Project Title: AI-Powered Social Engineering Simulation Framework

---

Short Project Description:

The system identifies human vulnerabilities by creating and launching personalized phishing campaigns, tracking interactions, and generating actionable risk reports.

Component	Role
Target Information Collector	Scrapes public information (OSINT, etc)
Phishing Campaign Generator	Designs realistic phishing emails/messages
NLP Text Personalizer	Customizes messages using AI (GPT, transformers, etc)
Campaign Launcher	Sends phishing attempts ethically
Interaction Logger	Captures clicks, credential submissions
Risk Report Generator	Produces social engineering risk analysis report

---

Component Details:

1. **Target Information Collector**:
    - Gathers LinkedIn, Twitter, public GitHub information, etc.
    - Builds profiles of targets.
2. **Phishing Campaign Generator**:
    - Creates templates for phishing based on target data:
        - Subject lines
        - Offer types (fake promotions, invoices. etc)
        - Etc
3. **NLP Text Personalizer**:
    - Uses AI models (like GPT) to personalize phishing content:
        - Name insertion
        - Company-related references
        - Natural-sounding language
        - Etc
4. **Campaign Launcher**:
    - Sends emails or SMS messages ethically.
    - Tracks delivery and interaction.
5. **Interaction Logger**:
    - Detects:
        - Link clicks
        - Credential entries
        - Attachment downloads
        - Etc
6. **Risk Report Generator**:

    - Summarizes:
        - Success rates
        - High-risk individuals
        - Recommendations
        - Etc

Overall System Flow:

- Input: Organization or target list
- Output: Full social engineering risk assessment
- Focus on **AI-enhanced phishing simulation and user vulnerability detection**.

Internal Functioning of Each Module:

1. Target Information Collector

- **Sources**:
    - LinkedIn scraping (e.g., names, positions, etc)
    - GitHub scraping (email addresses, tech stacks, etc)
    - Public social media data
    - Etc
- **Tools**:
    - Scrapy spiders
    - Google Dorking
    - Etc
- **Entity Resolution**:
    - Combine multiple identities into one profile using fuzzy matching.

---

2. Phishing Campaign Generator

- **Base Templates**:
    - Password reset
    - Fake invoices
    - Executive impersonation
    - Etc
- **Adaptive Fields**:
    - Personalize sender names
    - Include company-specific references
    - Etc

---

3. NLP Text Personalizer

- **AI Models**:
  - GPT-2 / GPT-3 / fine-tuned transformers, etc.
- **Custom Text Generation**:
  - Insert individual recipient's name
  - Contextual adjustments ("Dear John, regarding your AWS bill...")
  - Etc
- **Tone Styles**:
  - Formal, informal, urgent

---

4. Campaign Launcher

- **Delivery**:
  - SMTP servers for email
  - SMS gateways for mobile phishing (smishing)
- **Monitoring**:
  - Tracks opens, clicks, submissions.

---

5. Interaction Logger

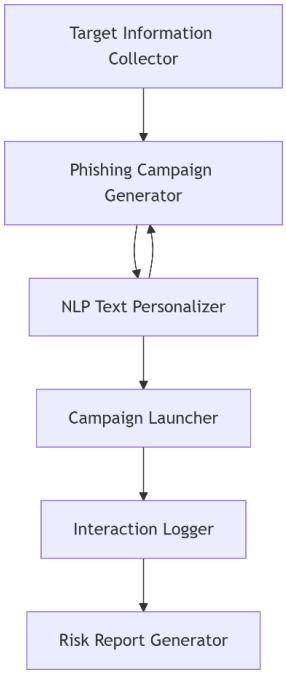
- **Tracking**:
  - Link click events (UUID per user)
  - Credentials entered (recorded securely for reporting)
  - Attachment downloads
  - Etc

---

6. Risk Report Generator

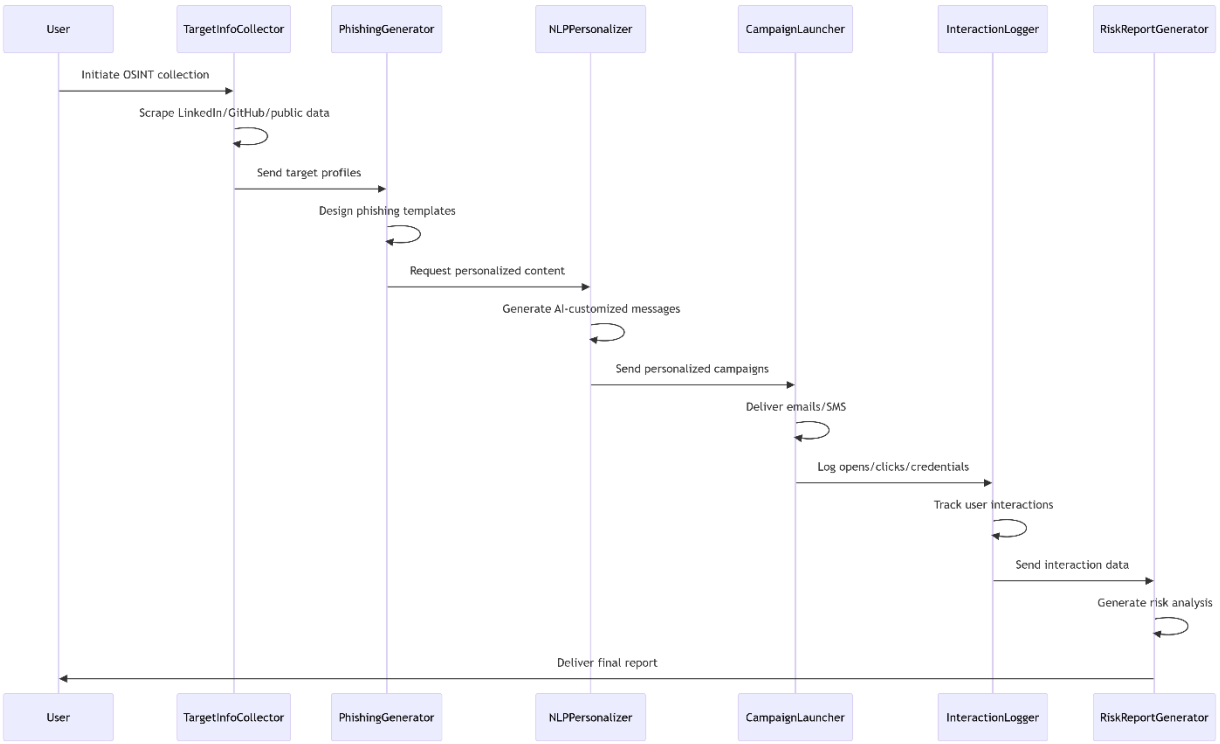
- **Report Includes**:
  - Which targets interacted
  - Timeline of events
  - Recommended user training improvements
  - Etc

Component Diagram



- The **Target Information Collector** feeds OSINT data to the **Phishing Campaign Generator**, which designs templates.
- The **NLP Text Personalizer** refines messages using AI (e.g., GPT) and sends them to the **Campaign Launcher**.
- The **Campaign Launcher** executes the campaign and logs interactions via the **Interaction Logger**.
- The **Risk Report Generator** aggregates interaction data to produce a security risk report for the **User**.
- The **NLP Text Personalizer** also feeds back into the **Phishing Campaign Generator** for iterative template refinement.

Sequence Diagram



- The **User** starts the process, triggering OSINT data collection.
- The **Phishing Campaign Generator** creates templates, which the **NLP Text Personalizer** customizes using AI.
- The **Campaign Launcher** sends personalized phishing attempts and logs interactions.
- The **Interaction Logger** tracks responses (clicks, credentials, etc) and shares data with the **Risk Report Generator**.
- The **Risk Report Generator** compiles results into an actionable report for the **User**, highlighting vulnerabilities and training needs.

Detailed Project Description: AI-Powered Social Engineering Simulation Framework

An AI-driven framework for simulating social engineering attacks. The framework identifies human vulnerabilities by creating and launching personalized phishing campaigns, tracking interactions, and generating actionable risk reports.

---

1. System Overview

The framework simulates phishing and smishing (SMS phishing) attacks using AI to personalize messages and OSINT (Open-Source Intelligence) data. It assesses an organization's susceptibility to social engineering and provides recommendations for improving security awareness.

Key Components

1. **Target Information Collector**
2. **Phishing Campaign Generator**
3. **NLP Text Personalizer**
4. **Campaign Launcher**
5. **Interaction Logger**
6. **Risk Report Generator**

---

2. Component Design & Implementation

2.1 Target Information Collector

Functionality:

- Gathers public data about targets (employees, organizations, etc) from social media, GitHub, professional platforms, etc.

Implementation Steps (e.g.):

1. **OSINT Tools**:
    - **LinkedIn/Public Social Media**: Use `Scrapy` or `Selenium` to scrape profiles (names, job titles, email patterns, etc).
    - **GitHub**: Extract repositories, commit history, and email addresses using the GitHub API.
    - **Google Dorking**: Find exposed documents (e.g., `site:company.com filetype:pdf`).
    - **Etc**
2. **Data Structuring**:
    - Normalize data into profiles:

```
{
  "name": "John Doe",
  "position": "Software Engineer",
  "email": "john.doe@company.com",
  "social_links": ["linkedin.com/johndoe"],
  "tech_stack": ["Python", "AWS"]
}
```

3. **Ethical Compliance**:
    - Only collect publicly available data.
    - Anonymize data unless explicit consent is obtained.

Output:

- Structured profiles of targets.

Tools (e.g.):

- `Scrapy`, `Selenium`, GitHub API, Google Custom Search JSON API, etc.

2.2 Phishing Campaign Generator

Functionality:

- Designs phishing templates tailored to target profiles.

Implementation Steps (e.g.):

1. **Template Types**:
    - **Password Reset**: "Your account requires immediate verification."
    - **Fake Invoice**: "Payment overdue for AWS services."
    - **Executive Impersonation**: "Urgent request from CEO."
    - **Etc**
2. **Dynamic Fields**:
    - Insert placeholders for personalization (e.g., `{{name}}`, `{{company}}`).
3. **Template Storage**:
    - Store templates in YAML/JSON for easy modification:

```yaml
templates:
  - type: "invoice"
    subject: "Urgent: Payment Required for {{company}} AWS Services"
    body: "Dear {{name}}, your invoice #{{invoice_id}} is overdue...
"
```

Output:

- Phishing email/SMS templates with placeholders.

2.3 NLP Text Personalizer

Functionality:

- Customizes messages using AI to increase believability.

Implementation Steps (e.g.):

1. **Model Integration**:
    - Use OpenAI's GPT-3.5/4 or Hugging Face's `transformers` for text generation.
    - Example prompt:

```
prompt = f"Generate a phishing email targeting {name}, a {role} at {
company}, about a fake AWS invoice."
```

2. **Personalization**:

    - Replace placeholders with target-specific data (e.g., name, role, company projects, etc).
3. **Tone Adjustment**:
    - Use AI to mimic formal, urgent, or casual tones.

Output:

- Personalized phishing messages ready for deployment.

Tools (e.g.):

- OpenAI API, `transformers` library, `spaCy` for entity recognition, etc.

2.4 Campaign Launcher

Functionality:

- Sends phishing emails/SMS ethically and tracks delivery.

Implementation Steps (e.g.):

1. **Email Setup**:
    - Use SMTP services (e.g., SendGrid, AWS SES) with spoofed sender addresses.
    - Example using Python's `smtplib`:
      ```
      import smtplib
      server = smtplib.SMTP('smtp.sendgrid.net', 587)
      server.sendmail("ceo@company.com", target_email, personalized_messag
      e)
      ```
2. **SMS Setup**:
    - Use Twilio API for smishing:
      ```
      from twilio.rest import Client
      client = Client(account_sid, auth_token)
      client.messages.create(body=message, from_=twilio_number, to=target_
      phone)
      ```
3. **Ethical Safeguards**:
    - Include visible disclaimers (e.g., "This is a simulated phishing test").
    - Limit campaign scope to authorized targets.

Output:

- Deployed phishing campaigns with tracking UUIDs.

Tools (e.g.):

- SendGrid, Twilio, AWS SES, `smtplib`, etc.

2.5 Interaction Logger

Functionality:

- Tracks clicks, credential submissions, and downloads.

Implementation Steps (e.g.):

1. **Click Tracking**:
    - Embed UUIDs in links (e.g., `https://company.com/track?id=123`).
2. **Credential Capture**:
    - Use mock login pages (e.g., Flask/Django) to log submissions without storing real credentials.
3. **Data Security**:
    - Encrypt logs and store them in a secure database (e.g., PostgreSQL with AES encryption).

Output:

- Logs of user interactions (clicks, submissions, downloads).

Tools (e.g.):

- Flask/Django for mock pages, PostgreSQL, UUID4, etc.

2.6 Risk Report Generator

Functionality:

- Analyzes interaction data and generates risk reports.

Implementation Steps (e.g.):

1. **Metrics Calculation**:
    - **Click Rate**: `(Clicks / Emails Sent) × 100`
    - **Submission Rate**: `(Credentials Submitted / Clicks) × 100`
2. **Report Content**:
    - High-risk targets (e.g., users who submitted credentials).
    - Timeline of interactions.
    - Training recommendations (e.g., "Implement phishing awareness workshops").
3. **Visualization**:
    - Use `Matplotlib` or `Plotly` for charts (e.g., click rates by department).

Output:

- PDF/HTML report with risk analysis and recommendations.

Tools (e.g.):

- Pandas for data analysis, `Jinja2` for HTML templates, `WeasyPrint` for PDF, etc.

---

3. Technology Stack (e.g.)

- **OSINT**: Scrapy, Selenium, GitHub API, etc.
- **NLP**: OpenAI API, Hugging Face Transformers, etc.
- **Campaign Delivery**: SendGrid, Twilio, AWS SES, etc.
- **Backend**: Flask, PostgreSQL, Docker, etc.
- **Reporting**: Pandas, Matplotlib, Jinja2, etc.

---

4. Evaluation & Validation

1. **Effectiveness Metrics**:
    - **Click-Through Rate (CTR)**: Measure engagement with phishing links.
    - **False Positives**: Track users who reported the phishing attempt.
    - **Etc**.
2. **Ethical Testing**:
    - Run simulations in a controlled environment with informed participants.
3. **Baseline Comparison**:
    - Compare AI-personalized campaigns vs. generic templates.

---

5. Development Roadmap

1. **Phase 1**: Build OSINT scraper and phishing template engine.
2. **Phase 2**: Integrate NLP personalization and campaign launcher.
3. **Phase 3**: Develop interaction logging and reporting.
4. **Phase 4**: Conduct pilot tests and refine based on feedback.

---

6. Challenges & Mitigations (optional)

- **Ethical/Legal Compliance**: Obtain explicit consent and anonymize data.
- **Spam Filters**: Use domain warming techniques and rotate sender IPs.
- **AI Over-Personalization**: Ensure messages remain within ethical boundaries.

---

7. Glossary

- **OSINT**: Open-Source Intelligence
- **NLP**: Natural Language Processing
- **SMTP**: Simple Mail Transfer Protocol

- **UUID**: Universally Unique Identifier
