not be complacent in a belief that penetration testers "always win" when they're on the inside. We believe it's important to actually measure and publish the results of our penetration testing practices in order to provide risk context to organizations when they're considering the results of their own penetration testing programs. It's also valuable to quantify penetration testing results in order to properly prioritize what defensive strategies are most effective in detecting and defending against actual breach attempts by "real" threat actors.

The rest of this paper will examine and explore the data gathered from our most recent season of penetration testing, as well as provide anecdotes drawn from real-world experiences in the field. We expect readers to come away from these pages with a baseline understanding of how penetration testers help organizations identify their own unique (and not-so-unique) IT risks.

In our 2016 study, we found that only 21% of engagements were purely internally focused, whereas this year's nearly 32% figure indicates more appetite for these internal assessments.

SCOPING ENGAGEMENTS

The single most important part of any penetration test is to nail down the scope of the engagement with the customer; this discussion happens while negotiating the overall statement of work (SOW) and covers the kinds of assets and data the client wants to see tested, as well as how much time the penetration tester has to devote to assessing and hacking the organization's assets.

Internal versus External

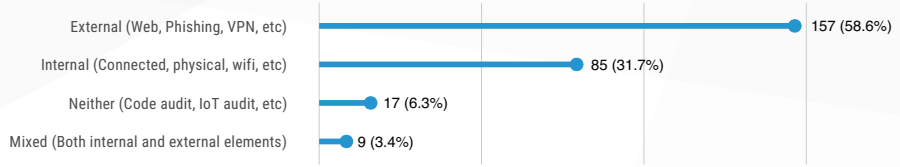
The most important distinction in scoping the engagement is that of an "external" versus an "internal" penetration test. External engagements focus on web-based attacks against the target organization's website or customer-facing web applications, email-based phishing campaigns (both targeted and broad), and credential collection efforts via externally-facing endpoints like VPNs and cloud-hosted assets. In contrast, internal assessments tend to focus on the internal local area network (LAN) and wireless LAN (WLAN), and the systems that are not (intentionally) exposed to the internet: payroll systems, factory floor equipment, internal source code repositories, and the like.

Over the course of our study, we found, unsurprisingly, that client organizations far preferred an externally-based penetration test, as illustrated in Figure 1. Sixty-two percent of all Rapid7 engagements include an external component.

This finding comports with our notion of the traditional penetration test. Client organizations tend to be primarily concerned with external threats attacking their enterprise from the outside, over the internet. After all, when we think of a breach, we tend to think of that externally-based attacker, who is likely sitting in a far-off, extradition-proof region of the world.

Figure 1: Engagement Types

Aggregation is across all engagements (n = 268)



Source: Rapid7

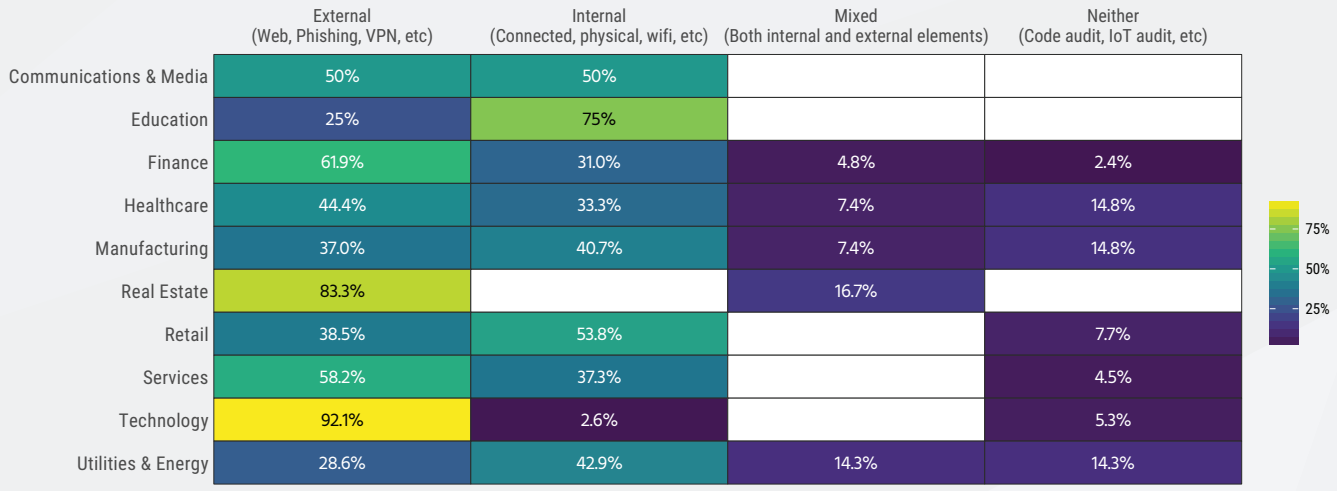
However, we do see that organizations are also taking the insider threat a little more seriously this year; these are engagements where the penetration tester is on-site or near-site and is expected to connect to the internal network (LAN) or wireless network (WLAN) to conduct attacks. In our 2016 study, we found that only 21% of engagements were purely internally focused, whereas this year's nearly 32% figure indicates more appetite for these internal assessments.

This uptick in internal assessments is an indicator that organizations are, in general, taking a more holistic approach to their network security and are more likely to assess both their internal and external attack surfaces. We expect these organizations are better equipped at defending the internal network in the event of either a rogue insider or an external attacker masquerading as an insider with commensurate insider-level privileges.

We can drill down into which industries are more likely to be concerned with internal versus external threats by looking at Figure 2.

Figure 2: Pentest engagement scope (by industry)

Percentage value is number of engagements in industry. Industries with low number of representations removed.



Source: Rapid7

Within our sample data, we can see that communications and media, healthcare, manufacturing, retail, and services organizations tend to employ more defense-in-depth strategies with a healthy balance of internal and external assessments, while the real estate and technology industries are more focused on external threats, and education institutions are more focused on internal threats.

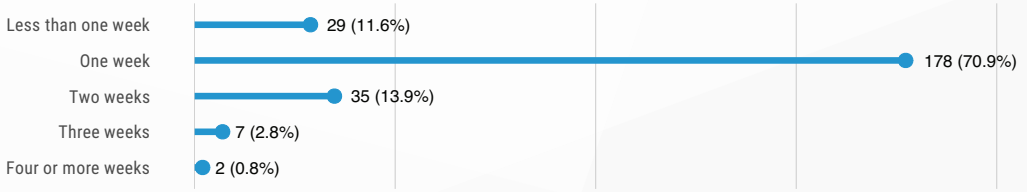
The Time Box

One of the criticisms of penetration testing as a security practice is the notion of fixed start and end times. After all, "real" criminals and other threat actors aren't explicitly bound to a particular set of calendar dates to perform reconnaissance and launch attacks. In contrast, penetration testing is almost always performed as an agreed block of billable hours, much like any other professional service. It's true that many reported corporate breaches are later recognized as the result of a persistent attacker who maintained a network presence for weeks or months before detection, and this is a model of attack that is frankly ill-suited to the time-boxed penetration testing model.

That said, the vast majority of individual "attacks" that internet-connected organizations are expected to weather are discrete, untargeted events; most malicious activity is performed by automated scanning and reconnaissance, tuned to look for a small set of pre-defined vulnerabilities and misconfigurations of which to take advantage. In light of this, an engagement time scope of a week or 10 days is much longer than the seconds to minutes that a typical untargeted, opportunistic incident is measured in.

Figure 3: Pentest engagement times

Aggregation is across all engagments where engagement length was recorded (n = 251)



Source: Rapid7

Figure 3 shows typical engagement duration for a penetration test.

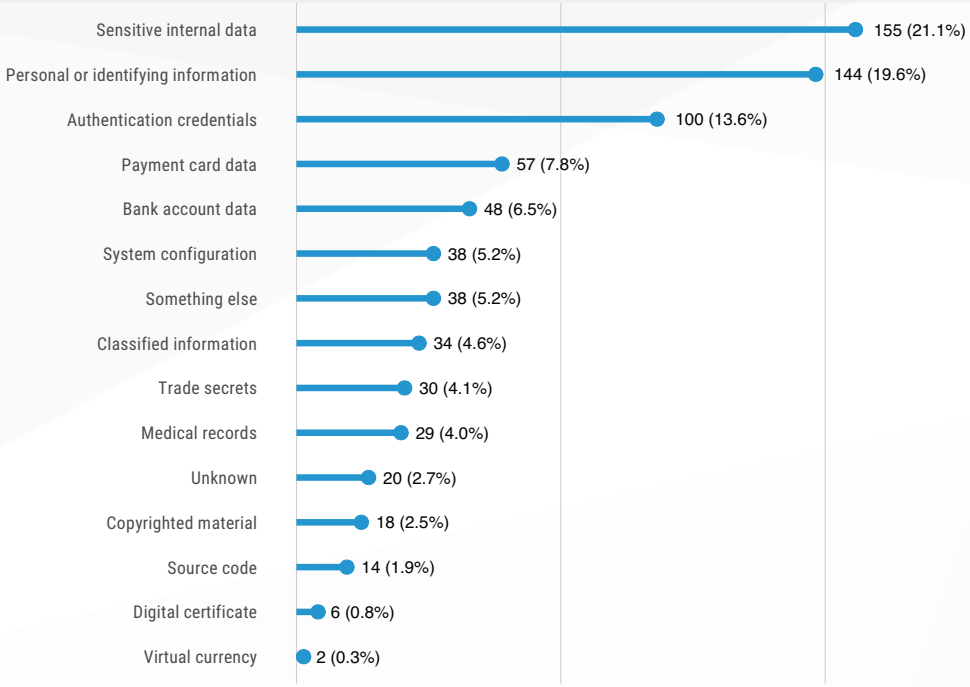
We can see that client organizations far prefer a one-week engagement over any other time scope; 71% of all engagements land on this one-week scale, with all other engagement times falling in popularity. While longer engagements do tend to uncover more issues, most critical top-level findings can be discovered and documented by a skilled pentester inside a week, and the expected return for more protracted engagements falls off pretty significantly.

What's at Stake?

The final factor in scoping a penetration engagement is the characterization of what the client organization is interested in protecting. Rarely are penetration tests structured in such a way that the pentesters are told to just go wild and see what they can see—organizations nearly always have some outcome in mind. Are they worried about protecting the personal information of their clients and customers, or are they more interested in how they protect their own intellectual property? Figure 4 illustrates what was reported as most important for the surveyed organizations.

Figure 4: Data types of interest for security validation

Aggregation is across all engagments (n = 268)



We can see that organizations are primarily interested in securing their own sensitive data—this tends to be data such as internal communications, financial metrics, and other business-critical internal products. This category is followed closely by the personal or identifying information of customers and employees, and specifically, the authentication credentials of the organization's network users.

While trade secrets and classified information showed up in the mid-to-lower of the priorities list, this is likely because a relatively low number of organizations have information they categorize this way. However, for those that do, protecting their secret recipe of X-number of herbs and spices is a top priority. While the general emphasis on protecting various categories of internal data suggests that cyber-espionage is a persistent concern among CISOs and other security practitioners, it also appears that the more fundamental priorities eclipse worries about unfair and illegal competitive practices. This makes sense—after all, in order to protect the highest value data, it's critical to ensure that an organization's overall security posture is reasonably healthy. It doesn't make a lot of sense to pour resources into protecting a specific internal source code repository if the state of the network it's connected to is untested.

THIS ONE TIME ON A PENTEST:
PDLC VIGILANCE

ENGAGEMENT TYPE:
IoT Product Audit

VERTICAL:
Utilities & Energy

INVESTIGATOR:
Jesse Gardner

With the constant barrage of consumer electronics finding their way onto our networks and into our lives, one often wonders about how all these exciting innovations come to fruition. This is a short tale of IoT product development life cycle (PDLC), where a seemingly unrelated lapse in holistic security resulted in mass pwnage, and underscores the importance of periodic security reviews across all systems supporting these products.

If you haven't heard of PDLC before, it is akin to the software development lifecycle (SDLC), which many tech savvy people understand pretty well, but unlike SDLC, PDLC includes the hardware platform. Simply put, PDLC is the process of developing a product and is part of product life-cycle management (PLM). It also includes hardware and is concerned with the complete development of a product. This process may, and often does, involve numerous software components that may or may not be connected to the cloud.

Usually, the scope of an IoT/embedded device product audit is very specific to small components or functionality of a device. Occasionally, the scope is wide open and our customer wants us to go after anything and everything we might find that is related to the target device. This engagement was the latter, and being that it was a "black box" approach, we weren't given any information other than, essentially, "we want you to target this application that runs on the device." We were not provided any domain names, internet endpoints, or other intelligence relevant to the application.

Because the focus target was a cloud-connected application running on the device, we began by reviewing network activity emanating from the device. This was done using Wireshark and port mirroring of the target device's Ethernet connection. The target application was executed on the device, and we watched to see which hosts it reached out to on the internet (DNS lookups, API connections to the cloud, etc.). Based on this analysis, we discovered some fully qualified domain names (FQDNs) that we could use to seed our subsequent information gathering efforts.

By searching for content on those FQDNs, and by using DNS open source intelligence (OSINT) tools like dnsdumpster.com, we were able to identify several, likely related targets that shared similar names to the DNS entries used by product under review. As we scanned new targets with nmap and other reconniscience tools to learn more about the services they might offer, we were off to the races.

We enumerated services across targets, and lo and behold, we discovered a lonely web URL path that was accessible without authentication. This was odd, because everything else we threw at that target returned an HTTP "403 Forbidden" response. This available URL was discovered through the use of dirsearch, a directory and file bruteforce discovery tool.

With the interesting URL in hand, we fired up Burp Suite, loaded the webpage in our browser, and inspected what happened next. Upon accessing the page, it was immediately apparent that this was a development test site that was used to perform automated unit testing for a test instance of the application we were targeting <insert drool emoji>.

As the automation scripts fired up, our browser asked if we wanted to remember the username and password. Now, keep in mind that we didn't provide any user credentials, but as it turns out, there were credentials embedded in the test script. Yes! Pulling those embedded credentials from our proxy logs was easy, and now we had an administrative user account on the application's test environment.

The natural next step was to test those credentials against the main production application, and they worked like a charm. This granted fully fledged administrative access to the backend production web application.

The customer was thankful for this discovery and activated their incident response process to ensure we were the only ones who had tasted that particular forbidden fruit.

The takeaway from this tale is that PDLC is multifaceted and requires vigilance across many systems, including pre-production QA and development systems. When figuring scope for an embedded device audit, don't be too stingy with scope in assessment activities—safeguard development systems and include them in security review/auditing. Insecure password storage and reuse is definitely a thing, and controls and guidance around those secondary systems are a must-have for a complete assessment. ◆

This edition of Under the Hoodie saw a significant increase in the rate that software vulnerabilities are exploited in order to gain control over a critical networked resource.

VULNERABILITIES EXPLOITED

Software vulnerabilities can be thought of as **unintentional, undocumented functionality**, and exploits are special-purpose programs designed to take advantage of that functionality. Unfortunately, as code and systems get more complex and more interconnected, the likelihood of introducing vulnerabilities in a networked environment increases, more or less to the point of inevitability.

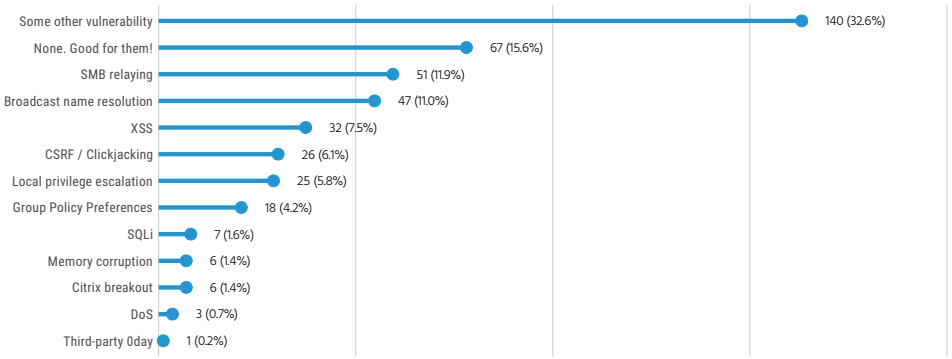
In practice, when vulnerabilities are exploited, they allow the attacker to bypass some kind of security control. The effects can range from information leaks (the attacker learning a secret that they shouldn't be able to learn) through privilege escalation (the attacker gaining special rights otherwise unavailable), all the way to arbitrary code execution (the attacker running code of their choice on the target computer, completely subverting the vulnerable software).

Software vulnerabilities are at the core of penetration testing, and this edition of Under the Hoodie saw a significant increase in the rate that software vulnerabilities are exploited in order to gain control over a critical networked resource. Our last report saw about a 68% rate of vulnerability exploitation. Using a glass-half-full point of view, 32% of sites surveyed for the current edition of this report were free of exploitable vulnerabilities.

This edition's survey tells a different story, as seen by Figure 5.

Figure 5: Vulnerabilities encountered during engagement

Aggregation is across all engagements (n = 268)



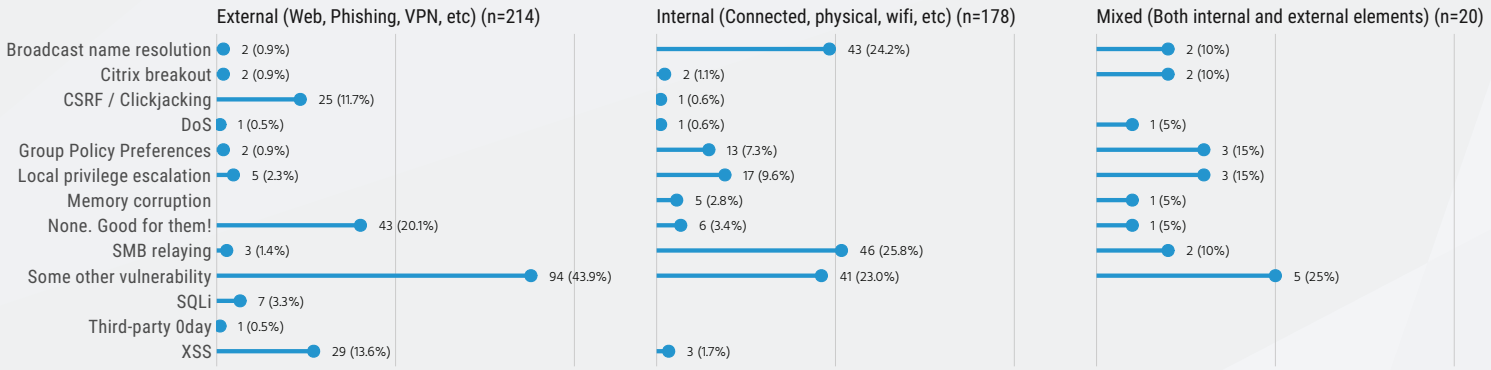
Source: Rapid7

Among the 2017–2018 sample period, the environments where software vulnerabilities were encountered grew significantly; during the survey period, only 16% of sites tested did not suffer vulnerability exploitation at the hands of penetration testers. Unfortunately for defenders, 84% of surveyed sites were compromised to some degree through vulnerability exploitation. We delve deeper into these results in Figure 6 by breaking down these figures among external, internal, and mixed engagements.

84% of surveyed sites were compromised to some degree through vulnerability exploitation

Figure 6: Vulnerabilities by engagement scope

Counts and percentages are reflections of aggregations by scope



Source: Rapid7

Relying entirely on an automated solution or a short list of canned exploits is likely to meet with failure, while a more thorough, hands-on approach nets significant wins for the attacker.

Among the 178 internal engagements surveyed, less than 4% were software-vulnerability-free; in other words, at least one vulnerability was exploited in over 96% of internal penetration tests.

Since we know that most engagements are externally based, we would expect to see a preponderance of cross-site scripting (XSS), cross-site request forgery (CSRF), and SQL injection (SQLi) vulnerabilities, since these are most commonly associated with externally-facing web applications. However, this is not the case—penetration testers reported encountering "some other vulnerability" more than 32% of the time, usually (56%) in combination with at least one of the other more specific vulnerability categories.

Penetration testers (and real adversaries) do rely on the homogenic character of corporate networks, using tried-and-true commodity exploits that are effective nearly everywhere. The most obvious example of this is SMB Relaying, which is exploited almost 26% of the time on internal engagements and 12% overall. This vulnerability is common to Microsoft Windows networks—which is to say, common to nearly all corporate networks—and exploiting this vulnerability involves the attacker impersonating an SMB server in an environment where SMB traffic is unsigned. Combined with Broadcast name resolution, which is also common on such networks (and exploited about 24% of the time on internal engagements), attackers can exploit these issues in most cases to gain site-wide control of any given Microsoft-based network. (For a detailed walkthrough of SMB relay, see Rapid7's Leon Johnson's seven-minute video at <www.rapid7.com/resources/smb-relay-attacks-explained/>.)

However, these results also imply rather conclusively that there is often an additional, custom component to penetrating a client's network, especially in external engagements where SMB networking vulnerabilities may not be available to the attacker. The practice of penetration testing appears to involve significant situational awareness and leveraging multiple vulnerabilities after the first one is uncovered and exploited. Relying **entirely** on an automated solution or a short list of canned exploits is likely to meet with failure, while a more thorough, hands-on approach nets significant wins for the attacker.

Another interesting data point is the absolute rarity of third-party zero-day vulnerabilities encountered and exploited on site. In the one reported case of zero-day use, it was in combination with XSS and "some other vulnerability," so even in this case, the zero-day was incidental to the engagement's success. The data here supports the intuition that competent attackers do not need newly discovered, unreported vulnerabilities in order to successfully compromise a site. Known third-party vulnerabilities, sometimes in combination with site-specific vulnerabilities, is often enough to gain significant control over a network.

THIS ONE TIME ON A PENTEST:
SCRAPING MEMORY WITH HEARTBLEED

ENGAGEMENT TYPE:
External Assessment

VERTICAL:
Utilities & Energy

INVESTIGATOR:
Trevor O'Donnal

I was part of a team of several consultants tasked with testing the security of a company's public-facing infrastructure to determine if there were any vulnerabilities that might lead to theft of company data. The scope we were given included several servers running web applications for various inter-company business processes. We found only the standard web protocols exposed to the internet, along with a few SSH interfaces here and there. It didn't take us long to discover that the organization we were testing used an internal single sign-on (SSO) service that allowed a single set of credentials to be used on most of the applications. We knew that if we could get our hands on some valid credentials, we would be able to get to the target data. Our first barrage of attacks involved testing the firewall for weaknesses and attempting to coax SMB connections from the internal network, using specially crafted emails to employees of the company. Those tactics failed (good for the client!), so we were finally left with "the hard way"—we needed to hack our way through the exposed web applications.

We began the task of testing the security of each and every web server. Not long into the process, we identified three servers that hadn't had their OpenSSL installations updated in a while; they were consequently vulnerable to the "Heartbleed" bug of 2014. We divided up the three servers amongst the team and began harvesting memory contents from the three servers in 65 KB chunks. We set up a looping process to continually request chunks of memory and append them to the ever-expanding file of leaked secrets. We let our automated memory grabbers run overnight, and by the next morning, we awoke to find a wealth of information.

The memory files we had generated contained cleartext usernames and passwords for users that logged into the servers in question. In addition, we were able to confirm that one of the servers also acted as a secure messaging server which handled sensitive emails. As a result, our memory dump file contained full message bodies of hundreds of (otherwise) secure emails!

To press our advantage, we parsed the memory dumps for all the user accounts we could find and ended up with just over a dozen sets of credentials. Unfortunately for us, none of the user accounts we harvested had VPN access to the internal network. However, they DID have access to almost every public-facing web application, as we had suspected in the beginning. We started digging, armed with our newly captured credentials. In the end, we gained access to the company's HR application, which yielded a large amount of sensitive employee information, including corporate credit card numbers, passport data, W-2s (which include employee social security numbers), and health insurance enrollment information with dependents' names, birth dates, and addresses.

This engagement helped the client to understand that an attacker doesn't always have to get access to the internal network to extract critical data and cause tremendous damage, and illustrated the need to keep up on patch management across all networked assets and the libraries that underpin software running on those assets. ◆

Some kind of network or service misconfiguration is encountered on an internal penetration test over 96% of the time.

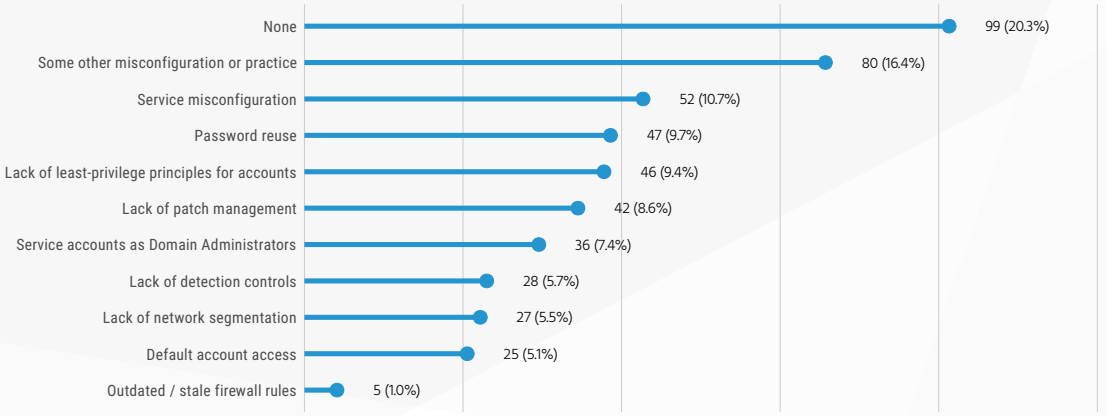
MISCONFIGURATIONS ENCOUNTERED

Distinct from software vulnerabilities are network misconfigurations. These are issues that, while not baked into the software itself, tend to arise from implementation errors on the part of the targeted organization's IT staff. While penetration testers were able to exploit software vulnerabilities about 84% of the time, there was a slightly lower rate of leveraging misconfigurations (about 80% of the time), as shown in Figure 7.

After "none" and "other," the most prevalent named misconfiguration is a "service misconfiguration." These tend to be network services either in default configurations, which are inappropriate for the network, or are configured in such a way that some shipping security feature is disabled. For example, if a cryptography service allows for a fallback to a weak, easily cracked encryption algorithm, then that would be a misconfiguration; it's likely intended functionality, but it's also not appropriate in terms of modern security standards.

Figure 7: Misconfigurations leveraged per engagement

Aggregation is across all engagements (n = 268)



Source: Rapid7

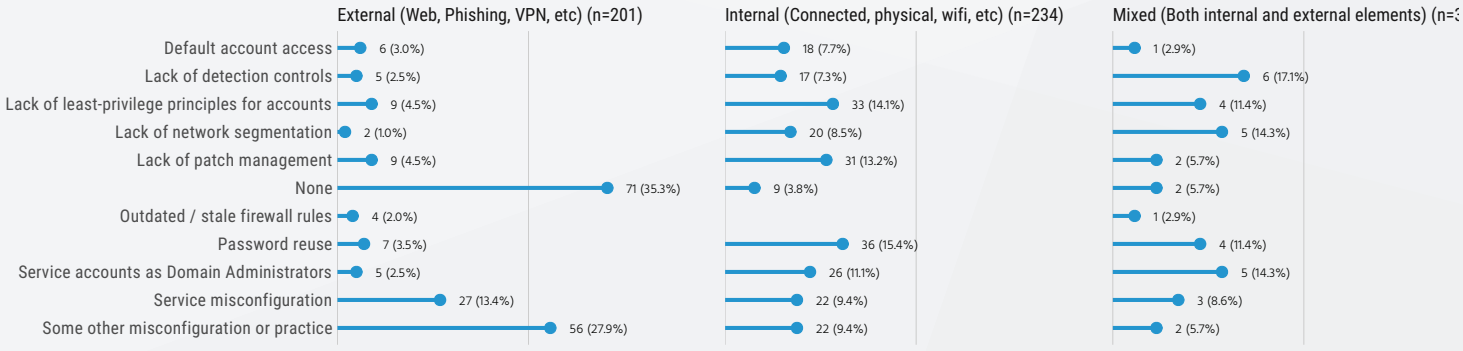
Once again, we should take these figures in the context of internal, external, and mixed engagements in order to better understand what kinds of misconfigurations are likely to be encountered on a given penetration test.

As seen in Figure 8, some kind of network or service misconfiguration is encountered on an internal penetration test over 96% of the time, which is coincidentally exactly the same rate as vulnerability exploitation. As with vulnerability exploitation, penetration testers rarely exercise exactly one misconfiguration at a site. A lack of least-privilege principles, in combination with password reuse and service accounts running with unfettered Domain Administrator privileges, all point to a habitual use of too-powerful accounts to perform normal business functions. Penetration testers and criminal attackers alike will tend to focus their efforts on these service accounts once they are discovered.

Similarly, a lack of network segmentation and a lack of detection controls will tend to mean that once an attacker has compromised one computer on one network, then that attacker has a straight shot to nearly any other computer on site, regardless of each of those computers' functions.

Figure 8: Misconfigurations leveraged by engagement scope

Counts and percentages are reflections of aggregations by scope



Source: Rapid7

THIS ONE TIME ON A PENTEST:
YOU MISSED A SPOT

ENGAGEMENT TYPE:
Internal Network Assessment

VERTICAL:
Utilities & Energy

INVESTIGATOR:
Ted Raffle

Penetration testing is a valuable tool for clients to identify and remediate vulnerabilities and misconfigurations. On one engagement, I was able to demonstrate how strong security controls and threat mitigation can miss the mark, even when only one or two systems fall through the cracks.

Like many internal network penetration tests, I started by looking for NetBIOS Name Service (NBT-NS) and Link Local Multicast Name Resolution (LLMNR) requests to exploit. These auxiliary name resolution services can be used to resolve hostnames that are not found in DNS, which vulnerable hosts transmit out to the local network in broadcast and multicast requests, respectively. By sending poisoned responses that claim the desired host name, a malicious actor can do things such as receiving NetNTLMv2 password hashes from SMB connections.

Testing across two different local networks, I identified six hosts making NBT-NS requests. Poisoning attacks resulted in Rapid7 receiving SMB connections and password hashes for only one domain user.

One of the most immediate risks of obtaining these password hashes is that they could be cracked. During this assessment, Rapid7 was unable to crack the one NetNTLMv2 password hash that was obtained.

Another risk of being able to solicit these SMB connections is SMB relay, where that challenge-response authentication traffic can be replayed against other hosts without the password needing to be cracked. Using RunFinger.py, I found three SMB hosts that did not require message signing, only one of which was a Windows server joined to the domain.

Leveraging SMB connections from the one user account that was affected by NBT-NS poisoning, and using Smbrelayx to relay that connection to the one domain host that did not require SMB message signing, Raffle was quickly able to retrieve password hashes for local user accounts from that host.

Unlike NetNTLMv2, NTLM password hashes are not challenge-response pairs, and can be used for "pass-the-hash" authentication; the uncracked password hash can itself be used as a password for network-based authentication. I was able to use these same local administrator credentials on nearly half of the SMB hosts that were in scope for the assessment. Using these credentials, I retrieved cached plaintext passwords and password hashes with Mimikatz. Mimikatz is an application that pulls cached credentials from the memory of a Windows computer, including domain logins, so I was able to recover a cached NTLM password hash for one of the domain administrators.

After using domain administrator access to retrieve NTLM password hashes for all of the domain users, I still had very limited success cracking passwords—valid credentials were only discovered for three working user accounts. While these accounts were enough to find sensitive documents in file shares and SharePoint, the password cracking results showed that the client was very serious about password requirements. When discussing these results during an on-site debrief, the client explained that most of their users have smart cards and long, randomly generated passwords. Despite these strong passwords and very limited findings on most hosts, though, this engagement showed that a few missed systems can still allow a malicious actor to gain a foothold using common attacks. ◆

CREDENTIALS CAPTURED

We've discussed the vulnerabilities exploited and misconfigurations leveraged by penetration testers, and while these are certainly the more glamorous, Hollywood-hacker tactics employed, most penetration testers looking for a quick win will target an organization's user credentials. After all, even if you have a perfectly configured network and your patch management processes are bulletproof, you still need some mechanism to allow your users to actually use the resources you've provided for them. Credential-gathering attacks can either be the object of exploiting a vulnerability or leveraging a misconfigured network service, but these attacks don't necessarily rely on mistakes made by software developers or IT administrators. Instead, it can be performed through easier techniques, such as phishing and social engineering campaigns.

In nearly all cases, a credential is a username and a password. While some organizations assign passwords to their users, most organizations today allow users to choose their own passwords, provided they fall within some minimum set of standards of length and complexity. Unfortunately, this strategy continues to be the source of much angst and gnashing of teeth among security professionals and pundits.

For example, while an organization may allow users to pick their passwords, they will tend to enforce certain complexity rules in order to encourage "good" passwords. A common list of requirements might be, "at least eight characters, including at least one upper case, one lower case, one number, and one special character." As a result, a very common password that too-clever humans pick is "Summer2018!" (with the exclamation point). It fits the proscribed pattern, but because it's so easy to type, remember, and change every 90 days, it is one of the worst passwords a person can choose.

Figure 9 describes the success rate of penetration testers against a site's credentialing procedures.

Figure 9: Credential capture success rate across all engagements

Obtaining credentials was not an engagement goal for all engagements (n=251)

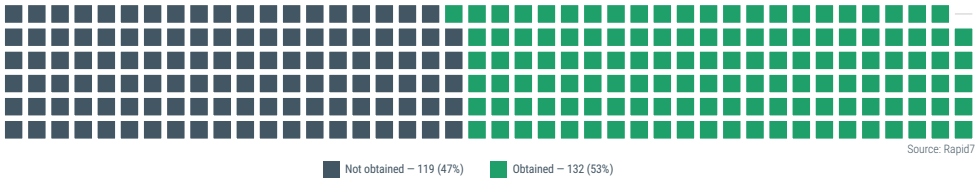
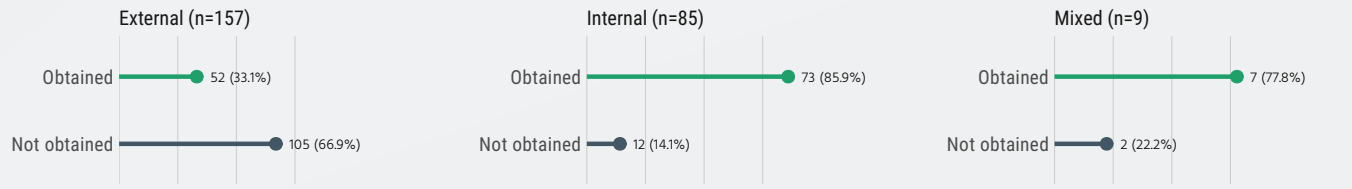


Figure 10: Credential capture success rates by engagement scope



Source: Rapid7

Across all engagements where the target organization's networked assets were part of the scope, penetration testers were able to successfully compromise credentials 53% of the time, making it slightly more likely than not that an attacker could impersonate at least one authorized user on the network. If the penetration tester is on the local network as part of an internal or mixed assessment, the success rate for credential compromise is even more stark:

Given LAN or WLAN connectivity, penetration testers were able to capture credentials 86% of the time; this is the one security control every single person in the organization should be the most aware of, and yet this is where penetration testers thrive.

Perhaps more alarming, though, is the 33% success rate among external engagements. Often, these engagements don't even have network credential testing as part of the stated scope of the engagement, so this figure is a little underreported in our survey, and yet, our data shows that credentials fall out in the findings by accident about a third of the time.

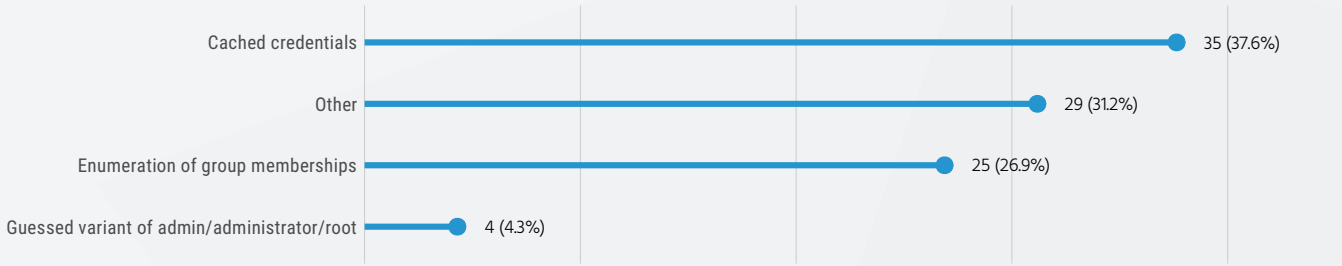
Identifying Privileged Credentials

Regardless of the mechanisms of defining discrete levels of privilege, accounts typically fall into the category of "user-level" accounts and "privileged" accounts. The former are used by people (and rarely, services) who don't need or want enhanced privileges on a given computer; they can't install system-altering software by themselves, delete logs, or otherwise affect the overall security of the host operating system. The latter, in contrast, are used by people (and often, services) who do need these rights; they're typically administrator accounts, or otherwise have unusual levels of control over the authenticating system. Figure 11 illustrates the methods that penetration testers use to discover, and then target, privileged accounts.

This chart illustrates the most common mechanism of elevating one's privileges from a mere user account to a privileged account: scraping memory for cached credentials, usually on Windows-based workstations, and usually with the techniques employed by the venerable security assessment tool Mimikatz, which is open source and available at <github.com/gentilkiwi/mimikatz>. While these techniques used to be the exclusive purview of penetration testers and manual, hands-on criminal attackers, the WannaCry event of February 2016 catapulted these techniques into the mainstream for modern ransomworms. In many Windows networks, if a domain account (including service accounts) has logged in to a workstation, that password hash will be stored in memory, available to users who have at least local administrator privileges. This includes domain accounts with domain passwords; therefore, in cases where local users have local administrator access (either intentionally or accidentally), it is often trivial to escalate privileges to domain administrator.

Figure 11: Ways privileged accounts were identified/compromised

Not all engagements requested credential harvesting (n = 93)



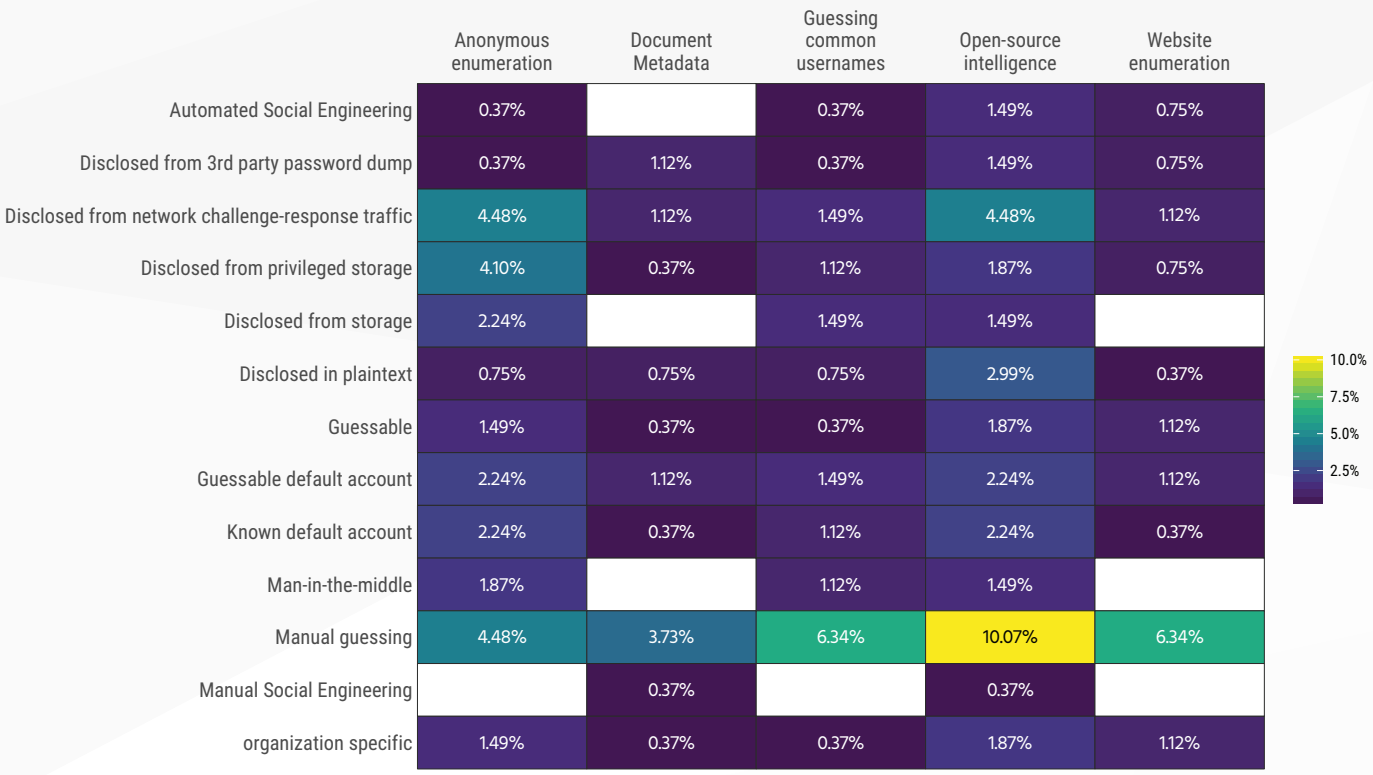
Source: Rapid7

Methods of Capturing Credentials

Regardless of the trickery described above, the most common mechanism of actually obtaining credentials is painfully low tech, as illustrated in Figure 12.

Figure 12: How usernames and passwords were captured

Percentage value is relative to unique combinations of unique username and password exploit methods



Source: Rapid7

Regardless of the method of guessing usernames (which are rarely secret, or only lightly obfuscated), the most common mechanism employed by penetration testers is "manual guessing," which is exactly like it sounds: a professional human password-guesser manually selecting passwords to try against a list of usernames.

DEEP DIVE

INVESTIGATOR:
Patrick Laverty

VERTICAL:
All

PASSWORD COMPLEXITY IN THE WILD

Pentesters are always trying to get access. Even the lowest level of access can be enough to get a foothold in a network. One of the best ways to get access is with a password. So how do pentesters get passwords? One way is through guessing, but the more effective credential capture strategies don't involve just random guessing. Through experience, we know typical passwords that people choose, and we also know the general format.

At Rapid7, we have a set of about 130,000 passwords and their corresponding hashes, pulled from the domain controllers of several client sites where we gained site-wide administrator access. No usernames are saved with the passwords, nor is any other client-specific information. These passwords are transmitted and stored securely, and access to this infrastructure is limited only to select Rapid7 employees from Rapid7-controlled endpoints. The data is relatively fresh, with the oldest only a few months old, so most of these passwords were selected by users in 2017 or 2018. Duplicate passwords are also kept in order to learn how often certain passwords and password patterns are employed between disparate organizations. The value of this dataset is in the fact that these are real passwords that real people use at their real jobs; we expect that very few of these accounts represent "throwaway" accounts as one might find in many of the public datasets that come from website breaches. It follows, then, that we should expect these VPN and workstation passwords to be reasonably complex and "secure." But are they? In short: not very.

Three Most Common Password Patterns

In this dataset, there are three extremely common passwords. The first is one that most people would guess, and that is "password." To be fair, there are many variations of "password." We see **Password1**, **Password123**, **Password2**, **Password1!**, and many others. If we are looking at specific passwords that follow this popular pattern, then Password1 is the most common. But due to the many variations, we can add them all up and see that these variations on "password" and some minor decorations are the most common password pattern, with 4,001 entries out of 129,812, or just about 3%.

The next password pattern may not be as obvious, but when you think about the thought process of the user, it makes sense. As mention above, the most common company password policy requires that people change their password every 90 days, which is about every three months. The other thing that changes every three months? The season. Many people have "invented" a system where they have a password that is easy to remember and never repeats by simply choosing the current season and appending the year. When looking at examples like **Winter2018**, **Summer2017!** and **Spring16!** We count a total of 1,788 passwords, or 1.4% of the total set.

The third password pattern isn't a specific word, but it is the most common approach in the list: the organization's name. When guessing passwords, one of the first patterns penetration testers will try are variations of the company's name. We found a total of 6,332 instances of passwords that included the target company's name, which works out to just under 5% of the total set. The base of these passwords includes the company name, but then the variations on it are similar to what we saw with "password." Examples include **Company123!**, **Company1**, **C0mp@ny1**, and **Company2018**. So, while "password" is the most common password pattern base across our data set, decorating the organization's name as a password is the most common strategy employed.

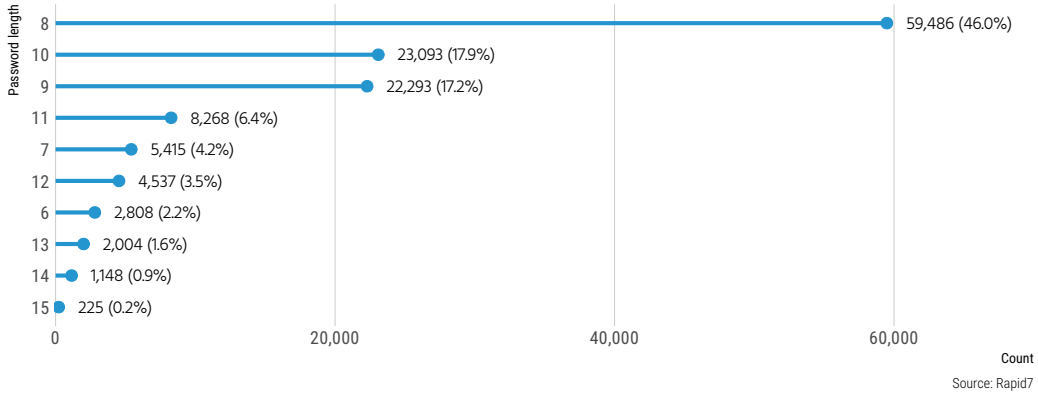
These percentages may not seem large, but keep in mind that a malicious actor might only need a single set of working credentials to gain access a network. If you have 100 users, then there's a good chance that five will contain the company's name, three will be based on the word "password," and one or two will be the current season and year. Multiply these percentages out to the number of users a company has, and it increases the likelihood of a correct password guess in the absence of site-wide, username-agnostic rate-limiting.

Numerical Patterns Emerge

During penetration tests, we often find that the target organization's minimum password length is set to eight characters. With this in mind, let's look at the aggregated password lengths from our dataset. One thing that sticks out is that people often stick to exactly this minimum length. If we check the length of passwords in our datasets, the top ten are:

Figure 13: Distribution of password lengths

Only the most common password lengths are shown. Percentages calculated based on full data set.

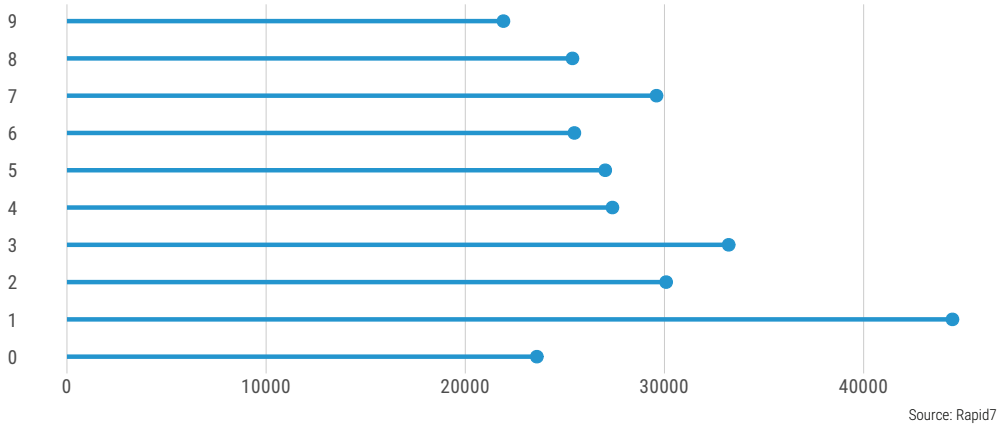


It turns out, eight characters is not only the most common, but it actually is more common than the next four combined! This information can help someone trying to perform password guessing or password cracking simply by avoiding passwords that are nine or ten characters long, or pretty much any other password length.

We can also look into the character patterns used in our two datasets. We are able to generate patterns and figure out which characters are used in each position. This can show where someone prefers to place the uppercase letter, the digit, the special character, and the lowercase letters. People tend to end their password with a digit. Eight of the top ten password patterns end with a digit. Knowing this, we will want to do most of our password guessing with at least one digit at the end. But which digit?

Figure 14: Distribution of final digit frequency in passwords

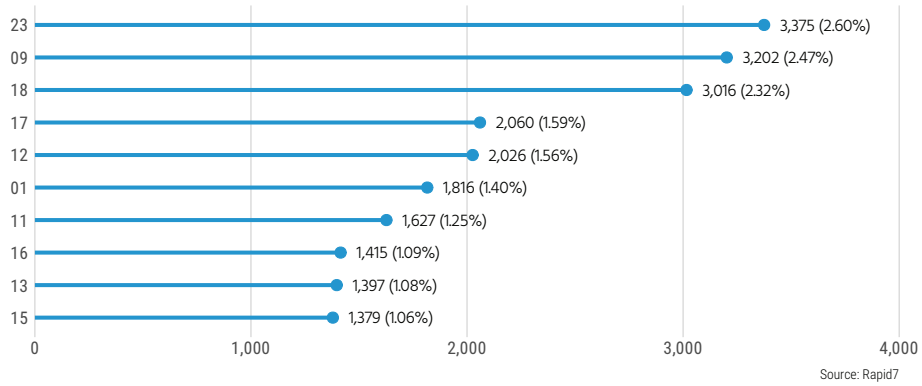
Data derived from potfile on Password Cracking Server.



When required to use a digit, most people will simply stick a "1" on the end, and our data proves this out. We saw an example of this earlier with Password1 and Company1. What about if there is more than one digit at the end, what do they use?

Figure 15: Most common trailing two digits in passwords

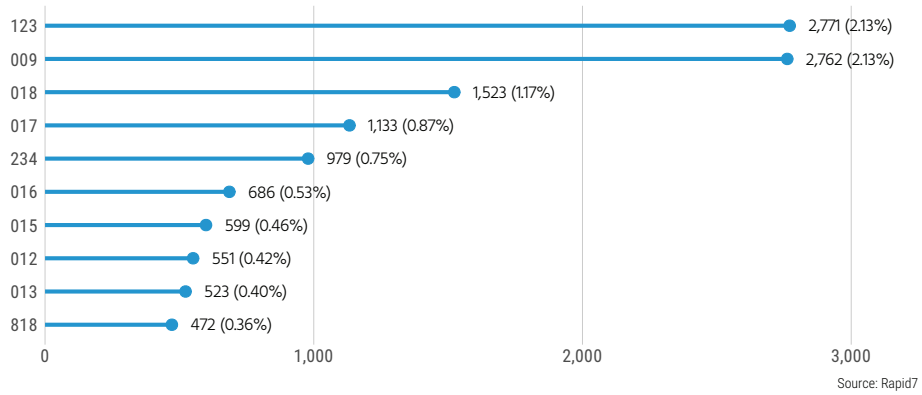
Only the most common password endings are shown. Percentages calculated based on full data set.



The top result of "23" may point to another pattern of human choices. Let's look at the most common last three digits:

Figure 16: Most common trailing three digits in passwords

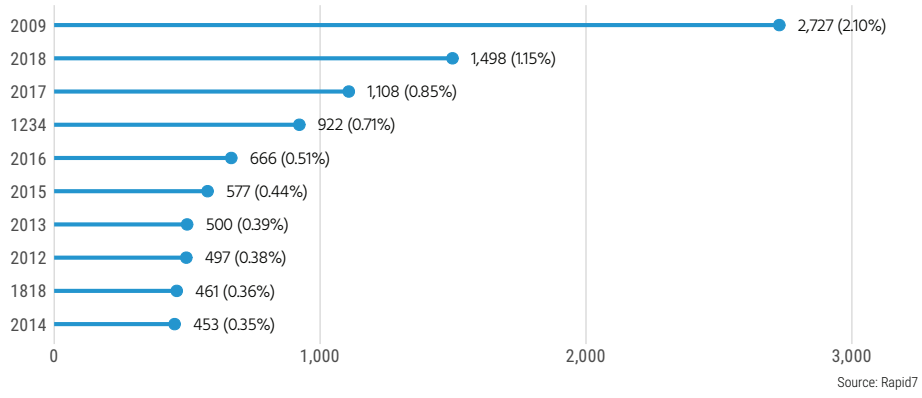
Only the most common password endings are shown. Percentages calculated based on full data set.



We do see that people are using a very memorable pattern of digits at the end, 123. Maybe this is to pad length on a short, five-character password, or maybe this is to not just have a single digit, thinking that three digits is (somehow) better. So what does it look like if we have four digits at the end? Maybe surprisingly, a new candidate emerges.

Figure 17: Most common trailing four digits in passwords

Only the most common password endings are shown. Percentages calculated based on full data set.



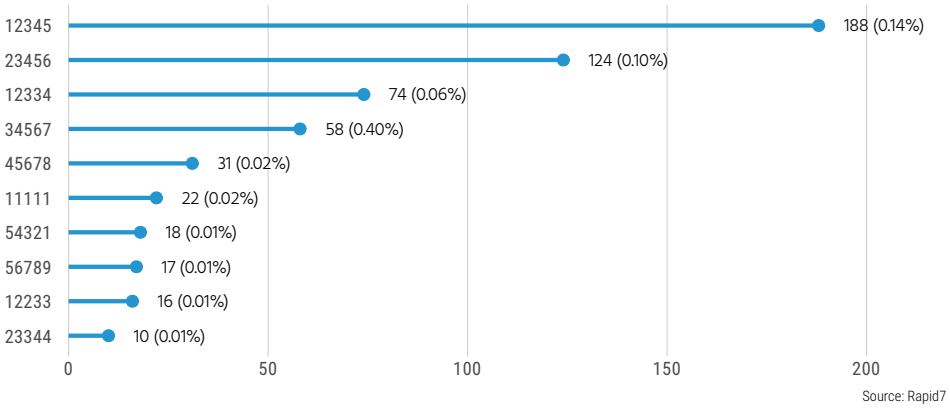
When there are four digits at the end, the most common appear to resemble a year, but 1234 is also represented. In penetration testing, we do often see people putting the year at the end of the password, as in the Summer2018 example. The top result of "2009" remains a bit of a mystery, since most of the year-like patterns indicate more recent years, but this may be an artifact of a single penetration test that netted a large number of legacy accounts which haven't seen a password rotation in several years. Or, these passwords might be connected to multiple employees across many organizations who all happen to have eight- or nine-year-old kids. It's hard to know for sure, given the data anonymization we perform before adding passwords to the set.

If we take this just one step further and look at the last five digits, our original pattern returns with sequential digits, starting with "12345."

When there are four digits at the end, the most common appear to resemble a year, but 1234 is also represented.

Figure 18: Most common trailing five digits in passwords

Only the most common password endings are shown. Percentages calculated based on full data set.



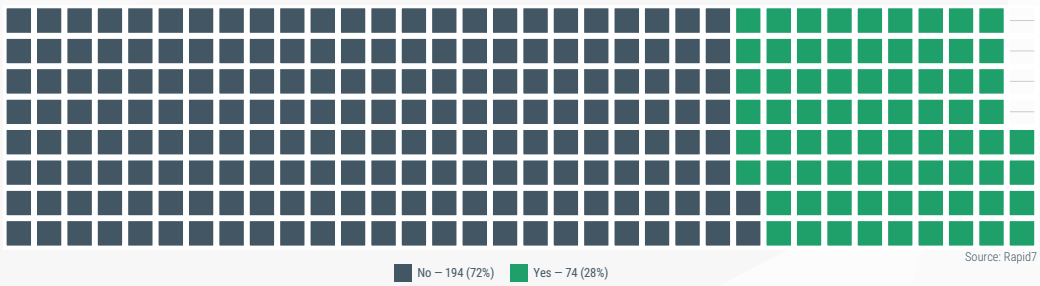
In summary, the data collected and presented here shows that humans are predictable when they create their own passwords. People have a lot of passwords to remember and may have heard that password reuse is a bad idea, so instead, they use a password pattern that is memorable for the service and choose the name of the company as their base word. We also see patterns where numbers are most commonly used at the end of a password, and particular digits and patterns of digits that stand out as being more common choices for users. ◆

Extending Privilege

Once a valid credential is captured or a critical service is compromised through vulnerability exploitation or configuration abuse, the usual goal of a penetration tester is to gain site-wide administrative control, almost always via a Domain Administrator or Enterprise Administrator credential. Figure 19 illustrates how often this prized goal is achieved on engagement.

Figure 19: Site-wide administrative control success rate (overall)

Aggregated across all engagements (n=268)

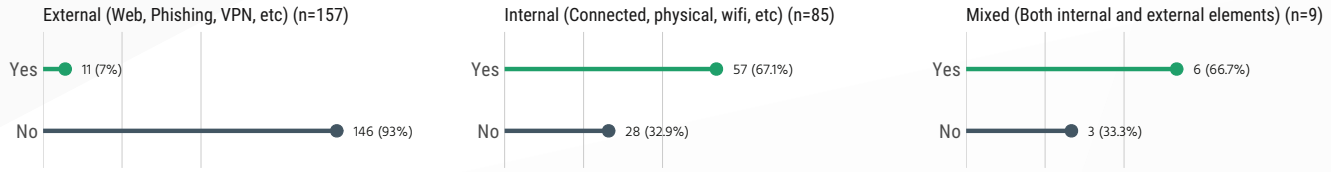


About 28% of all engagements result in site-wide administrative control of the target organization, but we can also break down this success rate between external penetration tests and engagements that have an internal component:

Given the significantly higher chances of encountering a vulnerability, misconfiguration, or weak credential when LAN or WLAN access is obtained, it should come as no surprise that pentesters on an internal engagement were able to gain site-wide administrative control 67% of the time.

Figure 20: Site-wide administrative control success rate (by scope)

Counts and percentages are reflections of aggregations by scope



THIS ONE TIME ON A PENTEST:
THE PERILS OF PASSWORD REUSE

ENGAGEMENT TYPE:

External Assessment (Web Application)

VERTICAL:

Education

INVESTIGATOR:

Steven Laura

As penetration testers, we are continually talking to our clients about the importance of a strong password policy for the organization, as well as the use of two-factor authentication (2FA). While we feel that we constantly talk about this issue, in many cases the recommendations are overlooked or set aside. However, it is pretty safe to say that one of the most common reasons we are able to access systems and networks to reach sensitive information is because of weak credentials resulting from weak password policies.

One of the most important aspects of penetration testing is the initial stage of information gathering. We use various techniques during this stage, one of which is looking for leaked credentials for a client organization posted online—perhaps in public password dumps.

In a recent web application penetration test that I performed, a number of email addresses and passwords were found in a public password dump. In the case of this web application, the login required a user's email address and their password for access. With the list of usernames and passwords, I kicked off my login attempts, and soon found that one of the credential sets listed in the password dump worked on the corporate network. This is likely because the user was reusing passwords between sites and had not changed their corporate password after the breach of the other site.

With access to the account, I was able to download a large amount of customer PII data, as well as financial information from the affected user's email inbox. I discovered that this user recently left the company, so access to other areas with their account to additional customer PII data was removed. Yet, their email inbox was still fully accessible. In addition, with access to the account, further information was then gathered, including a list of internal users, the password policy for the web application, and more. In addition, there was the risk of a malicious actor utilizing this abandoned email address as a "trusted" account for phishing against other users both internally and externally. ◆

Furthermore, these results imply that if the penetration tester is not detected within a day, it's unlikely the malicious activity will be detected at all.

DETECTION AND DEFENSE

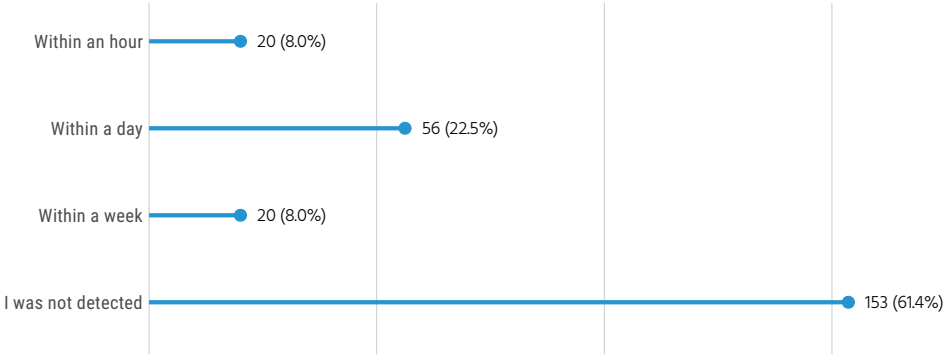
As stated above, penetration testers are nearly always limited by a relatively short engagement window and are rarely afforded more than two weeks to assess and compromise a given target. This time-box necessarily limits the opportunity for stealthy attacks, which tend to take much longer than the typical Metasploit run. Low-and-slow scans are nearly impossible to detect, and careful planning and re-planning during a phishing, social engineering, or even traditional hacking campaign can mean the difference between domain admin and getting caught in the act.

Despite these caveats, Rapid7 penetration testers remained undetected on 61% of engagements, as seen in Figure 21.

Despite these caveats, Rapid7 penetration testers remained undetected on 61% of engagements

Figure 21: Detection rates (overall)

Detection rate percentage for engagements where detection evasion was part of scope (n=249)

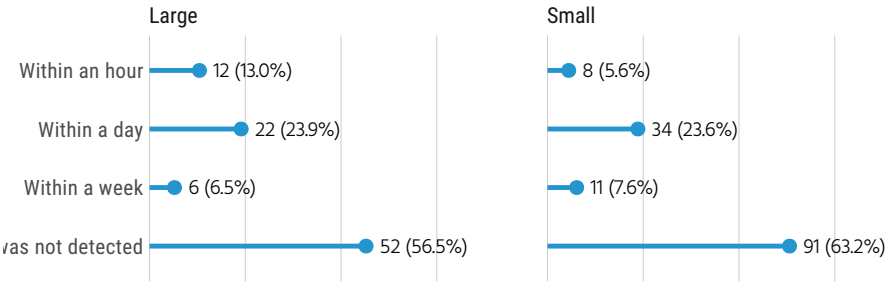


Source: Rapid7

Furthermore, these results imply that if the penetration tester is not detected within a day, it's unlikely the malicious activity will be detected at all. We also took a look at detection rates in large organizations versus small, as shown in Figure 22.

Figure 22: Detection rates split by organization size

Detection rate percentage for engagements where detection evasion was part of scope (n=249)



Source: Rapid7

Here, we see that the penetration tester remained undetected somewhat more often in small enterprises of less than 1,000 employees (63% of the time) versus large enterprises (at 57% of the time). The difference here is worth discussing. Large enterprises have more assets to protect, and larger, more complex networks tend to provide more opportunity for finding unpatched vulnerabilities, misconfigurations, and weak credentials to take advantage of. However, large enterprise also will tend to have more existing detection capabilities thanks to larger IT budgets and a more experienced staff. This detection capability is further suggested by the fact that when the attacker is detected, that alarm is sounded and acted upon in the first hour somewhat more often in the larger enterprises in our survey.

However, the overall detection rate delta is not that large, at only about 6% better performance on the part of the target organization if it is in the "large" category. Both large and small organizations would do well to review their detection capabilities before their next penetration test.

Credential Management

As we can see from this study, credential capture is often the least complex method to ultimately compromise a network. At first glance, it would seem that it is also the easiest security control that an IT team in an organization can shore up with some detection and defense strategies. After all, it is effectively impossible to guarantee that modern, complex software and networks are going to be 100% free from vulnerabilities or misconfigurations, but it should be comparably easy to institute some basic controls for managing user accounts.

Alas, as far as we have observed in the field, this is not the case.

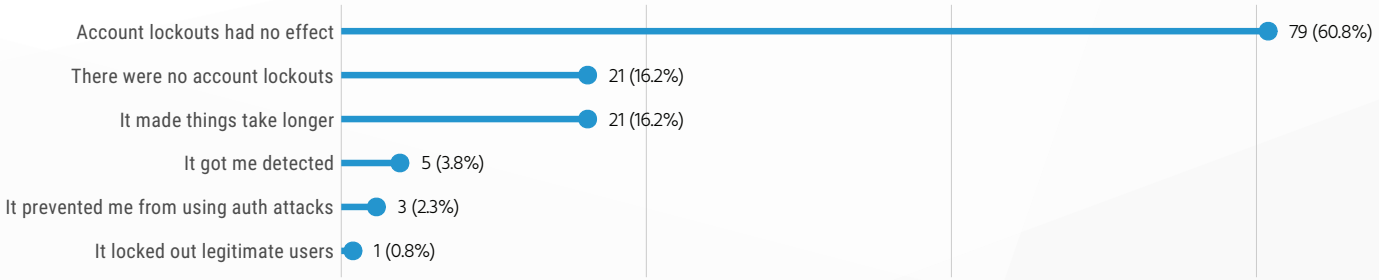
Account Lockouts

Locking out an account for some amount of time (often forever) after some number of failed password attempts (often five), is among the oldest automated techniques for securing credential access in the face of an active attacker.

Lockouts are the boogeyman of all junior penetration testers. Triggering a lockout can not only get a penetration tester detected, but it can cause a personal denial-of-service condition for the legitimate user of that account and make for an unhappy client. However, the legendary lockout is relatively rare in the field, as illustrated in Figure 23.

Figure 23: Account lockouts during engagement

Aggregation is across all engagements that tested for lockouts (n = 130)

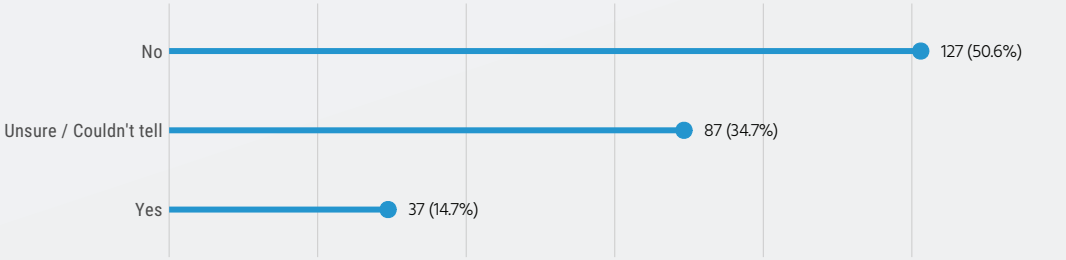


Source: Rapid7

A common tactic to capture credentials is "bruteforcing," or automatically attempting many, many passwords per account in a more-or-less blind way. Among those tests where bruteforcing credentials was a permitted, in-scope action, it was a serious factor only 7% of the time when the lockout strategy did, in fact, cause the penetration tester to get detected, abandon authentication attacks entirely, or cause a service outage. Although lockouts do tend to slow down credential capture (16% of the time), it was either ineffective or absent on 77% of the engagements where lockouts were specifically being tested.

Figure 24: Two-factor authentication across engagements

Not all engagements requested credential harvesting or compromising 2FA (n = 251)



Source: Rapid7

Two-Factor Authentication

A more recent credential control is two-factor authentication (2FA), also sometimes known as multi-factor authentication (MFA). After a password is correctly entered, the user is challenged for an additional value. This is usually a short series of algorithmically generated numbers based on a shared secret between the 2FA device and the authenticating computer. Some organizations deploy this solution by either issuing employees with a purpose-built 2FA device, or by leveraging the smartphones employees use, which may be personally owned devices.

While 2FA continues to grow in popularity, it is still rare to find it in the field, as seen in Figure 24.

2FA was present and effective on only 15% of all engagements, with the remaining 85% of engagements unperturbed by this defensive strategy.

2FA was present and effective on only 15% of all engagements, with the remaining 85% of engagements unperturbed by this defensive strategy.

Vulnerability and Misconfiguration Management

As stated above, it is practically inevitable that an experienced penetration tester will uncover at least one vulnerability or misconfiguration and use it to their advantage. However, this should not cause IT, security, and development teams to lose heart; there are strategies available to help minimize the impact of a breach, both simulated by a penetration tester or caused by a real threat actor.

The number one issue that causes the most consternation among penetration testers is solid **network segmentation**. If they cannot traverse logical boundaries between environments, it can be extremely difficult to leverage a set of ill-gotten workstation credentials to escalate to domain-wide administrative privileges; even if a powerful service account has been compromised, if there's no route between targets, the pentester must effectively start over again with another foothold in the network.

Speaking of those powerful service accounts, a principle of least-privilege can help contain the damage suffered by losing control of that service account. IT administrators should review the actual permission requirements for service accounts and devise a non-root, non-administrator account permission scheme that allows the service just enough privilege to perform its intended function. If a particular product vendor insists that their software must have Domain Administrator credentials, it's worth a conversation with that vendor to discuss which user permissions are minimally required. For Windows environments, Microsoft Active Directory provides the "Protected Users" group, which should be investigated to protect specifically against caching domain credentials on local workstations. More on this strategy can be found in the excellent article by Jim Shaver at <jimshaver.net/2016/02/14/defending-against-mimikatz/>.

It should go without saying that a robust vulnerability and patch management solution should be employed at every organization. Many end-user systems today are configured by default to check for and apply software patches automatically, but some organizations are reluctant to employ the same strategy to business-critical servers. While a patching routine may not necessarily be technically automatic for these systems, it's imperative that IT and security organizations work together to ensure that patches are rolled out as quickly and seamlessly as practicable. Just as shipping vulnerabilities is an inevitable consequence of general purpose computing, patch and vulnerability management should be treated as equally inevitable and routine procedures.

Socializing Security

Last but certainly not least, a culture of "see something, say something" should be embraced in any organization. Implementation mistakes and security trade-offs are the reality in today's fast-paced, ever-changing networked world. This is not just the responsibility of IT, security, and development staff, but for every user on the network. Training end users to spot phishing campaigns, social engineering operations, and other relatively low-tech attack techniques goes a long way to extending the security team's vision and reach. When a problem is spotted and reported, the security team should be on point to not only address and mitigate the issue, but to publicly and positively acknowledge the issue once it's been mitigated.

Ultimately, if people are encouraged to report issues like this through positive feedback, the dark, unmonitored corners of the network will shrink.

THIS ONE TIME ON A PENTEST:
GIVING IT ALL AWAY

ENGAGEMENT TYPE:
Red Team Engagement

VERTICAL:
Manufacturing

INVESTIGATORS:
Kirk Hayes and Nick Sanzotta

When it comes to a red team engagement, most customers have a hardened exterior, and this customer was no exception. On a recent red team engagement, my colleague Nick Sanzotta and I were tasked with a black box point of view, where no information was given except for a company name. The customer challenged us to gain internal access and obtain sensitive intellectual property (IP), while they would see if they were able to detect and respond to a targeted attack. As with any red team exercise, only the point of contact knew about the operation, where the defending blue team did not, so the organization's detection and response procedures would be put to the test.

As always, open source intelligence (OSINT) was the first stop, where we found a hardened exterior complete with multi-factor authentication (MFA) requirements on most logon interfaces.

Using an open MobileIron webpage, we started in with manual password guessing and accessed a number of user accounts. But with MFA in place, we had no way to gain internal access with the accounts. We instead discovered that access to the customer's Exchange Web Services (EWS) server was not protected by MFA. I used MailSniper to connect to mailboxes, download the Global Address List (GAL), and access emails. I also modified MailSniper to send email as users to perform internal phishing attempts.

After these attempts were unsuccessful, Nick called the customer Help Desk posing as a user who was on vacation and needed access to the VPN. Nick stated he was at a funeral, but needed access to OWA to respond to an important email. Nick spoke with the help desk employee and gathered information which could be used in further calls. A while later, after the first help desk personnel was off work, Nick called back, used the same pretext along with the fresh information gathered, and asked for assistance. The help desk employee was so helpful he gave Nick an account to use that would bypass the MFA entirely and gain access to the VPN.

Once VPN access was achieved, we moved quickly to discover sensitive data by accessing email accounts with the earlier guessed passwords. In one of those email accounts, I discovered a help desk ticket that included the machine name and IP address for a user who had local administrative access granted on their laptop. I used this access to compromise the user's workstation and establish a Command and Control (C2) channel. I also logged in to a Citrix server with user credentials obtained earlier in the engagement, escaped the Citrix application sandbox due to lax configuration standards, and established a more stable C2 channel due to the system being a high availability system.

We used these two systems to elevate privileges in the domain and exfiltrate ten gigabytes of data, including sensitive documentation on upcoming products. At no point were Nick or I detected. ◆

APPENDIX A: METHODOLOGY

This report's primary data source is a post-engagement survey answered by Rapid7 penetration testers from September of 2017 through mid-June of 2018. There were 268 total responses, but as with any survey, some engagement data points were not captured completely. The complete survey is reproduced in Appendix B.

Target Demographics

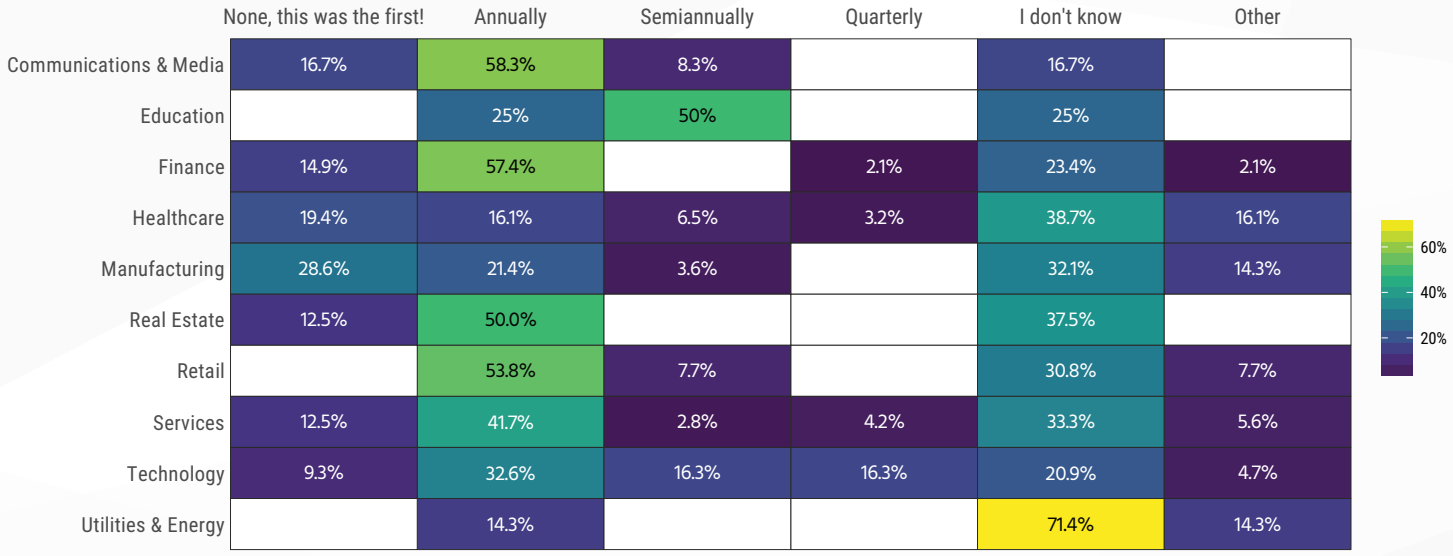
In order to parse these results, it's important to get a clear handle on what kinds of companies and organizations were tested by Rapid7. Of course, the details of these organizations must remain confidential, but we can share broad, anonymized statistics based on both the industries these client companies are involved in as well as the size of the tested organizations.

Target Industries

Figure 25 describes the industries that are represented by five or more organizations in this sample set , as well as how often the companies in those industries were tested.

Figure 25: Penetration testing frequency by industry

Percentage value is number of engagements in industry

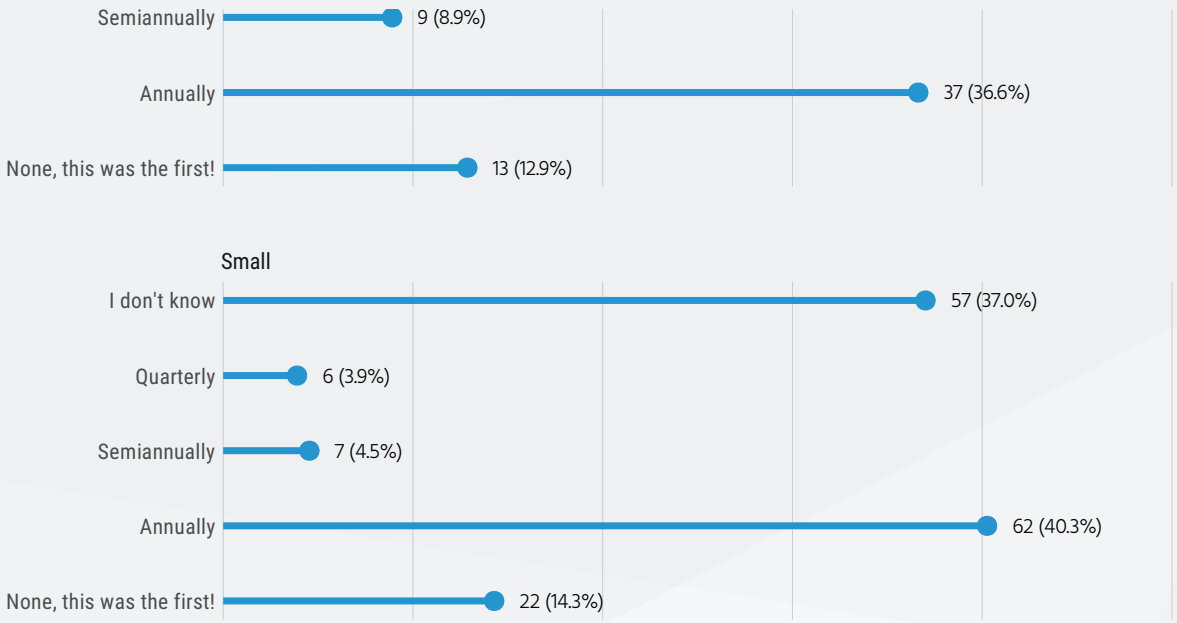


Source: Rapid7

We can see from this data that most companies favor an annual penetration testing routine, regardless of industry.

Figure 26: Penetration testing frequency by organization size

Percentage value is number of engagements in census org size group. Note free scale.



Source: Rapid7

Target Sizes

Another way to cut the targeted organizations is by size. That chart is presented in Figure 26.

In total, there were 101 large companies, which have 1,000 or more employees, versus 154 small companies identified, so our sample is skewed toward those smaller organizations which have fewer than 1,000 employees. Again, both sizes of organizations tend to have penetration tests on an annual cycle, as opposed to quarterly, semiannually, or none until now.

APPENDIX B: PENETRATION TESTING EXIT SURVEY

1. **Was the penetration test internal or external? * (Mark only one oval.)**
   - Internal (Connected, physical, wifi, etc)
   - External (Web, Phishing, VPN, etc)
   - Mixed (Both internal and external elements)
   - Neither (Code audit, IoT audit, etc)
2. **How often is the client penetration tested? (Mark only one oval.)**
   - I don't know
   - None, this was the first!
   - Annually
   - Semiannually
   - Quarterly
3. **How many pentests has the client had, including this one? (Mark only one oval.)**
   - I don't know
   - None, this was the first!
   - Two
   - Three
   - Four
   - Five or more
4. **How long was the engagement? (These should be calendar days, not person days. Mark only one oval.)**
   - One day or less
   - One week / 40 hours
   - Two weeks / 80 hours
   - Three weeks / 120 hours
   - Four weeks / 160 hours
   - More than four weeks
5. **How quickly were you detected? (Mark only one oval.)**
   - I was not detected. I am a ghost. A ninja. A ghost ninja.
   - Within an hour of starting
   - Within a day
   - Within a week
   - More than a week
6. **What was the client interested in protecting?**
   - Check all that apply.
   - Bank account data
   - Classified information
   - Copyrighted material
   - Authentication credentials
   - Digital certificate

- Sensitive internal data
- Medical records
- Payment card data
- Personal or identifying information
- Trade secrets
- Source code
- System configuration
- Virtual currency
- Unknown
- Something else

7. **Were you able to obtain credentials? (Mark only one oval.)**
   - Yes
   - No (Skip to question 14)
8. **How did you gather usernames? (Check all that apply.)**
   - Guessing common usernames (jsmith, testuser)
   - Linkedin, GitHub, forums, or other OSINT
   - Anonymous enumeration of domain controller/LDAP, or other services
   - Website enumeration (myBFF/Exchange timing/Password reset page)
   - Document Metadata
   - Some other method
9. **How did you obtain passwords? (Check all that apply.)**
   - Manual guessing (Summer2018!)
   - Guessable default account (admin/admin)
   - Known default account (admin/cisco)
   - Guessable, organization specific (CompanyName2018)
   - Manual Social Engineering (e.g., phone call to tech support)
   - Automated Social Engineering (e.g., phishing with an executable payload)
   - Disclosed in plaintext (source code, accessible FTP sites, etc)
   - Disclosed from storage (passwords.xls)
   - Disclosed from privileged storage (/etc/shadow, pass-the-hash)
   - Disclosed from 3rd party password dump (pastebin)
   - Disclosed from network challenge-response traffic
   - Man-in-the-middle
   - Some other method
10. **How effective were account lockouts? (Check all that apply.)**
    - There were no account lockouts
    - Account lockouts had no effect
    - It made things take longer
    - It got me detected
    - It prevented me from using auth attacks
    - It locked out legitimate users
    - It disrupted services due to service account lockout

11. **Did you compromise privileged accounts? (Mark only one oval.)**
    - Yes
    - No (Skip to question 14)
12. **Did you gain Domain Admin or the equivalent of site-wide root? (Mark only one oval.)**
    - Yes
    - No
13. **How did you find privileged accounts? (Check all that apply.)**
    - Guessed variant of admin/administrator/root
    - Enumeration of group memberships via anon/auth DC/LDAP enumeration
    - Cached credentials
    - Some other method
    - Two-Factor Authentication
14. **Was two-factor enabled in the domain? (Mark only one oval.)**
    - Yes
    - No (Skip to question 17)
    - Unsure / Couldn't tell (Skip to question 17)
15. **Were you able to bypass or compromise 2FA? (Mark only one oval.)**
    - Yes
    - No (Skip to question 17)
16. **How did you compromise 2FA? (Check all that apply.)**
    - Email intercept
    - SMS intercept
    - Visual intercept (shoulder surf)
    - Physical device theft (phone, yubikey, etc)
    - Some other method
17. **What misconfigurations did you leverage? (Check all that apply.)**
    - None
    - Default account access
    - Lack of least-privilege principles for accounts
    - Service accounts as Domain Administrators
    - Lack of network segmentation
    - Outdated / stale firewall rules
    - Lack of patch management
    - Service misconfiguration
    - Lack of detection controls
    - Password reuse
    - Some other misconfiguration or practice

18. **What kinds of vulnerabilities did you encounter? (Check all that apply.)**
    - None. Good for them!
    - SQLi
    - XSS
    - CSRF / Clickjacking
    - DoS
    - Local privilege escalation
    - Memory corruption
    - Third-party 0day
    - Locally site-specific 0day
    - Group Policy Preferences
    - SMB relaying
    - Citrix breakout
    - Broadcast name resolution
    - Some other vulnerability
    - Exploits Used
19. **What exploits did you use? (Check all that apply.)**
    - None
    - Well-known exploits (Metasploit, sqlmap, PoCs from exploit-db, etc)
    - Bespoke code (tool you wrote during this or another engagement)
    - Manual command line exploitation (typing on a bash or cmd prompt)
20. **Were you able to collect any confidential data? (Mark only one oval.)**
    - Yes
    - No (Skip to question 22)
21. **What kinds of data did you acquire? (Check all that apply.)**
    - PII (Personally identifying information)
    - PHI (Personal health information)
22. **IP (Intellectual property or trade secrets)**
    - Financial data
    - PCI (Payment card industry)
    - Other:
23. **Final thoughts**
    - Was there anything about your findings that we missed with these questions that you want to capture?
    - Were there misconfigurations, vulns, or credential types that we didn't cover that we should?
    - Any other feedback about your findings that you want to save here for posterity? (Remember, please leave customer names out of your free-text answer).

ABOUT RAPID7 GLOBAL CONSULTING

Rapid7 powers SecOps not only through technology, but also by giving you access to experts who provide peace of mind, clarity and guidance to your program. Our managed and consulting services extend the reach of your team, while our industry research, reports, and open source tools constantly feed the Insight platform—and you—with new insights.

Our practitioners have extensive experience building and managing security programs, with expertise in vulnerability management, fraud detection, penetration testing, threat intelligence, incident response, red team programs, and more.

Learn more about Rapid7 services at www.rapid7.com/services.

ABOUT RAPID7

Rapid7 powers the practice of SecOps by delivering shared visibility, analytics, and automation that unites security, IT, and DevOps teams. The Rapid7 Insight platform empowers these teams to jointly manage and reduce risk, detect and contain attackers, and analyze and optimize operations. Rapid7 technology, services, and research drive vulnerability management, application security, incident detection and response, and log management for organizations around the globe. To learn more about Rapid7 or get involved in our threat research, visit www.rapid7.com.

QUESTIONS?

Email us at research@rapid7.com