Phase 1: Research & Planning (1-20)

1. Understand CAPTCHA Limitations – Analyze issues with existing CAPTCHA systems like reCAPTCHA, hCaptcha, and FunCAPTCHA, focusing on security vulnerabilities, user friction, and accessibility concerns.
2. Study Automated Bot Behavior – Research bot frameworks (e.g., Puppeteer, Selenium, Playwright, PhantomJS) and how they attempt to bypass existing CAPTCHA mechanisms.
3. Define Key Performance Metrics – Establish clear success benchmarks, including false positive/negative rates, response time, bypass success rate, and user experience impact.
4. Identify Passive Signal Categories – Determine which behavioral signals to monitor, including mouse dynamics, scroll patterns, keystroke timing, and device fingerprinting.
5. Create an Attack Model – Develop a comprehensive catalog of bot attack techniques (e.g., script automation, human-assisted botting, adversarial ML attacks).
6. Select the Machine Learning (ML) Approach – Choose between supervised, unsupervised, semi-supervised, or hybrid learning models based on data availability and adaptability.
7. Evaluate Privacy & Legal Compliance – Ensure compliance with GDPR, CCPA, and other privacy laws by anonymizing sensitive user data and allowing opt-out options.
8. Benchmark Against reCAPTCHA – Compare proposed models against existing CAPTCHA services in terms of accuracy, performance, and bot detection capability.
9. Design Initial System Architecture – Define data pipelines, real-time inference models, and decision-making processes for classifying humans vs. bots.
10. Plan Data Storage & Security – Implement encrypted databases and secure storage for interaction logs, ensuring minimal data retention.
11. Define API and Integration Methods – Plan how websites and applications will integrate the passive CAPTCHA system (e.g., JavaScript SDKs, backend APIs).
12. Investigate Edge Computing Feasibility – Explore client-side AI processing for reduced latency and improved scalability.
13. Study Adversarial ML Risks – Research adversarial attack vectors that could manipulate the model and devise countermeasures.
14. Choose Development Technologies – Select appropriate programming languages (Python, TensorFlow/PyTorch for ML, Flask/FastAPI for backend, JS for frontend).
15. Develop a Prototyping Roadmap – Establish iterative development phases for building and testing each module of the system.
16. Prepare Test Environments – Set up local, cloud, and edge-based testing environments for controlled experiments.
17. Gather Sample Datasets – Collect real user behavior data from open datasets, simulations, or voluntary participants.
18. Plan Data Annotation Strategy – Design a semi-automated labeling system to differentiate human and bot behaviors.
19. Establish Model Evaluation Criteria – Define metrics for precision, recall, F1-score, and AUC-ROC to compare different models.
20. Draft Initial Documentation – Prepare technical documentation covering system architecture, data flow, and compliance considerations.

Phase 2: Data Collection & Feature Engineering (21-40)

21. Set Up a Data Pipeline – Develop a real-time data collection system to record user interactions securely.
22. Implement Data Logging Mechanisms – Store session data, user interactions, and network requests for analysis.
23. Collect Human Interaction Data – Record real user behavior, including typing speed, scrolling tendencies, and movement fluency.
24. Simulate Automated Bot Interactions – Use scripted bots to generate synthetic attack traffic for training and validation.
25. Extract Mouse Movement Features – Capture velocity, acceleration, and jitter in mouse movements to differentiate between humans and bots.
26. Analyze Click Timing Variance – Study how users click elements to detect unnatural patterns.
27. Monitor Keystroke Dynamics – Capture key press durations and transitions to build passive biometric authentication layers.
28. Evaluate Scroll & Touch Behavior – Track scrolling speed, touch pressure, and trajectory to identify human-like fluidity.
29. Implement Browser Fingerprinting – Collect non-invasive fingerprinting data such as WebGL, timezone, installed fonts, and CPU/GPU configurations.
30. Analyze Sensor Data on Mobile Devices – Utilize accelerometer and gyroscope readings to measure natural device movements.
31. Monitor API Call Patterns – Detect abnormal request frequency indicative of scripted automation.
32. Analyze Network & IP Data – Use IP reputation scoring, ASN identification, and geolocation consistency checks.
33. Study Session Timing Distributions – Compare time spent on different pages to distinguish between human and automated navigation.
34. Implement TLS Fingerprinting – Track SSL handshake variations between browsers and bots.
35. Capture Passive Audio Cues (Optional) – Research subtle environmental audio patterns for additional human verification.
36. Label Data for Model Training – Establish ground truth for supervised learning by correctly labeling collected data.
37. Normalize & Augment Data – Preprocess interaction data for consistency and improve model robustness with synthetic data augmentation.
38. Implement Feature Selection – Use feature importance techniques (e.g., SHAP values) to identify the most effective classification signals.
39. Prepare Data for ML Models – Split data into training, validation, and test sets for iterative model improvements.
40. Develop a Data Security Plan – Ensure proper encryption, pseudonymization, and access controls.

Phase 3: Model Development & Training (41-60)

41. Select Initial ML Models – Experiment with decision trees, random forests, and logistic regression for baseline comparisons.
42. Train on Small Datasets – Perform quick iterations on small datasets for faster feedback.
43. Validate Model Performance on Live Traffic – Test real-world data for accuracy and false positive rates.
44. Introduce Deep Learning Models – Implement CNNs/RNNs to detect sequential interaction patterns.
45. Apply Unsupervised Anomaly Detection – Train Isolation Forests and autoencoders to detect outliers.
46. Develop an Adaptive Learning Pipeline – Automate model updates as new threats emerge.
47. Fine-Tune Hyperparameters – Optimize learning rate, dropout, and regularization.
48. Implement Real-Time Model Deployment – Set up low-latency inference engines.
49. Benchmark Against Human Users – Compare detection rates on verified human testers.
50. Train with Adversarial Bots – Introduce adversarial training to defend against evolving bot strategies.

Phase 4: Advanced Feature Engineering & Security Enhancements (61-80)

51. Introduce Temporal Analysis – Study session behaviors over time.
52. Implement Graph-Based Detection – Model relationships between interactions.
53. Detect Low-Interaction Bots – Identify bots with minimal interaction footprints.
54. Develop Real-Time Bypass Detection – Monitor and flag sudden success rate drops.
55. Enhance Edge AI Processing – Deploy lightweight AI models on clients.
56. Optimize Model for IoT & Smart Devices – Adapt detection to non-traditional devices.
57. Simulate AI-Powered Bot Attacks – Train against state-of-the-art AI-driven bot behaviors.
58. Introduce Self-Healing Models – Automate retraining without manual intervention.
59. Monitor Botnet Network Activity – Track correlated logins across multiple devices.
60. Implement Multi-Factor Adaptations – Adjust challenge levels based on risk scores.

Phase 5: Deployment & Continuous Improvement (81-100)

81. Deploy in Controlled Environments – Test on small-scale production.
82. Develop a Rollback Mechanism – Ensure safe updates.
83. Monitor Latency & Performance – Optimize real-time classification.
84. Implement Auto-Scaling Infrastructure – Handle high traffic loads dynamically.
85. Set Up Continuous Security Audits – Identify new attack patterns.
86. Collect User Feedback – Optimize UX for seamless experience.

87. Refine Detection for Zero-Day Bots – Improve adaptability.
88. Optimize for Mobile Browsers – Adjust for mobile UI/UX.
89. Ensure Minimal False Positives – Prevent user lockouts.
90-100. Continuous Model Refinement & Threat Adaptation – Maintain long-term efficiency.