Automated Adversary-in-the-Middle(AiTM)- Attack Environment (IaC)

Jaakko Oja

Introduction and Goals

- **The Problem**: Manual deployment of attack servers is slow, error-prone, and often leaves digital footprints (**OpSec**). Modern Red Team operations require speed and reproducibility.
- **The Solution:** An automated, customized **Infrastructure as Code (IaC)** pipeline that deploys and configures AiTM- attack environment without human intervention. Also ensuring operational persistence via tmux ? systemd and enable rapid “burn and rebuild” capabilities
- **Technologies Used:** Evilginx2, Ansible, Go, Linux, Systemd.

Architecture

- **Explanation:** The Ansible controller is effectively the "engine" of this setup —> the core. It executes the **playbook** and transforms a fresh Linux installation into a fully functional attack platform. All of this construction occurs within a VPS: the Ansible controller, running from a local machine, establishes the connection to cloud services.

Operational Advantages

By defining the infrastructure as code, we achieve:

- Immutable Infrastructure: Every deployment is mathematically identical. No configuration drift.
- Opsec Agility: If and IP is flagged, I can tear down the server and redeploy on a fresh node quickly.
- Persistence: Custom systemd units keep the session alive while allowing interactive operator access.

Figure 1. The image shows the Evilginx2 GUI.

Evilginx2

Evilginx2 is an advanced Adversary-in-the-Middle (AiTM) attack framework used to capture user credentials and session cookies. Its primary feature is the ability to bypass **Multi-Factor Authentication (MFA)** by acting as a proxy between the victim and the legitimate service.

This tool is being used by the offensive side of cybersecurity, whether it is a real adversary or a red teamer.

Phishlets

- These are an essential component of Evilginx2. They are small configuration files required for the software to function. They are used specifically to configure the framework for advanced phishing-style attacks.
- There are numerous repositories including custom phishlets available on Github. For this project, I implemented a customized Outlook phishlet sourced from the “simplerhacking” repository.
- This customized phishlet is highly comprehensive and includes:
    - Proxy Hosts: The code defines an extensive network of subdomains covering not only Microsoft but also GoDaddy and Okta services. This suggests a wide-reaching attack surface.

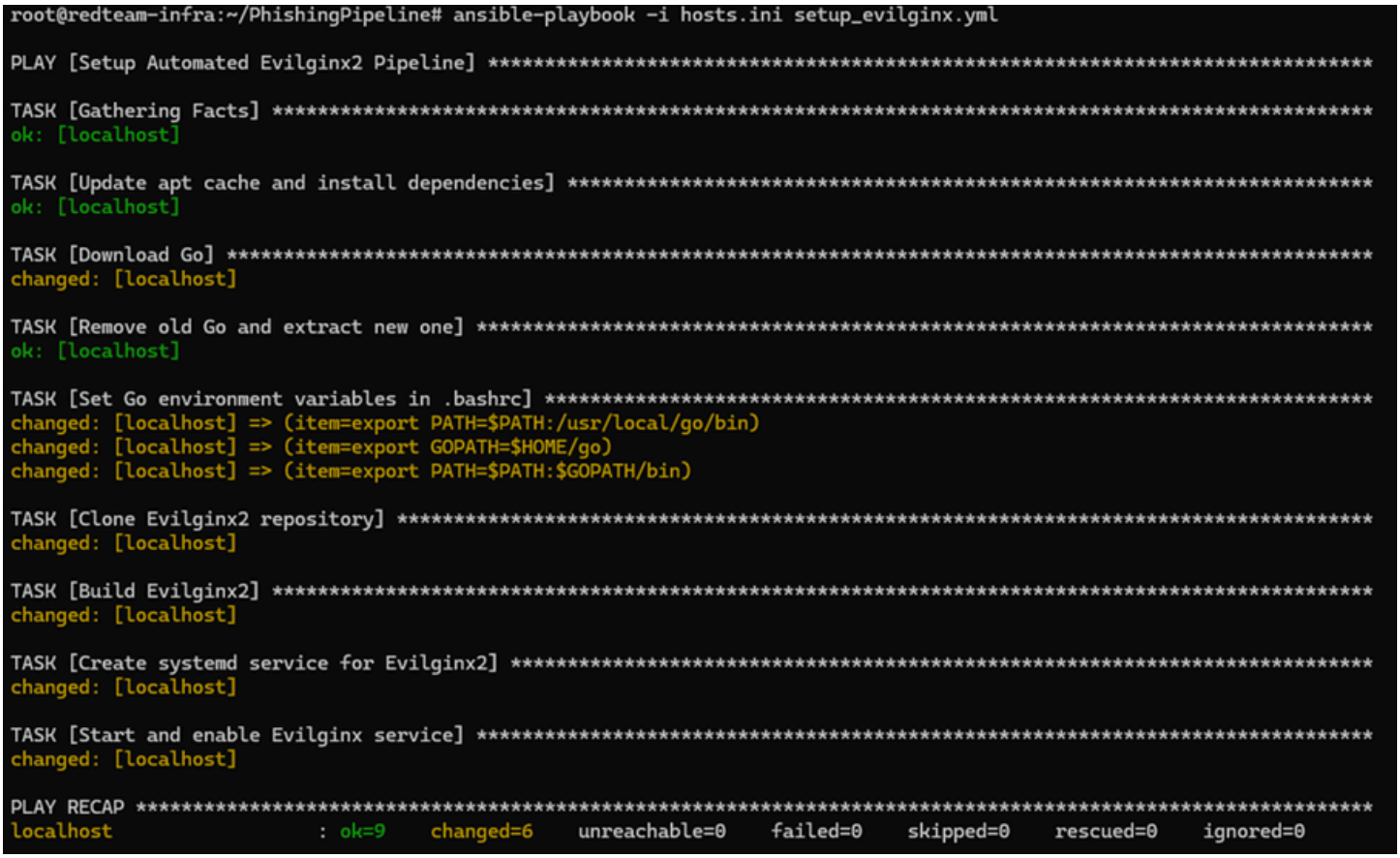


Figure 2. Ansible builds all necessary instances with a single command, providing a rapid solution for system deployment

The Blueprint (Ansible)

The entire environment is defined in a modular Ansible playbook structure. This separates sensitive data (hosts) from the logic (roles).

Repository structure basically is:

RedTeam-Automation/

- hosts.ini // Target Inventory
- setup_evilginx.yml // Master Playbook
- roles/
    - golang-arm64 // Runtime Dependencies
    - evilginx-build // Source Compilation
    - persistence-layer // Systemd & Tmux

Advanced Persistence (Systemd - Tmux Bridge)

Running Evilginx2 as a background service is standard, but Red Teamers need to see the console to capture live tokens (cookies for example). I engineered a solution where Systemd manages a Tmux session.

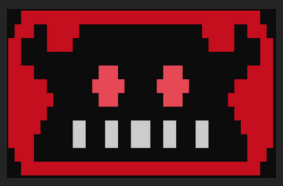
The Mechanism:

- Systemd ensures the service auto-starts on boot (persistence).
- Instead of running the binary directly, it spawns a named tmux session.
- Operators can SSH in and attach to the session (tmux attach -t evilginx) to interact with the console and detach without killing the process.

Tradecraft Acquired:

- Enterprise Automation: Translating manual hacking procedures into idempotent Ansible roles.
- Service Orchestration: Manipulating Linux init systems (systemd) for offensive persistence.
- ARM64 Optimization: Compiling Go-based attack tooling for cost-effective cloud architectures.

Multi-Factor Authentication Bypass (MFA)



Proof of Concept is always fun!

To ensure the automated infrastucture functions as intended, a controlled “live-fire” exercise is performed using a strictly isolated environment.

Test Environment:

- Target: A non-production Microsoft Outlook account created specifically for this validation. This ensures no real user data is at risk.
- Security Controls: The test is conducted within a controlled sandbox.

Validation Methodology: The process follows a four-step “Kill Chain” to confirm the MFA bypass capability:

1. **Lure Generation:** The Evilginx2 generates a unique, disguised URL targeting the Microsoft identity provider (using the custom Outlook phishlet)
2. **Victim Simulation:** The operator logs into the fake Outlook account via the phishing URL, completing the full Multi-Factor Authentication challenge (also approving a mobile app prompt)
3. **Capture Verification:** The system is monitored via the persistent Tmux console to confirm the successful interception of credentials and the session cookie.
4. **Session Replay (The Bypass):** The captured session token is imported into a fresh session on the attackers machine.

Operational Execution & Findings

Following the methodology outlined above, the automated infrastructure was put to the test. The results demonstrated the effectiveness of the deployment pipeline while highlighting the advanced defensive capabilities of modern Identity Providers.

1. Infrastructure Validation The Ansible playbook successfully provisioned the attack server. Upon initialization, the Evilginx2 framework automatically negotiated with Let's Encrypt to generate valid SSL certificates for the phishing domain, effectively bypassing standard CDN proxy restrictions.

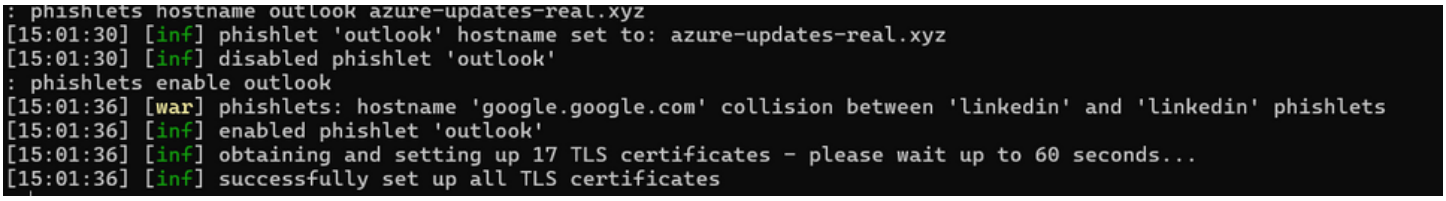


Figure 3. Automated TLS Provisioning.

2. The Lure & Credential Interception The victim simulation was conducted using a disguised Microsoft login portal. The high-fidelity lure successfully deceived the user, presenting a secure (HTTPS) connection that mimicked the legitimate service.

- **Action:** The operator navigated to the lure URL and entered credentials for the test account jaakko.adversary.oja@outlook.com.
- **Result:** The proxy successfully decrypted the traffic and captured the credentials in cleartext.

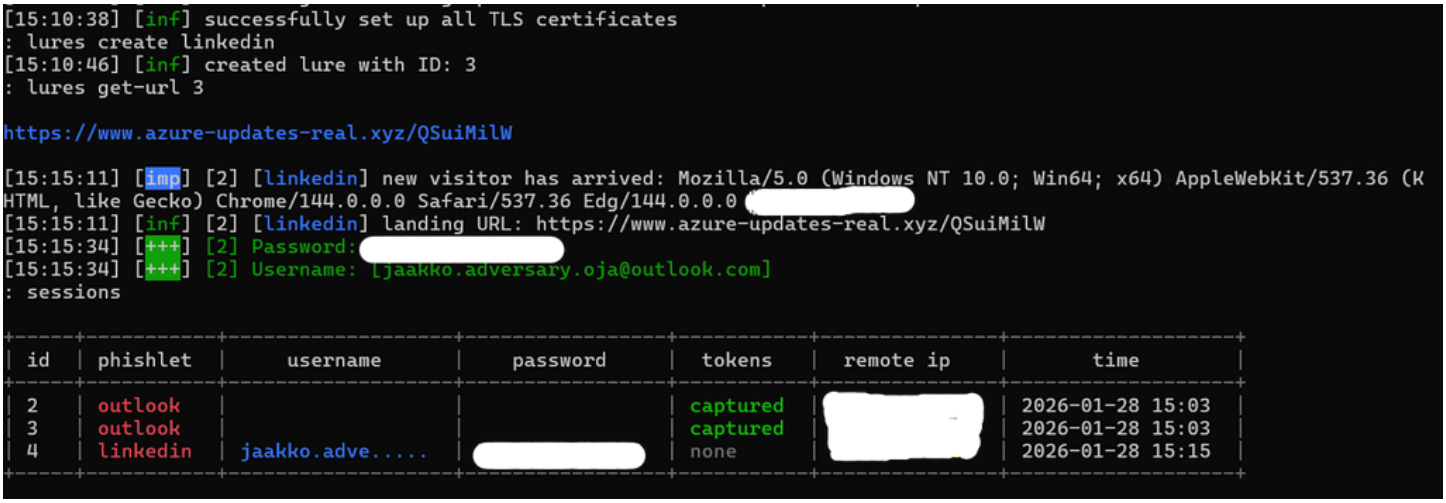


*Figure 4. Successful Credential Interception logs showing captured username and password (F**********6).*

3. Evasion Attempts & Defensive Triggers To simulate advanced tradecraft, the operator utilized browser developer tools to spoof a mobile User-Agent (iPhone), attempting to bypass desktop-specific security challenges and emulate a mobile login flow.

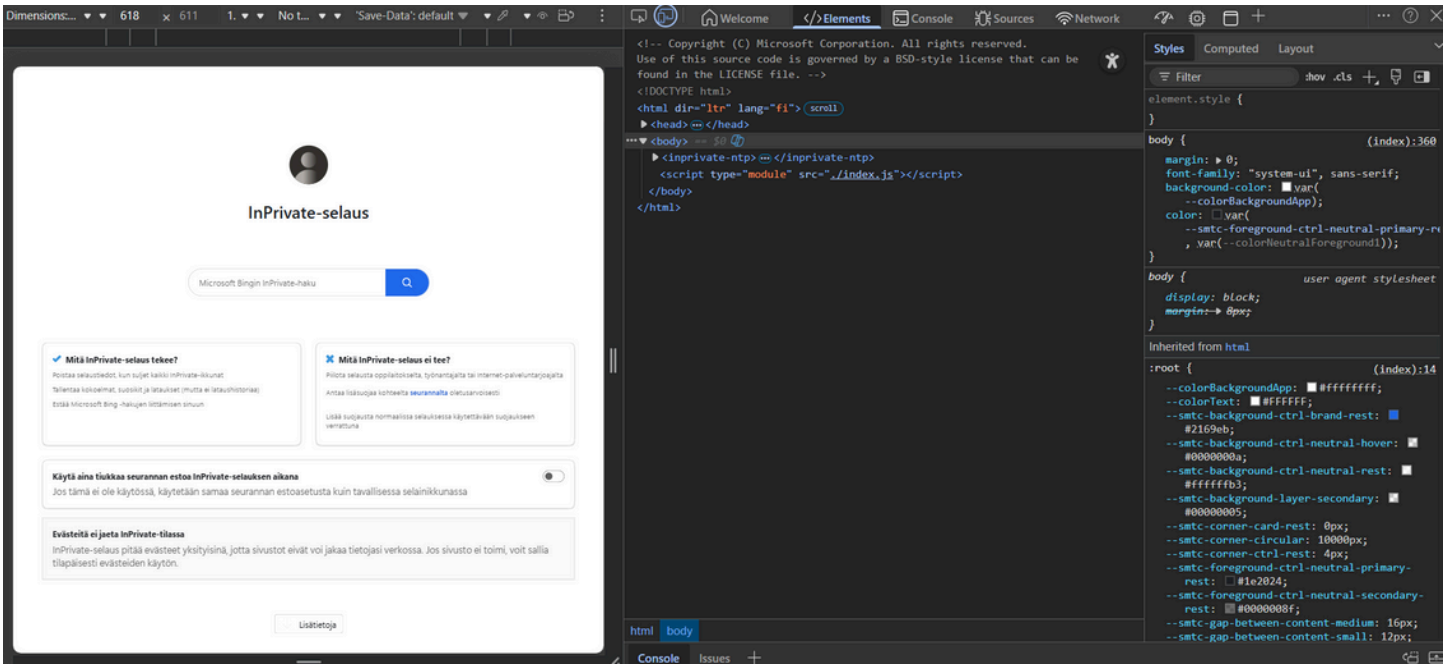


Figure 5. Evasion via Device Emulation.

4. Blue Team Analysis: The "Stop" Mechanism While credential harvesting was successful, the final stage of the "Kill Chain" (Session Token Hijacking) provided valuable insight into defensive mechanisms. The Identity Provider detected anomalies in the traffic fingerprint—likely identifying the proxy behavior despite the mobile spoofing.

- **The Trigger:** The provider enforced a domain-bound **reCAPTCHA** challenge ("Invalid domain for site key").
- **The Impact:** This challenge blocked the final issuance of the authenticated session cookie to the proxy server. Consequently, the "Pass-the-Cookie" attack vector was neutralized by the provider's behavioral analytics.

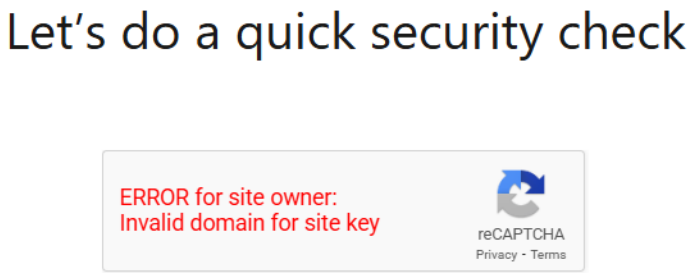


Figure 6. Defense Activation preventing token issuance.

Conclusion on Results

The project achieved its primary goal: *Automating the rapid deployment of offensive infrastructure*. The Ansible pipeline correctly provisioned an operational attack server in minutes. The test highlights that while infrastructure can be automated (IaC), the phishlets

require constant manual development to keep pace with the evolving detection signatures of major platforms.

Key Takeaway: The test highlights a critical distinction between infrastructure automation and evasion automation. While the server deployment was flawless, the failure to bypass the final MFA check demonstrates that static "phishlets" are no longer sufficient against major platforms like Microsoft and LinkedIn. These providers now employ dynamic browser fingerprinting and behavioral analysis that require constant, manual development of custom evasion logic to bypass.

Final Verdict: The infrastructure is operational and successful in harvesting credentials, but the MFA bypass capability requires further development of custom phishlets to overcome modern "Bot Detection" mechanisms.

SOURCES

Evilginx2 Official Wiki: All credits goes to developer Kuba Gretzky.

Outlook phishlet